A Model-Driven Engineering Framework for Architecting and Analysing Wireless Sensor Networks

Krishna Doddapaneni*, Enver Ever*, Orhan Gemikonakli*, Ivano Malavolta†, Leonardo Mostarda*, Henry Muccini†
*Computer Communications Department, School of Engineering & Information Sciences, Middlesex University, UK
Email:{k.doddapaneni, e.ever, o.gemikonakli, l.mostarda}@mdx.ac.uk
†Dipartimento Di Informatica De L'Aquila Via Vetoio L'Aquila 67100, Italy
Email:{ivano.malavolta, henry.muccini}@univaq.it

***Abstract*—A Wireless Sensor Network (WSN) is composed of distributed sensors with limited processing capabilities and energy restrictions. These unique attributes pose new challenges amongst which prolonging the WSN lifetime is one of the most important. Challenges are often tackled by a code-and-fix process that relies on low-level hardware and software information.**

Recently, the need of abstracting an implementation view into an architectural design is getting more realized. A clear separation of concerns is needed since hardware and software aspects are locked and tied down to specific types of nodes, hampering the possibility of reuse across projects and organizations. This means that exploiting the right level of abstraction, and keeping explicit (and separated) software and hardware architectural details will surely ease developers' job.

In this paper we propose a modeling framework that allows developers to model separately the software architecture of the WSN, the low-level hardware specification of the WSN nodes and the physical environment where nodes are deployed in. The framework can use these models to generate executable code for analysis purposes. In this paper we focus on energy consumption analysis.

I. Introduction

A Wireless Sensor Network (WSN) consists of spatially distributed autonomous sensors that monitor environmental conditions in order to accomplish a task such as fire monitoring and home temperature control [12]. WSNs lead to many challenges [16] such as abstraction, separation of concerns and reuse. When current practices on WSNs are considered, it is quite evident the lack of engineering methods and techniques to manage these challenges.

Beside the need of programming abstraction it is well-acceted the need of **abstracting** an implementation view into an architectural design. As remarked in [14], *"end users require high-level abstractions that simplify the configuration of the WSN at large, possibly allowing one to define its software architecture based on pre-canned components"*. **Separation of concern** is limited since hardware and software components are locked and tied down to specific types of nodes, hampering the possibility to **reuse** components across projects and organizations. Moreover, while the focus is mostly on software components and hardware, there is still a missing piece from the WSN modeling puzzle: the physical environment where the WSN application will be deployed. Since the physical environment plays a fundamental role especially when the energy consumption of WSNs is considered, lack of an explicit representation of the physical environment is an important limitation of existing approaches (see Section V). Under this perspective, approaches abstracting implementation details from the underlying hardware and physical infrastructure are strongly advised [12], [1]. Some initial effort has been conducted for architecting WSNs [9], [6], however, they partially meet the expectations.

This paper proposes *a model-driven engineering (MDE) framework to support an architecture-driven development and analysis* of WSNs. The framework makes use of a multi-view architectural approach [7] to model separately (i) software components and their interactions, (ii) the low-level and hardware specification of the nodes, and (iii) the physical environment where the nodes are deployed. Although using different models helps for the separation of concerns, it introduces the challenge of linking the models together to get a complete view of the system under development. For this purpose, we use suitable weaving models to map software components into different hardware nodes and virtually deploy the nodes into specific areas of the physical environment. Model-to-text transformations can use these models to generate executable code, e.g., for analysis purposes. This paper shows how energy consumption analysis can be conducted starting from the model-driven framework we propose. Different model-based analysis techniques can also be plugged into the proposed modeling framework. The contribution of this paper is a model-driven engineering framework that:

- enables **reuse** by clearly separating software, hardware, and environment descriptions of a WSN;
- improves **abstraction** by masking the complexity of low-level hardware details (code is also automatically generated from the models);
- facilitates **model-based analysis** by automatically generating analysis models out of the modeling framework.

