WebShawn, Simulating Wireless Sensors Networks from the Web

Diego Alberto Godoy
Centro de Investigación en Tecnologías de la Información y Comunicaciones
Universidad Gastón Dachary
Posadas, Argentina
diegodoy@citic.ugd.edu.ar

Eduardo Omar Sosa
Secretaría de Investigación y Posgrado -SECIP.
Universidad Nacional de Misiones
Posadas, Argentina
eososa@unam.edu.ar

Rebeca P. Díaz Redondo
Departamento de Ingeniería Telemática. E. E. Telecomunicación
Universidad Vigo
Vigo, España
rebeca@det.uvigo.es

Hernán Bareiro
Centro de Investigación en Tecnologías de la Información y Comunicaciones
Universidad Gastón Dachary
Posadas, Argentina
hbareiro@citic.ugd.edu.ar

Abstract— **Although there are several tools available to simulate a Wireless sensor networks (WSN), few of them allow on-line simulation as WebShawn does. The tool can be considered as a complement of the existing experimental platforms allowing focus on implementation, collection, and analysis of data, in a relatively simple manner. The adopted software architecture is based on the remote simulation and visualization model, where the simulation engine is located at the server side, and the access to the simulation system is performed using a conventional web browser, obtaining the results through the Internet interface. This particular option is the main advantage of WebShawn regarding other existing applications. Being tested in the classroom as well as in a laboratory research, WebShawn is a tool ready for use by Information and Communications Technology (ICT) industry, researchers, students, or whoever needs to simulate WSN scenarios with a huge number of nodes. WebShawn allows users to assign their efforts to develop, design the network, and plan the scene, without losing time in the installation stage.**

Keywords- Shawn, WSN simulators, Web Simulation

I. INTRODUCTION

Wireless sensor networks are heterogeneous networks that comprise small, resource-constraint sensor nodes, gateways and back-end systems. In the future, sensor nodes will measure ambient parameters such as temperature, motion, and humidity; and the gateways will provide the integration with traditional networks while backend systems process and visualize received data. Application development for WSNs is not easy as it joins the challenges of distributed applications and embedded programming. Other elements like heterogeneity, scalability, and environmental conditions can worsen the scenarios, thus turning this into an unmanageable task. In spite of the long-term research and development of widely available simulation tools focused on simulation systems for WSNs, only a few of them have received the acceptance of the research community like NS-3 [1], OMNeT++ [2], Cooja [3], OpenWSN [4]. However, their installation and usage are still difficult while the learning curve is steep, thus reducing their chance of reaching a broader user base. Undoubtedly, one major issue is the variety of programming methodologies used for WSN applications. It is important to adapt business processes and the underlying software infrastructure quickly and flexibly, to be ready to react to minor changes in markets. To reach this goal, organizations, already in the early 1990s, used to focus on modeling, analysis and adapting business processes. With the birth of the Service-Oriented Architectures (SOA) [5] based on Internet standards, being followed by an ever-increasing number of companies.

To succeed in building up algorithms and protocols for WSNs, a thorough knowledge of this type of ad-hoc networks is necessary. Currently, there are three different approaches: analytical methods; real-world experiments or computer simulations [6].

At the beginning of the WSN communication protocols design stage, simulation is a fundamental tool to simulate the behavior of a number of nodes in a short period. Before deploying the WSN in the real world, the WSN software and firmware must be thoroughly tested since WSN nodes do not offer suitable debugging interfaces, and they are typically located in distant running scenarios after deployment.

The use of a WSN simulator can be practical in the evaluation of new applications and protocols on a large scale, thus allowing system developers to tune the WSN parameters carefully (at low and high levels) and dramatically reduce development time and cost. For this purpose, there are several tools available [7], [8], [9].

Although the characteristics of the analytical methods are generalization (the act or process of forming opinions that are based on a small amount of information) and articulation (the action or manner of jointing or interrelating), their approach oversimplify the WSN behavior reducing their scope only to general cases and covering mainly scalability issues, thus resulting in marginal use to develop WSN applications. Furthermore, they have difficulties in capturing, in a formal manner, the particular details of the selected scenario.

