Sensor Network Modeling as a Service

Anca Daniela Ionita, Florin Daniel Anton and Adriana Olteanu
Department of Automatic and Industrial Informatics, University Politehnica of Bucharest, Bucharest, Romania

Keywords: Cloud Services, Model as a Service, Sensor Network Modeling.

Abstract: Cloud Computing opens new possibilities of service provisioning for sensor networks, necessary as they become more pervasive and distributed. This paper introduces a Model as a Service approach and a modeling language specifically defined for representing sensor network architecture, based on a four-phased method. It describes the metamodel and the correspondent environment for graphical modeling, with examples of sensor network models for road traffic monitoring. The sensor modeling environment was integrated on a private Cloud platform, within a virtual machine template, to provide sensor network modeling as a service, which is currently available for our university students. This serves as a foundation for delivering new services based on the interpretation of the resulted sensor network architecture models.

1 INTRODUCTION

Sensor networks are present in plenty of distributed applications characteristic to the Internet of Things (IoT) era (Flammini and Sisinni, 2014), and they have a great potential to get advantage of the deployment in Cloud Computing environments for various goals, like data integration, advanced processing, or global interoperability. This paper approaches a kind of Cloud services based on the Model as a Service (MaaS) concept, whose applicability for sensor networks may be considerably extended. MaaS is derived from the Model Web and Everything as a Service concepts, as shown in (Roman et al., 2009). Roman et al. give an example of MaaS application for providing access to oil spill models, with the purpose to support decisions and increase the preparedness to this type of hazard. The work outlines important challenges related to the explicit manifestation of the model, separate from the rest of the software, as well as the importance of performing model updates based on data collected by sensors that might indicate a change in the environmental conditions. Researchers discovered a big potential for MaaS in domains like geosciences (Li et al., 2014) and, generally, whenever the models engage big volumes of data and must be executed repeatedly. Thus, MaaS may be considered a means to get models more accessible and widely used in executable settings and not just as simple documentation.

This paper presents a MaaS-enabled environment for sensor network architecture modeling, offered as a service in an IBM CloudBurst platform, and used for educational applications. The graphical modeling language was specifically conceived for sensor networks and is accompanied by an editor and by interpreters developed within the Generic Modeling Environment (GME) (GME, 2017) and migrated to MaaS. The language resulted from the definition of a metamodel and the customization of a concrete notation. It is useful for representing models of sensor network architecture, which may subsequently be interpreted for reconfiguration and visualization of ad hoc networks, or for generation of standard descriptions in SensorML. It was elaborated based on the analysis of several sensor taxonomies and on the study of multiple sensor networks presented in the scientific literature.

The remainder of the paper is structured in the following sections: Section 2 presents the Model as a Service concept; Section 3 describes our sensor network modeling environment and the way it was integrated in a Cloud Computing environment and offered as a service; Section 4 offers details about the realization of the sensor network modeling language, whose editor is delivered as a service. In Conclusion we discuss other services that may be provided as MaaS, starting from existing and future software, capable to automatically interpret sensor network models defined in this environment.

Ionita, A., Anton, F. and Olteanu, A.
Sensor Network Modeling as a Service.
DOI: 10.5220/0006670903460353
In *Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER 2018)*, pages 346-353
ISBN: 978-989-758-295-0

2 MODEL AS A SERVICE

Model as a Service is a form of XaaS (Everything as a Service). A more detailed classification introduces the concept of Model and Simulation as a Service (MSaaS) (Cayirci, 2013), with three types of services: Modeling as a Service, Model as a Service, and Simulation as a Service. Another idea resulted from (Cayirci, 2013) is that it is possible to develop composite MSaaS, located either on a single datacenter, or on multiple ones.

A variant of MaaS called Model Web aims at creating "webs of interacting models", and an initiative is proposed within the earth observation community (Nativiti et al., 2013). For instance, Model Web applications were introduced for climate change models (Skøiena et al., 2013), and the solution was also found suitable to give access to non-homogeneous models for geo-analysis (Yuea et al., 2016).