The paper is organised as follows: Section II provides an overview of the framework; Section III briefly discusses the model-driven engineering framework for architecting WSNs;

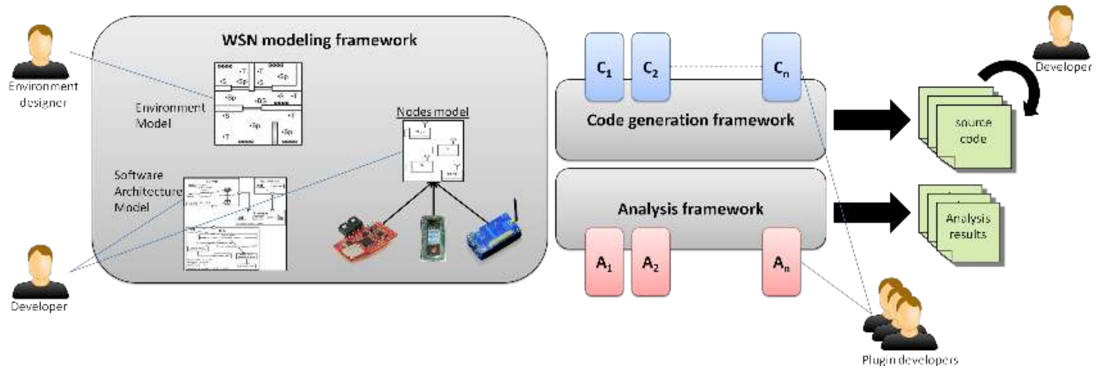

Figure 1. Overview of the modeling framework

Section IV presents how energy analysis can be performed; Section V provides the related wok; Section VI summarises the paper.

II. Overview of the modeling framework

The main goal of our research is to take advantage of Model Driven Engineering (MDE) techniques to support an architecture-driven development and analysis framework for improving WSNs. Figure 1 shows an overview of the framework we are working on. It is composed of three main parts.

The WSN Modeling Languages. The main building blocks of such a framework are three modeling languages: the *Software Architecture Modeling Language for WSN* (*SAML*), the *Node Modeling Language (NODEML)*, and the *Environment Modeling Language (ENVML)*. Depending on the analysis to be performed, those modeling languages can be linked together through weaving models, in order to create a combined software, nodes, and environmental view of the WSN.

The Code generation framework. This framework will manage a repository of code generation engines. Each engine will be realized as a plugin of this framework. A generic engine knows at run-time which code generation plugins are installed into the framework, and automatically provides to the developer the available target implementation languages.

The Analysis framework. This framework is similar to the code generation one, but it manages analyses for WSNs (e.g., coverage, connectivity, energy consumption analysis).

Our framework is generic since it is independent from the programming language, hardware and network topology. Starting from a set of models (each one reflecting a certain WSN viewpoint), the code generation and analysis components can be plugged into the framework for generating executable code or analysis outcomes. Even if the proposed modeling languages are very abstract, the current framework is very much focused on energy consumption analysis, and thus it might be necessary to extend the modeling languages to provide the needed concepts for supporting other analysis or code generation engines. In this context, introducing changes at the metamodel level has a strong impact in the already developed plug-ins (model editors, model transformations, etc.); this problem is called metamodel co-evolution management and it is well-known in the MDE research field [**?**], [**?**]. If we look at this problem from a different perspective, similarly to what we proposed in a previous work on architectural languages interoperability [**?**], a possible solution could be to provide a systematically defined extension process for our modeling languages, in which their extensions are organized into a hierarchy obtained by systematically extending a root modeling language.

This paper briefly describes the modeling framework and focusses on how to use those models to simulate WSN energy consumption. In order to demonstrate the validity of our approach, we make use of a *home automation* case study [5], [3] by focussing on the *fire alarm* and automatic heating systems. The fire alarm system considered is composed of two types of sensors that are temperature and smoke sensors. There are also sprinkler actuators that are used to enable the water flow in case of fire. Temperature sensors monitor the temperature at regular intervals (every 30 seconds). When a temperature sensor reads a value that exceeds a specified threshold, it sends an alert message to the smoke sensors. Each smoke sensor receives the alert and checks for smoke. When a smoke sensor detects smoke it sends an alarm message to the sprinklers to activate the water flow. The automatic heating system is composed of the same temperature sensors of the fire alarm system, a base station and various heaters. Temperature sensors send readings every 30 seconds to a base station. This is placed at the centre forming a star topology. The base station averages the readings and decides whether or not the central heating system should be on.