There are several free experimental platforms allowing focus on implementation, collection, and analysis of data in a relatively simple manner; but the simulation of WSNs is most frequently considered to show and demonstrate that a given network will provide the service asked for a particular environment. Besides, the environment influences at random, making it difficult to identify sources of errors as well as allowing reproduction of the obtained results. Additionally, real world deployment is a hard task, involving other management tasks not related to the problem. Hence, they are typically limited to a few dozens of devices, while future scenarios anticipate networks of several thousands to millions of nodes. Computer simulations are a mean to tackle the task of an algorithm and protocol engineering for WSNs. Some simulation tools are available for free on the web [1], [2], [3]. These tools consider a controlled and limited version of real-

978-1-5386-3839-2/17/$31.00 ©2017 IEEE

world conditions in a context where the simulation highlights the various affected properties, such as transmission and propagation, thus reducing the need for real-world deployments in early-stage designs and therefore, helping to increase their size. Although a substantial degree of details can be considered in simulations tasks, they overlap and hide other important issues, such as the inability to find and analyze a single parameter of interest.

Both general and specialized simulators are available to simulate WSNs. When a tool uses firmware and hardware to perform the simulation, it becomes an emulator and combines hardware and software behavior. The emulator is implemented in real nodes. Thus, it may provide more precision on performance. Usually, the emulator offers high scalability, which can include a substantial number of nodes at the same time. Setting up a configuration of a particular purpose simulation system consumes significant time resources. Either because it defines a specific requirement of the operating system used, or because it is necessary to acquire the hardware and to configure it correctly. It is widely known and accepted that getting results as close as possible to real systems requires well-defined systems. So the knowledge about what it is intended to be simulated is essential, as well as how to define parameters, and finally to get and display the outcome of the simulation. All these configuration tasks result in a critical time investment that frequently users are not ready to spend. However, validating conclusions out of simulation analysis is a not a minor duty. There are two key aspects in WSNs simulators: a) The correctness of the simulation models and b) the suitability of a particular tool to implement the model. A "correct" model based on reliable assumptions is mandatory to obtain trustful results. As a rule, the tradeoff should be precision and detail level versus performance and scalability.

The main feature of web applications is their portability, together with the easy way to access them seamlessly. A Client-Server model and a browser user interface within a web browser can provide controls via a dedicated graphical user interface (GUI), remaining the internal complexity of the isolated system at the server-side, being completely transparent for most users.

II. Related Works

Nowadays, there is a significant number of open-source, flexible and extensible WSN simulators, which can be classified into different categories according to their features and main applications. The test interface of these tools and its interaction with the WSN control processes is considerably weak.

Early in 2000, the WWW was considered as an environment for hosting modeling and simulation tools like SIMAN [10] as a web-based interface or toolkit developed for storing and executing SIMAN simulation models over the Internet [11]. Some Performance comparisons among well-known WSN simulators like NS-2, NS-3, TOSSIM, GloMoSim, Qualnet, EmStar, OMNeT++, J-Sim, ATEMU, and Avrora can be observed in the works of Imran [7], Das [12] and Chandrasekaran [13]. Others like Kim [14] considered to develop a parallel processing simulator enabling the user to mitigate the slowdown of execution speed using a Graphical Processor Unit (GPU). As usual, the GPU mode is multicore. Xian [15] and Fortino [16] propose a simulation-ased approach considering WSN applications as a part of the Building Management Framework (BMF).

Finally, Shawn [17] itself is another example of WSN simulator, but there is not exists web interface for it. Details about Shawn will be given in section 3. Nevertheless, none of them offers the chance of remote access from the web for designing and running the simulation.

III. Shawn Arquitecture

Shawn is a simulation framework whose central idea is to replace the physical layer of a network, with abstract and interchangeable models that can be used in the simulation of large networks within a reasonable time [17]. It takes less than a minute comparing to the 25 hours of Ns-2's running, it is written in C++ for maximum performance [18], and it is distributed under BSD license.