Model Web is considered an implementation style derived from Systems of Systems (SoS) based on semantic web. Guided by principles like open access and publishing new resources in a simple way, the idea is to support interactions between models and between modelers. The Model Web is rather a precursor of MaaS, but it is still confronted with strong interoperability challenges.

The concept of MaaS is also associated with quantitative analysis (Yung Rowe, 2014). Due to the large-scale data resulted from the Internet of Things and social media, it may be useful to subscribe to MaaS, and to use the model outputs for decision support. MaaS is considered an agile approach to deliver data services in the context of Big Data and of heterogeneous lakes of data that may be structured, semi-structured or non-structured (Piscopo, 2014).

An example of geospatial MaaS is ArcGIS Online (Esri, 2017), which offers services related to geoinformation models, including mapping and visualization. Generally, the use of MaaS in geosciences exonerates users not only from general software maintenance, but also from setting up models, which may be a repetitive and very complex task (Li et al., 2017). It also offers powerful resources in terms of data storage and computational speed. Li et at. proposed a framework where a model registry gives access to a repository of climate, polar and environmental models, based on a Cloud Computing platform. They refer to executable models, whose results can be obtained through a web interface.

3 SENSOR NETWORK MODELING IN CLOUD

3.1 Modeling Environment

This paper introduces a modeling environment realized within Generic Modeling Environment and deployed in Cloud for the restricted use of our students.

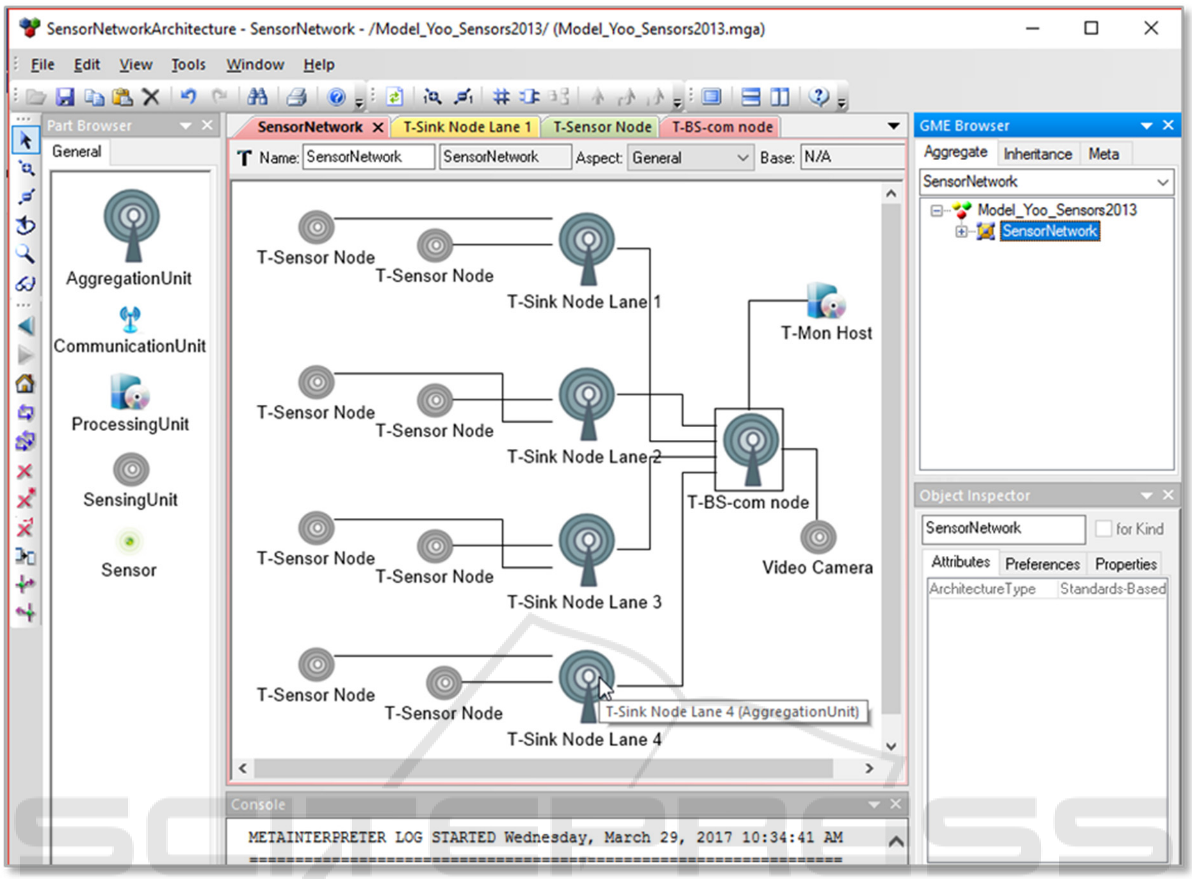
First, GME was used to represent the *Sensor Network* language in an abstract way, as a metamodel (further described in Section 4.3). Then, we introduced a concrete notation for each modeling element, and afterwards we used the GME metamodel interpreter to generate an environment configured for our language. The resulted modeling language is graphical, i.e. graph-based, and the model editor uses specific modeling elements, whose semantics are defined within the specific application domain, i.e. sensor networks.