III. Modeling Languages

In our approach model-driven techniques are used to model the software and hardware architectures of WSN nodes. Our approach also defines a physical environment

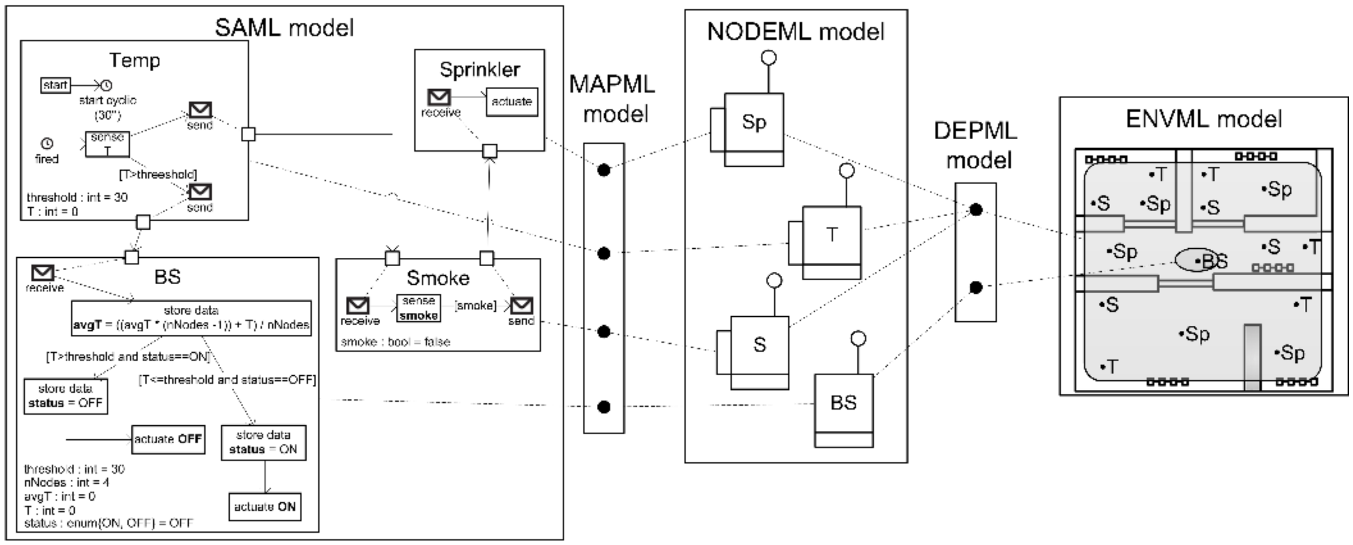

Figure 2. Modeling the *home automation* case study

model that enables virtual deployment of nodes. The main building blocks are three modeling languages (see Figure 2): SAML, NODEML, and ENVML. For the sake of space, this section describes the main concepts included in each language, a more detailed description is provided in Appendix and their implementation can be found in [11]. It is important to note that we defined the static semantics of the languages by means of their underlying metamodels. Also, each metamodel is complemented with a set of OCL[1] constraints that ensure properties that are not purely structural; however, the description of those constraints is omitted for the sake of brevity.

Software Architecture Modeling Language for WSN (*SAML*). It allows designers to define the software architecture of the WSN application. The software architecture of a WSN is defined as a set of interacting *components* which exchange messages by passing through *ports*. Each component can declare a set of *application data* that can be seen as local variables declared in the scope of the component; application data is manipulated by actions defined in the behaviour of the component. In SAML, components can contain a behavioural description. The behaviour of each component is represented by a list of *events*, *conditions* and *actions*, which together describe the control flow within the component from an abstract point of view. Examples of actions comprise: start or stop of a timer, sending a message via a specific message port, get data from a sensor, etc. Examples of events comprise: the message reception on a message port, a timer fired, etc.