Conceptually, it consists of three main components, namely, the simulation environment, the sequencer, and the models. The simulation environment contains simulated elements and their properties while the sequencer influences the behavior patterns of the environment [19]. Fig.1 shows the high-level architecture of Shawn and an overview of its core components.

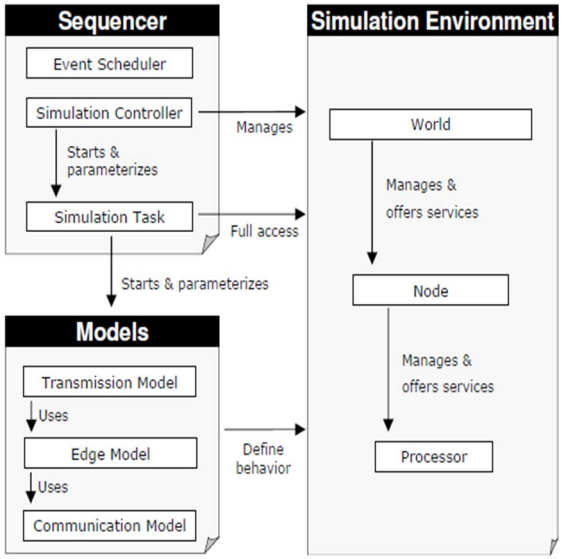

Figure 1. High-level architecture of Shawn [20]

While other simulators focus themselves on the simulation of a particular phenomenon, Shawn simulates the effect caused by the phenomenon itself. The main difference of Shawn to other WSN simulators is its design and its speed.

For example, instead of simulating a whole MAC layer, including radio propagation model; Shawn models only the possible effects that can happen on the MAC layer (packet

loss, corruption, etc.), having several implications for the outcome. Thus, the results are more predictable, and there is a significant performance improvement on the simulation. However, the tool is unable to reach deeper levels of detail and shows limitations at the physical layer.

Shawn has been developed as a distinct and powerful tool to simulate WSNs [17], being able to scale to several thousand of nodes. The main difficulties observed while deploying the tool are its complex processes for installation, configuration, and the setup of the several simulator's modules [21]. Because of that, and considering the hardware and operating system used, different problems must be solved until the installed tool is ready to work. Moreover, the fact that Shawn is a command line simulator, there is an additional issue that users should deal with, which is the simulation environment. When obtained from repositories, the user must use the simulator in an individual manner and install it on his workstation. These are the main reasons for adapting the tool to the web environment.

The primary goal of the present work has been the design of a WSN web-based simulator for Shawn [22], which has been used as a remote engine for the simulation and visualization.

We determined the requirements and information needed to customize and adapt Shawn as a simulation engine, thus allowing receiving parameters and values from the web. The tool should execute the simulation and send back the results to the internet application. For this purpose, we developed several software modules to link the front-end (client side) and the back-end (server side) to the simulation engine in order to consider the different scenarios.

IV. WEBSHAWN

The adopted software architecture is based on the remote simulation and visualization model as defined by Whitman [20], and Byrnea [21]. The simulation engine is located at server side; while the access to the simulation system is performed using a conventional web browser. A web browser, as a lightweight interface, has been chosen in order to input the parameters to the web server, where the values are captured. It is at that point that the interaction with Shawn properly starts.

Once time simulation ends, the outcome of the process is sent back to the user through the web interface. The data input provides functionality to nodes, providing the parameters that control the execution of the simulation as well as the code of the applications that would run on each node. Once uploaded the written code to the nodes, the script compilation and library linking start. After the expected results are achieved (no errors found), the application starts running on the simulator.

The user can choose different models in order to run several scenarios. Fig. 2 shows the schema of the general architecture solution of the Shawn web-simulator.

The GUI runs in the client side browser using HTML5, PHP, Java Script and CSS. To save information about the user´s simulation projects, application data, simulation models and data structures, PostGreSQL is used as a database management system, while RedBeanPHP is being used as Object Relational Mapper [22].