For configuring the model visualization, it is also possible to add one or more "aspects", which allow one to see just a part of the modeling elements, in order to reduce the model complexity and to show what is relevant for a particular point of view. For the *Sensor Network* language, we defined "aspects" that show parts of the model concerning *Communication, Power, Memory,* and *Sensors* respectively.

Thus, the modeling environment allows one to represent sensor network models conforming to the metamodel created in GME (see Figure 1). The next paragraph explains the environment, but the model illustrated in Figure 1, representing the architecture of a sensor network for road traffic monitoring, is presented in detail in Section 4.4.

The central pane represents a model drawn with the elements available on the left side (Part Browser), corresponding to the classes defined in the metamodel, but configured with specific notations (i.e. icons). The environment shows the editable attributes of a selected model element on the lower right side (Object Inspector).

With double-click on an element, a new drawing pane may open, and one can edit a diagram for showing the structure inside that element. When the mouse is placed over a modeling element, one can see the kind of that element (from the metamodel) - in parentheses - e.g. the kind *AggregationUnit* for *T-Sink Node Lane 4,* in Figure 1.



```
METAINTERPRETER LOG STARTED Wednesday, March 29, 2017 10:34:41 AM
================================================================
```

Figure 1: Modeling environment representing a sensor network for road traffic monitoring.

3.2 Modeling Service in Cloud

The previously described modeling environment for sensor networks was deployed in Cloud, in Virtual Machines (VM) replicated in respect with the number of students subscribed to a Model Driven Engineering course.

3.2.1 Implementation

The implementation used an IaaS (Infrastructure as a Service) Cloud system based on IBM Service Delivery Manager (ISDM), which has the same hardware configuration as the IBM CloudBurst 2.1 medium configuration (Lemos et al., 2012):

- 14 Blades HS22 (72 GB RAM)
- 28 TB Raw Data Space.

From the point of view of software, ISDM is more flexible and customizable, being able to integrate different types of servers (servers with Intel or Amd CPUs, Power systems, or even Mainframe) (Huche et al., 2012).

In order to implement the solution, a virtual machine was created, installed with the required software and transformed into a template. The template was then discovered by the TSAM component (Tivoli Service Automation Manager) and registered in the Cloud service catalog using the Self Service Interface offered by the Cloud system. From this point on, the service implemented by the virtual machine was available and ready to be requested and deployed into one or more projects.

The service can be offered using two methods:

- *As access to the modelling environment*
- *As a virtual machine for each service request in the form of an IaaS service*
  In this way, the user can benefit of the entire virtual machine, not only of the modeling software; after the service request has been processed and the virtual machine is available, one can connect using Remote Desktop Connection, and the username and password provided by the system

The first method, i.e. the access to the modeling environment, can be applied in two different ways:

- For each service request, a virtual machine can be created, and each user will use his or her own VM, like in the previous IaaS solution; however, each VM is customized such that, each time a connection is made, the explorer process is killed, and the modeling software is launched; thus, the user only has access to the requested software, so one can consider that, by using an IaaS system, the modeling service is offered as Software as a Service (SaaS)
- The other option is to deploy a VM for a predefined number of users. When a user is requesting a service, the VM is only reconfigured by TSAM, and a new user is added to the virtual machine; the VM configuration can be identical to the first solution, for offering access only to the software (killing explorer and starting the modeling software only) and can be installed with VNC (Virtual Network Computer), which offers the possibility to connect to the user session using a Java-enabled web interface. In this way, one can connect to the service only using a web browser.