By referring to the leftmost part of Figure 2, the SAML model of the *home automation* case study is composed of four main components: *Temp*, *Smoke*, *Sprinkler* and Base Station *(BS)*, each of them describing the logic of the software running on each type of node of the case study. The left-lower part of each component contains its *application data*; those data can be set either statically (e.g., *threshold* in *Temp*), or by some sensing action performed by the component (e.g., *T* in *Temp*). Application data can be used within conditions (e.g., see the condition on the link between *sense smoke* and *send message* actions in *Smoke*), or for passing values while triggering an actuator (e.g., see the *actuate OFF* action in *BS*). *Store data* actions are used to assign specific values or expressions to an application data (e.g., *store data status = ON* in *BS*). Messages can be sent to a given message port by using *send message* actions (e.g., *send* actions in *Temp*), and the receiving of a message to a specific port is represented by a *receive message* event (e.g., see the *receive* event in *Sprinkler*).

Node Modeling Language (*NODEML*). NODEML is our language for describing the low-level details of each *type of node* that can be used within a WSN. Indeed, different WSN applications can reuse the same NODEML models and organize them differently, depending on the requirements of the application. A NODEML model contains exclusively low-level, node-specific information, like its supported *operating system*, implemented *MAC protocols*, *routing protocols*, and so on. It also contains the hardware specification of the nodes, including their *energy sources* (e.g., batteries), *communication devices*, installed *sensors* and *actuators*.

The central part in Figure 2 shows the four types of models we decided to use in the *home automation* case study. More specifically, we decided to use a node for each type of sensor, rather than mounting multiple sensors on each node. This decision leads to a simpler implementation of each node, and to a more flexible network topology. This is a

[1]http://www.omg.org/spec/OCL/2.0/

typical architectural trade-off since the previously described benefits come at the cost of higher communication overhead between the nodes. The BS node type is not equipped with any sensor since it has to accomplish only tasks related to communication within the network. For each node we defined its low-level parameters; for example we decided to use the CC2420 radio defined by the Texas instruments, the output power of the different transmission levels varies between 0dBm and -25dBm, the radio bandwidth is 20 MHz, and T-MAC is used as a MAC protocol.

Environment Modeling Language (*ENVML*). It allows designers to specify the physical environment in which the WSN nodes are deployed in. An ENVML environment identifies a specific area in the 2D space in which *obstacles* can be freely positioned. Each obstacle has an *attenuation* coefficient that may represent a specific material like concrete, wood, glass, etc.

The rightmost part of Figure 2 graphically shows the physical environment in which we will virtually deploy the node types defined in NODEML. In this case we consider a flat composed of five rooms. We also consider different obstacles such as wooden doors (the thin obstacles in Figure 2), concrete walls (the large obstacles in Figure 2) and a glass partition (the darker obstacle in Figure 2). Behind the lines, each obstacle is represented as the set of coordinates of its perimeter in the 2D space.

In our framework we provide two auxiliary modeling languages for *linking together* the previously described languages, namely: MAPML and DEPML. Those models are technically called weaving models, and have been successfully used in many fields (such as software architecture [10], software product lines [2], etc.) to create semantic links among different models. This approach provides a clear separation between software components, WSN nodes, and the physical environments, thus promoting the reuse of models across different systems and projects.

Mapping Modeling Language (*MAPML*). A MAPML model is composed by a set of mapping link, each of them weaving together a node definition from the NODEML model and a component from the SAML model. The component in the SAML model will be physically deployed on the linked node in the NODEML model. In the scope of a given mapping link, a designer can define three types of link: (i) sensor mapping link: it allows designers to associate which sensor is actually used in the context of the sense data action, (ii) actuator mapping link: it is similar to a sensor mapping link, but it refers to actuators, rather than to sensors, (iii) communication device mapping link: it specifies what the communication device is (e.g., a radio antenna installed on the node) that corresponds to a specific port in the *SAML* model of the WSN.

When our *home automation* case study is considered, the MAPML model in Figure 2 defines how we mapped each component defined in the SAML model into its corresponding node type in the NODEML model. For the sake of clarity we do not show the other mappings defined in the MAPML model (e.g., the sensor, actuator and communication device mappings).

Each sense and actuate action is mapped to a unique sensor and actuator node type. Each send and receive message actions are mapped to its corresponding radio communication device.

Deployment Modeling Language (*DEPML*). It allows designers to consider each node type defined in the *NODEML* model and to *instantiate* it in a specific area within the physical environment defined in an *ENVML* model. Each node type can be instantiated ”n” times within a specific area. Currently, within a certain area each node type can be distributed in three different ways: (i) randomly within the area, (ii) as a grid with a certain number of rows and columns, and (iii) custom, i.e., each node instance can be manually placed within the area.