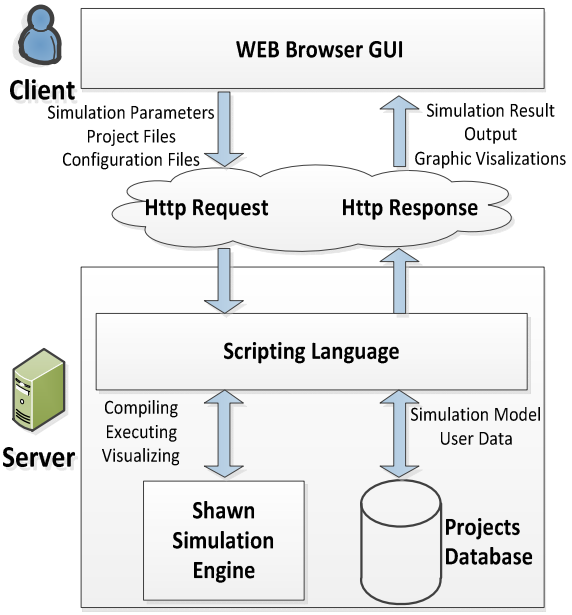

Figure 2. General Architecture for Shawn web-simulator.

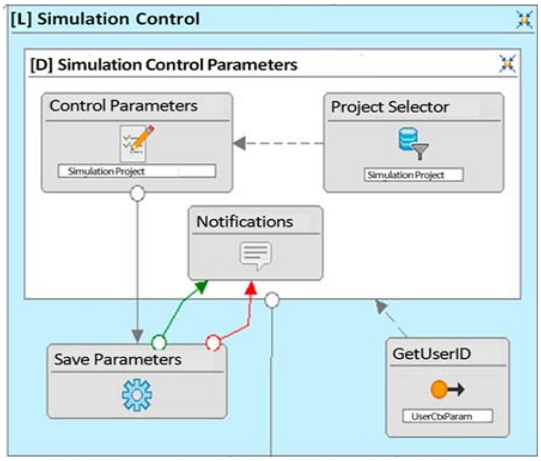

Figure 3. Hypertext model of the simulation control page.

The Web Modeling Language (WebML) [23] provides graphical, yet formal conditions, typified in a whole design process, which can be aided by visual design tools. WebML is a high-level language for modeling, design, and implementation of Web applications that make heavy use of data [24]. WebML combines developer´s well known traditional modeling languages with the data conceptual design using the Unified Modeling Language (UML). UML provides hypertext-modeling primitives based on the Entity-Relationship Model by using a straightforward and expressive specification, supported as well by an intuitive graphical notation [25].

Besides data modeling, which is a well-established activity, hypertext modeling is a new discipline, it still lacks basic well-established concepts, notes, and design methods.

The most important hypertext models of Web Shawn are the simulation control page, shown in Fig. 3, as well as the simulation output results page is presented in Fig. 4.