3.2.2 Service Request & Provisioning Process

When requesting a service, the user will connect to a web interface, which is the Cloud Self Service Interface; the user authenticates with a registered username and a password and then, selecting the interface options, he or she can request a service in the form of a project, including the number of virtual machines to be created based on a template. The service request is filled with the following information: the name of the project, the description, the period of time when the project will be available, the name of the virtual machine template used to deploy the virtual machines included in the project, the additional software that will be installed, the number of virtual machines included in the project and their hardware configuration (number of virtual CPU cores, the number of tenths of physical CPU cores, RAM, swap space and HDD capacity).

After the user submits the request, the service request is sent to the Cloud administrator to be processed. If the Cloud administrator approves the service request (SR), the SR is sent from the Self Service Interface to Tivoli Service Request Manager (TSRM) using REST API; the requester of the service is also notified that the request has been approved. Then, TSRM places the request into a queue and waits for the scheduled date and time.

When the scheduled date and time arrive, the provisioning of virtual servers begins; if the provisioning is successful, then the user who requested the service is notified by email and receives information on how to access the VMs (VM hardware configuration, IP address, username, password). From this point forward, the VMs can be accessed and used; if the user wishes to modify the project (for example the RAM capacity of VMs, or the project schedule) the request is sent to the administrator to be approved and then the project is modified. If the project is canceled or the project end date arrives, then the virtual machines are powered off and then erased, and the resources are freed; a copy of the VM image can be done if the user requested it when the project was created or canceled. The user is notified at the end of the project, and also two days before the end date arrives, so he or she can modify the project schedule if needed.

3.2.3 Testing the Solution

From the preliminary tests, we noticed that, for a template having a size of 10 GB and a project with 20 VMs, the process of offering the service takes about 22 minutes. Using the same template into a project with 100 VMs, the process only requires 63 minutes.

These performances are obtained because the VM template has a reduced size, and also because the project provisioning was done without processing other service requests in parallel. The small difference between the two tests exists because, in the first stage of project provisioning, TSAM checks the status of Cloud resources and has an intensive interaction with the Cloud hypervisor (VMware) and the Cloud database (DB2), which takes about 10 minutes, before the VM template cloning starts; the cloning process takes lesser time because the size of the template is reduced, but also because the cloning is done in parallel on eight threads.

4 SENSOR NETWORK MODELING LANGUAGE

4.1 Background

The service provisioning related to sensors is supported by OGC (Open Geospatial Consortium) with the Sensor Observation Service (SOS), for registering sensors and giving access to the data measured by them through web services (OGC, 2017). SOS does not support a service for modeling the architecture of the sensor networks that acquire the data.

For sensor modeling there is a standard called - Sensor Model Language (SensorML) - focused on data models, position, description of the observed phenomena, and processes (OpenGIS, 2007). It does not support the description of a network topology.

In the Model-Driven Engineering (MDE) community, there have been multiple attempts to model sensor networks. A comprehensive study on Wireless Sensor Networks (WSN) was presented in (Malavolta and Muccini, 2014), where several criteria were used for comparison: capacity to generate code, type of modeling language, aspects under consideration (e.g. structure, behavior, mobility, power consumption). Another survey, based on almost two thousand papers, was presented in (Essaadi et al., 2017).

4.2 Method

Based on the general theoretical background related to sensor networks, we applied an iterative method for developing the graphical modeling language.

Phase 1 is the analysis of examples of sensor networks, for which one identifies the main concepts (terms) that describe the physical and the computational architecture.

Phase 2 is focused on the creation of a metamodel, seen as the abstract syntax of the future language specific for modeling sensor networks. The aim is that any concept or relationship related to the real-world examples selected at *Phase 1* can be represented as an instance of a metamodel class. The metamodel also considers existing classifications and taxonomies regarding measurements, sensor characteristics, or Wireless Communication Networks (WCN) in general (see Section 4.3).