By considering our *home automation* case study, the DEPML model in Figure 2 shows how we virtually deployed the node types defined in NODEML into the physical environment defined in ENVML. More specifically, we defined two deployment areas: the first one contains the BS node only and it is placed in the center of the environment, the second area spans throughout the whole flat and contains all the other nodes in the WSN; those nodes are customly distributed by manually setting their position within the area, allowing us to place each type of node in each room of the flat.

IV. WSN Simulation

This section describes the application of our approach for estimating the WSN life time. We describe the generation of simulation scripts, the definition of the path loss formula employed and we discuss the results obtained.

The modeling languages described in the previous section can be translated into scripts that allow the simulation of the WSN. This translation has been implemented by using Acceleo[2] that is the Eclipse implementation of the Object Management Group (OMG) MOF Model-to-Text Language (MTL) standard[3]. Acceleo provides, among the other, the following features: customizable code templates, generation engine, debugger, etc. Fundamentally, our Acceleo application contains a set of templates that specify how the various model elements described in Section III are converted into text patterns. For the sake of brevity we do not go into the details on the generation of simulation scripts, we tested our Acceleo application by compiling and running a number of generated simulation scripts in Castalia, demonstrating satisfactory results.

[2]http://www.eclipse.org/acceleo

[3]http://www.omg.org/spec/MOFM2T/1.0/

The Castalia[4] simulator for WSNs has been chosen as target for the generation of simulation scripts. Castalia can simulate protocols and/or algorithms by using realistic wireless channel and radio models. It can simulate a wide range of hardware platforms. Energy consumption for each transmission level varies. For instance for 0 dBm, the power consumed for listening (receiving) is 62 mW and for transmission is 57.42 mW. Packet rate is kept at 250 kbps, the radio bandwidth is 20 MHz and the simulation runs for 9000 sec. Since T-MAC is used as the MAC protocol, the length of each frame period for all nodes is 610 milliseconds, and the duration of listen time out is 61 milliseconds. Although Castalia provides a good low level simulation platform; it does not provide any means to specify the application behaviour, the environment model and the path loss mathematical model. The application behaviour is needed to derive application level simulation parameters. Furthermore the environment model and the path loss mathematical model allow the calculation of the path loss values. Castalia assumes that the user provides path loss values, however it is necessary to derive those values from high level models such as the environment and path loss mathematical model. In this paper the details of the scenario considered is provided in SAML, NODEML and ENVML.

A *path loss model* can be specified in the ENVML model which is in turn used together with the physical environmental model in order to define the path loss between two nodes. Propagation path-loss models are important for the design of wireless networks to specify key parameters such as transmission power, frequency, antenna heights, and so on. The importance of these parameters is even more evident in case of WSNs, since they directly affect the residual energy. The propagation and path loss models are usually based on empirical studies on systems considered. For example the Okumura/Hata model has been used extensively both in Europe and North America for predicting the behaviour of cellular transmissions in built up areas. Indoor path-loss models are commonly used especially for picocells which cover a part of a building and span from 30 to 100 meters. They are used for wireless local area networks and picocell base stations, and wireless PBX systems. The earlier work for statistical measurement of signal amplitude fluctuations are dependent on empirical measurements for indoor office environment as well. Many of the researchers in the field, performs narrowband measurements within buildings mainly in order to determine the distance power relationship and to arrive at empirical path loss models for a variety of environments. In our case study we consider the dependant path loss model [8] that is widely used for indoor environment, however, in the future, it is desirable to perform measurements for a path loss model more suitable for the environment considered and WSNs:

[4]http://castalia.npc.nicta.com.au/

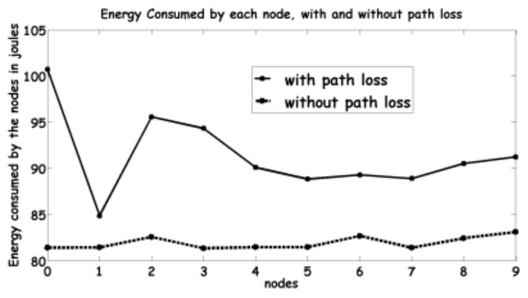

Figure 3. Energy consumed by each node with and without path loss

$$L_P = L_0 + 20log + \sum m_{type} w_{type}$$

where, L_P represents the path loss between two points, L_0 is the path loss in free space environment, m_{type} refers to the number of objects of the same type and w_{type} is the attenuation value attributed to that particular object. In the Table I we show some attenuation values in dB introduced by various materials.