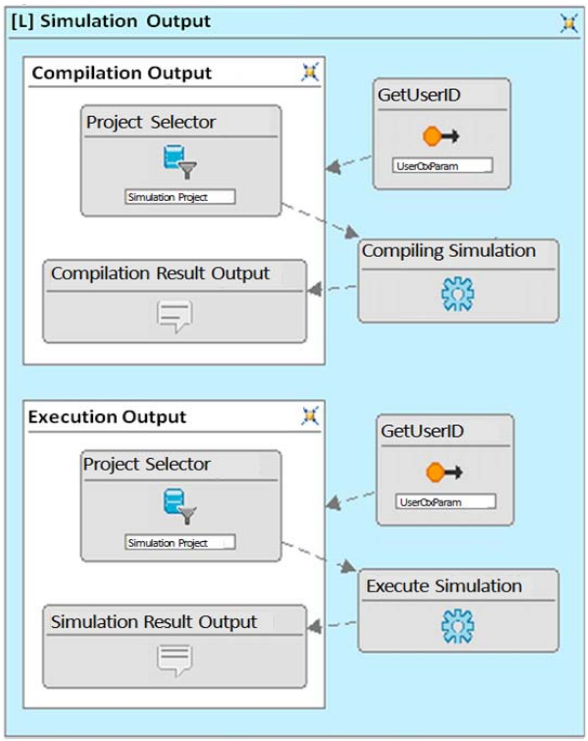

Figure 4. Hypertext model of the simulation output page

V. EXAMPLES AND EXPERIMENTS

The "Project_test" simulation project shown in Fig. 5, considers a very simple application. Each WSN node must start after a particular broadcast message network reaches it. The receiver node prints its identifier within the network, as well as the sender ID, to test the network connectivity. After a given number of iterations, the system identifies the known nodes and their neighbors and labels them. If no message is received the link is considered down and the node cannot be reached under the current simulation conditions. For simplicity, only six parameters have been considered as input. They are the number of nodes (count), the range of the sensor (range), the dimensions of a rectangular area (width and height), the simulation seed, and the number of iterations (max iterations), leaving all the other parameters by default.

In another scenario, a user model has been implemented considering a distributed routing algorithm. It generates a routing tree from every sensor node to the sink node according to the following tasks:

- The sink/gateway node sends a broadcast message (flood) to the entire network notifying that it is the given gateway node.
- Every node receiving a message stores the number of nodes, which are its predecessors and the known path and the number of hops along the route to the sink; and then sends that information to their neighbors. All neighbors perform the same action.
- At random, each node sends a state message to the gateway node informing the numbers of hops from itself to the sink. The message is routing to the gateway through their predecessors.

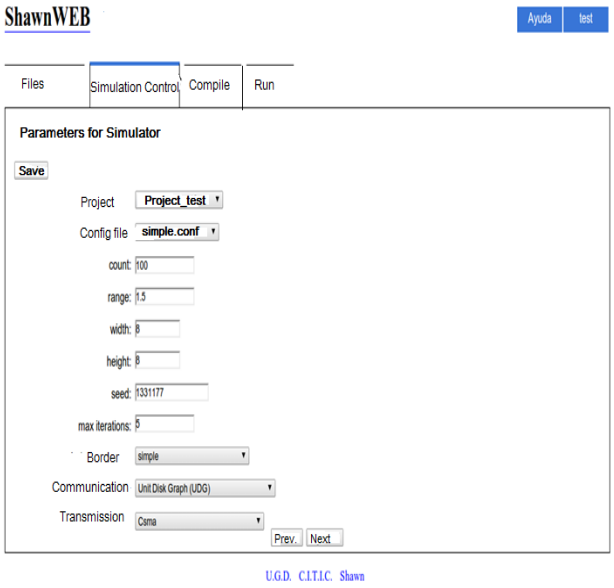

Figure 5. Input parameters interface

After entering the simulation parameters and building up the application, a graphical result is obtained (Fig. 6), where in this case a full-connected network can be observed. In Fig.7 a fragmented network can be observed, depending on parameters entered.