Phase 3 generates an editor of models conforming to the metamodel, introducing customized notations, and adding the abstractions that allow one to visualize different views and hide the rest of the model details – if the users need this. As explained in Section 3.1, in GME they are called "aspects".

Phase 4 was introduced for the metamodel validation. It consists of using the editor to represent models for all the examples analyzed at *Phase 1*.

4.3 Metamodel

Generally, sensor networks are represented informally for visualization purposes and formally for hypertext-based design, which requires programming skills. The language proposed here aims at reconciling the two approaches, i.e. obtaining an intuitive representation that is also compatible with a precise metamodel, which allows model execution within a program, and automatic transformation to other paradigms, for interconnection purposes.

Our concern was the sensor network architecture, with a focus on structural aspects and less details about the behavioral ones. Moreover, the purpose was to elaborate a little language, and to raise the abstracting level as much as possible, yet to be capable to model a large variety of real-life sensor networks. This fits to the existing trend of developing little Domain-Specific Languages (DSL) (Erwig and Walkingshaw, 2014), and then compose them according to the needs of large-scale applications (Estublier and Ionita, 2005).

The metamodel of the sensor network modeling language is composed of a core (named *Sensor Network*) and four other parts, defined according to the classification of WCN devices: *Communication, Power, Memory,* and *Sensors* (Cheekiralla and Engels, 2005). A sensor network may be composed of sensing, aggregation, communication, and processing units, and of sensors connected to each other (see Figure 2).

We define *ProcessingUnit* as an element that may contain *SoftwareUnit* elements, dependent on one another. A software unit in our language represents any piece of software, be it a physical or a logical artifact, e.g. an architectural layer, a software agent, a service, or even an algorithm.

A *SensingUnit* is an *AggregationUnit* that contains sensors, so it can perform acquisition of data. An *AggregationUnit* is used for gathering data originated from sensing units or from other aggregation units, and for transmitting them forward, but it may also perform some data processing. For this reason, it contains objects of the following kinds: *CommunicationUnit*, *ProcessingUnit*, and *PowerUnit*, connected to each other.

A *Sensor* contains the attribute *MeasurandType*, having one of the following values: *Acoustic, Biological, Chemical, Electric, Magnetic, Mechanical, Optical, Radiation,* and *Thermal*, as in the classification from (White, 1987).

The type of *SensorNetwork* is specified through an enumeration attribute: *Standards-Based, Internet-Connected, Context-Aware, Agent-Based, Service-Oriented, Secure and Fault-Tolerant, Vehicle-Based, Habitat-Monitoring,* and *Multi-Owner*, following the taxonomy proposed in (Fokum et al., 2008).

The sensors characteristics are given in conformity with the abstract syntax from the *Sensor* part of the metamodel, with classes like: *Operating Range, Power Supply, Output, Accuracy, Sensitivity* - according to (Kalantar-zadeh, 2013).