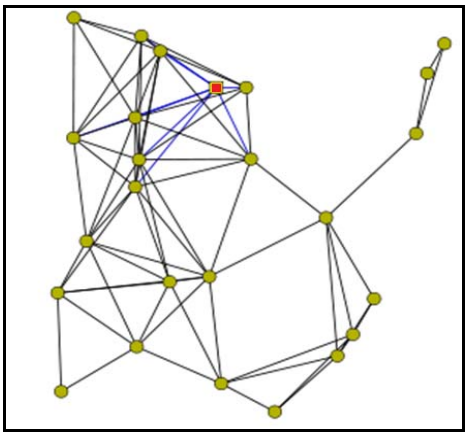

Figure 6. graphical visualization

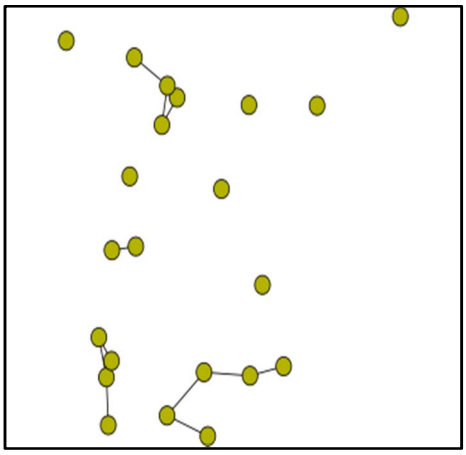

Figure 7. Graphical visualization: fragmented network

Circles represent each node while squares show gateways. The nodes (circles) are interconnected to the gateway (squares). They communicate with the gateway node either through their predecessors or directly.

Finally, Fig. 8 shows the text output, as an HTML built by the simulator.

Simulation Output

enrut_sim12 | parameters1.conf | Ejecutar | Descargar | Visualizar

```
  [ 25 active, 0 sleeping, 0 inactive ]

------------------------------ BEGIN ITERATION 8
processors.enrut_siml2: Number of nodes known to gate:  14
processors.enrut_siml2:   Min hops:  1
processors.enrut_siml2:   Max hops:  2
processors.enrut_siml2:   Avg hops:  1.42857
------------------------------ DONE ITERATION 8
  [ 25 active, 0 sleeping, 0 inactive ]

------------------------------ BEGIN ITERATION 9
processors.enrut_siml2: Number of nodes known to gate:  21
processors.enrut_siml2:   Min hops:  1
processors.enrut_siml2:   Max hops:  3
processors.enrut_siml2:   Avg hops:  1.95238
------------------------------ DONE ITERATION 9
  [ 25 active, 0 sleeping, 0 inactive ]

------------------------------ BEGIN ITERATION 10
processors.enrut_siml2: Number of nodes known to gate:  24
processors.enrut_siml2:   Min hops:  1
processors.enrut_siml2:   Max hops:  4
processors.enrut_siml2:   Avg hops:  2.20833
```

Figure 8. Output text from simulator

VI. Conclusions and Future Works

Due to their ability in checking the algorithms and protocols and for the fast prototyping and tackling of large-scale systems analysis simulation tools are broadly used. Many simulators show better performance when running on a computer, but it becomes unsuitable in real environments regarding completeness, complexity, correctness and realism. In this paper, the WebShawn prototype is presented. It is a tool for porting Shawn to the Internet. WebShawn is available for industry, researchers, students, or whoever needs to simulate WSN scenarios consisting of a huge number of nodes. WebShawn allows the user to dedicate most efforts in developing, designing the network and planning the scene without losing time in the installation stage but being able to customize and learn about how the simulator works. The prototype has been tested in the classroom as well as in a research laboratory.

Future work will examine the ability of WebShawn serving multiple clients, to determine concurrent service capacity. In the same line, the tool could be implemented as a piece of a WSN middleware developed by the same authors [26] of this paper, to use the acquired data by the WSN itself as well as data generated by the simulator.

References

[1] NS-3 Consortium, [En línea]. Available: https://www.nsnam.org/.

[2] OMNeT++ Discrete Event Simulator, [En línea]. Available: https://omnetpp.org/.

[3] Contiki , «Get Started with Contiki,» [En línea]. Available: http://bit.ly/1VLH9HK.

[4] OpenMote, «Open Hardware for the Internet of Things,» [En línea]. Available: http://bit.ly/1SS5hmi.

[5] T. Erl, Service-Oriented Architecture (SOA): Concepts, Technology, and Design, Prentice Hall PTR, 2005.

[6] D. Dujovne, «Doctoral Thesis Enhancing Experimentation in Wireless Networks. Networking and Internet Architecture,» Université de Nice Sophia Antipolis, Nice, France, 2009.

[7] M. Imran, A. Said and H. Hasbullah, "A survey of simulators, emulators and testbeds for wireless sensor networks," in *International Symposium in Information Technology (ITSim)*, Kuala Lumpur, 2010.

[8] H. Sundani, H. Li, V. Devabhaktuni, M. Alam and P. Bhattacharya, "Wireless Sensor Network Simulators. A Survey and Comparisons," *International Journal Of Computer Networks (IJCN),* vol. 2, no. 5, pp. 249-265, 2011.

[9] P. Chhimwal, D. Singh Rai and D. Rawat, "Comparison between Different Wireless Sensor Simulation Tools," *IOSR Journal of Electronics and Communicatio,* vol. 5, no. 2, pp. 54-60, 2013.

[10] C. D. Pegden, ""Introduction to SIMAN"," in *roceedings of the 15th conference on Winter simulation-Volume 1.*, 1983.

[11] A. Guru, P. Savory and R. Williams, "Web-based simulation management: a web-based interface for storing and executing simulation models," in *32 conference on Winter simulation. Society for Computer Simulation International*, 2000.

[12] T. Das and S. Roy, "Energy efficient and event driven mobility model in mobile WSN," in *2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, Kolkata, 2015.

[13] V. Chandrasekaran and S. a. S. Anitha, "A Research Survey on Experimental Tools for Simulating Wireless Sensor Networks," *International Journal of Computer Applications,* vol. 79, no. 16, pp. 1-9, 2013.

[14] H. Kim, H. Song, J. Park and Y. Jeong, "Parallel Processing Simulator for Separate Sensor of WSN Simulator with GPU," in *IEEE 29th International Conference on Advanced Information Networking and Applications (AINA)*, Gwangiu, 2015.

[15] X. Xian, W. Shi y H. Huang, «Comparison of OMNET++ and other simulator for WSN simulation,» de *3 IEEE Conference on Industrial Electronics and Applications*, Singapur, 2008.

[16] G. Fortino, R. Greco and A. Guerrieri, "Modeling and evaluation of the building management framework based on the Castalia WSN simulator," in *17 International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, Whistler, 2013.

[17] S. Fekete, A. Kroeller, S. Fischer and D. Pfisterer, "Shawn: The fast, highly customizable sensor network simulator," in *Fourth International Conference on Networked Sensing Systems*, Braunschweig, 2007.

[18] D. Pfisterer y S. Fischer, 2015. [En línea]. Available: https://github.com/itm/shawn/wiki/Performance.

[19] A. Kroeller, D. Pfisterer, C. Buschmann and S. Fekete, "Shawn: A new approach to simulating wireless sensor networks," *Proceedings of 3rd. Symposium on Design, Analisis and simulation of Distributed Systems (DASD 2005),* pp. 117-124, 2005.

[20] L. Whitman, B. Huff and S. Palaniswamy, "Commercial simulation over the web," in *Proceedings of the 30th conference on Winter simulation (WSC '98)*, Los Alamitos, CA, USA, 1998.

[21] J. Byrnea, C. Heaveya and P. Byrne, "A review of Web-based simulation and supporting tools," *Simulation Modelling Practice and Theory,* vol. 18, no. 3, pp. 253-276, 2010.

[22] RedBeanPHP, "RedBeanPHP Easy ORM for PHP," [Online]. Available: http://redbeanphp.com/. [Accessed 12 October 2015].

[23] WebML.org, "The Web Modeling Language," [Online]. Available: http://www.webml.org/. [Accessed 2016 Marzo 15].

[24] S. Ceri, P. Fraternali and A. Bongio, "Web Modeling Language (WebML): a modeling language for designing Web sites," *Computer Networks,* vol. 33, no. 1-6, pp. 137-157, 2000.

[25] N. Moreno, P. Fraternalli and A. Vallecillo, "A UML 2.0 profile for WebML modeling," in *Workshop proceedings of the sixth international conference on Web engineering (ICWE '06)*, New York, 2006.

[26] D. Godoy, R. Díaz Redondo y E. Sosa, «Internet De Las Cosas. Middleware De Gestión De Datos De WSN,» de *Conferencias Iberoméricanas WWW/Internet*, Florianapolis, 2015.