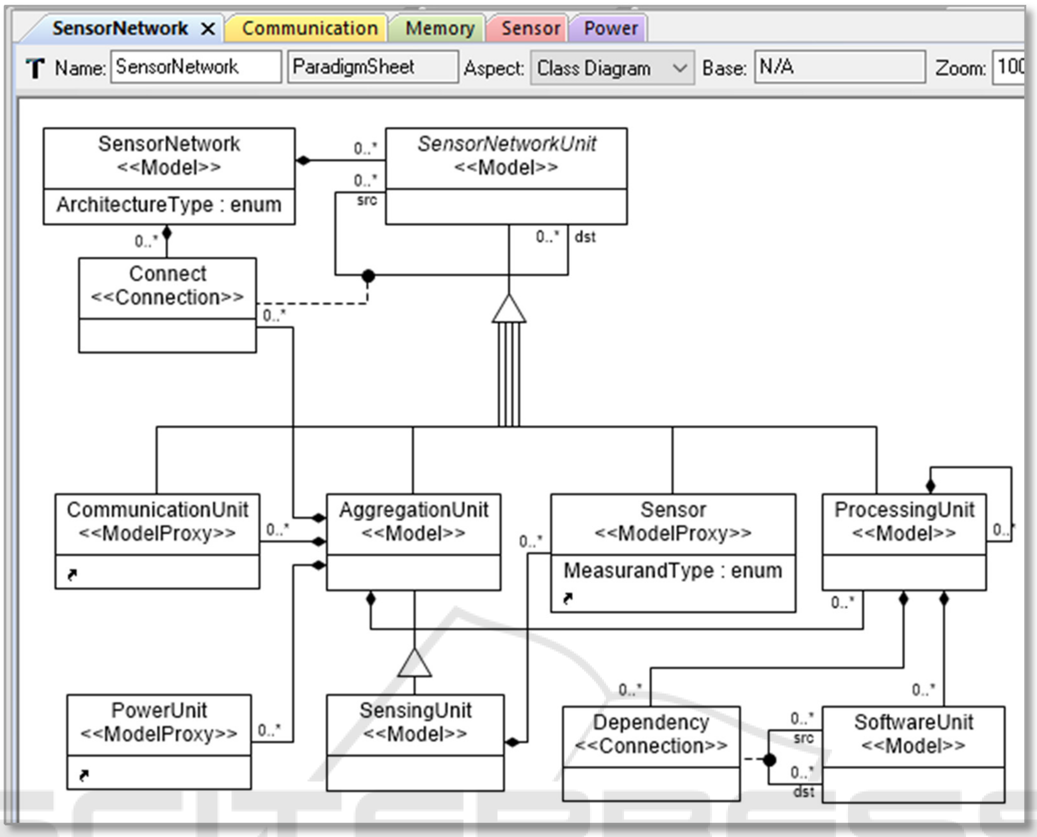


Figure 2: Part of the metamodel.

The metamodel was specified in the metamodeling language supported by Generic Modeling Environment. GME offers a metamodeling language whose main objects are: *Atom* (indivisible, yet characterized by attributes), *Model* (a composite of other modeling elements) and *Connection* (to represent links between model elements). The notation adopted in GME is very similarly to the UML (Unified Modeling Language) class diagrams, as noticed from Figure 2. As in UML, the triangle is used for generalization relationships, and the line ended with a diamond for compositions

4.4 Example of Model

According to the method described in Section 4.2, we performed multiple iterations for elaborating the metamodel and testing the use of the modeling editor. During one of these iterations, we studied a series of systems where wireless sensor networks were used for monitoring vehicle traffic. A comprehensive survey on this topic is presented in (Nellore and Hancke, 2016). Thus, in *Phase 1* from our method, we studied multiple sensor networks described in the scientific literature, then in *Phase 4* we modeled them with the specific modeling environment from Section 3.1.

One of these systems is meant to detect vehicles on a four-lanes road, using a Telematics Sensor Network (TSN) (Yoo, 2013).

Figure 1, also illustrating the modeling environment, contains the upper part of the TSN model, containing the following objects of kind *AggregationUnit*:

- four sink nodes, one for each lane, and
- one base station, named *T-BS-com node*, which has a video camera and transmits the data to a *ProcessingUnit* named *T-Mon Host*.

The names from this model are the ones given in (Yoo, 2013). Data are collected from two telematics sensor nodes per lane, named *T-Sensor Node*, instantiated from *SensingUnit*.

We also added details for some of these elements, according to the description from Yoo's article. For example, the elements inside a sensor node are:

- MSP430 MCU, micro-controller, of kind *ProcessingUnit*
- CC2590 RF Amplifier, of kind *ProcessingUnit*

- CC2420 Transceiver, of kind *CommunicationUnit* and
- HMC1041Z MR, of kind *Sensor*, with the *MeasurandType* attribute set to *Magnetic*.

All the objects previously mentioned have the type *Model* in the GME language, therefore it is possible to describe their inner structure. For instance, the sensor is described with three groups of characteristics, as specified in (Honeywell, 2017): *Bridge Elements, Set/Reset Strap,* and *Offset Straps*. The specific modeling environment allows one to specify values for each characteristic, e.g. for one that represents an *OperatingRange*, it is possible to set the conditions, the minimum, typical and maximum values, and their units. The values given in our model are those from the technical specification of the sensor.

5 CONCLUSIONS

The paper presents a modeling language and a graphical modeling environment for sensor network architecture, provided as an educational service on a private Cloud. The approach can be compared to WebGME, realizing Domain Specific Modeling Languages (DSML) in the Cloud (Maróti et al., 2014). Thus, our service is exclusively dedicated to modeling, not to metamodeling (as WebGME) and its users are only supposed to use the sensor network modeling environment AS-IS, and not to upgrade it, or configure it themselves. Any modifications at the metamodel level need, in our process, the creation of a new template for generating other virtual machines. The migration of models to the new VMs is clearly possible, but with the inherent limitations imposed by GME; for example, if a metamodel class is eliminated from the metamodel, a model containing an instance of that class cannot be opened with the new version of the sensor network editor. The modeling environment has evolved during our study and it is still subject to change, according as new examples are studied and modeled. For example, the language was applied for modeling volcano monitoring networks, and it may be considered one of the multiple modeling paradigms (Ionita and Mocanu, 2015) required for dealing with the complexity of hazard management systems in general.

According to the Cayirci's classification presented in Section 2, our approach currently fits in the Modeling as a Service type (Cayirci, 2013), because it stands in the availability of a sensor network modeling environment, deployed on a Cloud platform. The templates we use also have some similarities to the solution from (Li et al., 2017), where the models are executed on virtual machines, based on an image. The MaaS engine presented there generally corresponds to a model interpreter.

The template containing the modeling environment may be used for creating virtual machines according to the needs of various educational programs in our university, to allow the remote access to graphical modeling of sensor network architecture. It also supports the definition of models that are potentially executable, therefore the environment is a foundation for providing more services by adding model interpreters, in the spirit of the Model as a Service approach.

Future work will focus on adding new services that provide information about sensor network complexity, transform the model to standard formats, integrate the data acquired by sensors, or evaluate the impact of sensor network evolution.

ACKNOWLEDGEMENTS

This work was funded by University Politehnica of Bucharest, through the "Excellence Research Grants" Program, UPB – GEX 2017. Identifier: UPB-GEX2017, Ctr. No. 44 /2017 (AU 11-17-13).

REFERENCES

Cayirci, E., 2013. Modeling and Simulation as a Cloud Service: A Survey. In *Proceedings of the 2013 Winter Simulation Conference*, R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, eds, pp. 389-400.

Cheekiralla, S., Engels, D.W., 2005. A functional taxonomy of wireless sensor network devices. In *2nd International Conference on Broadband Networks*. BroadNets.

Erwig, M., Walkingshaw, E., 2014. Semantics-Driven DSL Design. In *Computational Linguistics: Concepts, Methodologies, Tools, and Applications*, Volume I, IGI Global, pp. 251-275.

Essaadi F., Ben Maissa Y., Dahchour M., 2017. MDE-Based Languages for Wireless Sensor Networks Modeling: A Systematic Mapping Study. In El-Azouzi R., Menasche D., Sabir E., De Pellegrini F., Benjillali M. (eds), *Advances in Ubiquitous Networking 2. Lecture Notes in Electrical Engineering,* vol 397. Springer, Singapore.

Esri, 2017, Esri Managed Cloud Services [online], Available from: http://www.esri.com/arcgis/services/managed-cloud [Accessed 14 Oct. 2017].

Estublier, J., Ionita, A.D., 2005. Extending UML for Model Composition, In *Proceedings of Australian Software Engineering Conference (ASWEC)*, Publisher IEEE

Computer Society, March 2005, Brisbane, Australia, pp. 31-38.

Flammini, A., Sisinni, E., 2014. Wireless Sensor Networking in the Internet of Things and Cloud Computing Era, In *Procedia Engineering*, vol. 87, pp. 672-679.

Fokum, D.T., Frost, V.S., Mani, P., Minden, G.J., Evans, J.B., Muralidharan S., 2008. A Taxonomy of Sensor Network Architectures. In *Technical Report, ITTC-FY2009-TR-41420-08,* University of Kansas.

GME, 2017. Generic Modeling Environment, http://w3.isis.vanderbilt.edu/Projects/gme/ [Accessed 14 Oct. 2017].

Honeywell, 2017. One-Axis Magnetic Sensor HMC1041Z [online]. Available from: https://aerocontent.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Missiles-Munitions/HMC1041Z.pdf [Accessed 14 Oct. 2017].

Huche, T., Koohi, B., Lam, T.V., Reynolds, P., Swehla, S.M., Wain, J., Vetter, S., 2012. *Cloud Computing Infrastructure on IBM Power Systems: Getting started with ISDM*. IBM Redbooks.

Ionita, A.D., Mocanu, M., 2015. Multiple Modeling Paradigms Applied for Accidental Pollution Management. In *Environmental Engineering and Management Journal*, Issue 9, Volume 14.

Kalantar-zadeh, K., Sensors, 2013. *An Introductory Course*, Springer.

Lemos, A., Moleiro, R., Ottaviano, P., Rada, F., Widomski, M., Braswell, B., 2012. *IBM CloudBurst on System x.* IBM Redbooks.

Li, Z., Yang, Huang, Q., Liu, K., Sun, M., Xia, J., 2017. Building Model as a Service to support geosciences. In *Computers, Environment and Urban Systems,* Volume 61, Part B, Pages 141–152.

Malavolta, I., Muccini, H., 2014. A Study on MDE Approaches for Engineering Wireless Sensor Networks, In *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, Italy, pp. 149-157.

Maróti, M., Kecskés, T., Kereskényi, R., Broll, B., Völgyesi, P., Jurácz, L., Levendovszky, T., Lédeczi, Á., 2014. NextGeneration (Meta) Modeling: Web-and Cloud-Based Collaborative Tool Infrastructure. In *Proceedings of the 8th Multi-Paradigm Modeling Workshop,* Spain, pp. 41–60.

Nativiti, S., Mazzetti, P., Geller, G.N., 2013. Environmental Model Access and Interoperability: the GEO Model Web Initiative. In *Environmental Modelling & Software,* Volume 39, pp 214–228.

Nellore, K., Hancke, G.P., 2016. A Survey on Urban Traffic Management System Using Wireless Sensor Networks, In *Sensors*, 16 (2) 157.

OGC, 2017. Sensor Observation Service [online], Available from: http://www.opengeospatial.org/standards/sos [Accessed 14 Oct. 2017].

OpenGIS, 2007. Open Geospatial Consortium, OpenGIS® Sensor Model Language (SensorML) Implementation Specification, M. Botts Ed., Version: 1.0.0.

Piscopo, N., 2014. Agile Big Data and Many-Particle approach change Marketing and Sales effectiveness [online], In *Cloud Best Practices Network*. Available from: https://cloudbestpractices.wordpress.com/author/nucciopiscopo/ [Accessed 14 Oct. 2017]

Roman, D., Schade, S., Berre, A. J., Bodsberg, N. R., Langlois, J., 2009. Model as a service (MaaS). In *AGILE workshop: Grid technologies for geospatial applications,* Germany.

Skøiena, J.O., Schulza, M., Duboisa, G., Fisherb, I., Balmanc, M., Mayc, I., Tuamad, É.Ó., 2013. A Model Web approach to modelling climate change in biomes of Important Bird Areas. In *Ecological Informatics*, Volume 14, pp 38–43.

Yoo, S., 2013. A Wireless Sensor Network-Based Portable Vehicle Detector Evaluation System. In *Sensors* 13(1), pp. 1160-1182.

Yuea, S., Chena, M., Wena, Y., Lua, G., 2016. Service-oriented model-encapsulation strategy for sharing and integrating heterogeneous geo-analysis models in an open web environment. In *ISPRS Journal of Photogrammetry and Remote Sensing,* Volume 114, pp 258–273.

Yung Rowe, B. L., 2014. The Models as a Service era has arrived [online], Available from: https://cartesianfaith.com/2014/06/05/the-models-as-a-service-era-has-arrived/ [Accessed 14 Oct. 2017]

White, RM., 1987. A sensor classification scheme. In *IEEE Trans Ultrason Ferroelectr Freq Control*. 34(2), pp124-126.