Table I
PARTITION DEPENDENT LOSSES FOR 2.4 GHZ

obstacles	attenuation in dB
Concrete wall	12
Wooden door	2.8
Glass wall	2
Cinder wall	4
window	2
Brick	5
Masonry brick	17
metal door	12.4

A. Numerical results and discussions

In this Section we discuss the application of our approach to our *home automation* case study. In this case study we use the CC2420 radio defined by the Texas instruments, and the output power of the different transmission levels in dBm is varied from 0 to -25dBm. Energy consumption for each transmission level varies; for instance for 0 dBm power consumed for listening (receiving) is 62 mW and for transmission is 57.42 mW. Packet rate is kept at 250 kbps, the radio bandwidth is 20 MHz and the simulation runs for 9000 sec. T-MAC is used as a MAC protocol. These details are specified in NODEML.

The path-loss due to the material has been calculated by considering the attenuation values of Table I (see [8] for a complete list of all materials). Information presented in Table I is used in ENVML together with the location of servers. For the sake of the representation of information in ENVML, we can use numbers to represent sensors. Node 0 represents the base station. Nodes 1, 4, 5, 7, 9 monitor the temperature in the environment (i.e., they correspond to nodes *T* in Figure 2), and nodes 2, 3, 6, 8 monitor

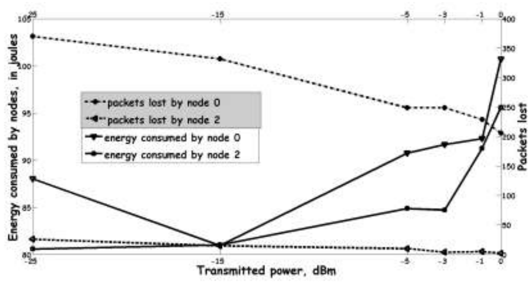

Figure 4. Energy consumed vs. transmitted power vs. packets lost

the presence of smoke ((i.e., they correspond to nodes *S* in Figure 2).

Figure 3 shows the energy consumption of each node in a free space environment and when the path loss is introduced. It is evident that ignoring the effect of path loss would be an optimistic assumption when energy consumed by each node is considered. For instance node 3 consumes 13 joules of more energy due to path loss, when compared to no path loss. This is consequence of an increasing number of retransmitted packets.

The trade-off between traditional performance measures such as packet loss and residual energy is presented in Figure 4. The dotted lines represent the packets lost and the straight lines represent the energy consumed by each node. As the transmission is decreased from 0 dBm to -25 dBm, there is a gradual increase in amount of packets lost. For node 0, as the transmission power is decreased from 0 dBm to -25 dBm, the number of packets lost increases to 370, from 206 and the energy consumed increases to 100 joules from 88 joules. Because of the retransmissions, more energy is consumed by the nodes. But the increase in transmission power does not necessarily mean increase in the life time as there are no retransmissions. Results presented in Figure 4 are particularly important to show the usefulness of a detailed, an realistic modelling tool. Analysing the tradeoff between the energy consumption and the packet loss specifies the operative area for applications which requires reliable transmission.

When the tradeoff between the packet loss and the energy consumed is analysed, it can be seen that the optimum transmission power should be between -15 to -5 dBm where the energy consumption is less than 95 joules and packet loss is less than 200 packets.

Please note that different levels of abstraction for software architecture, node model, environment model, mapping model, and deployment model (described by SAML, NODEML, ENVML, MAPML, DEPML respectively) introduces flexibility and efficiency for the researchers working in power management localisation routing, deployment technique, protocol development etc. of WSNs. For example, it is possible to change the environment considered, by modifying the ENVML, and the remaining definitions would not be affected. Similarly in case the specification of motes are altered, the user is able to modify NODEML, and run same experiments with various node specifications. In that sense the new architecture is very useful for optimisation of WSN architectures from various aspects.

V. Related Work

Currently, modelling is used to specify a WSN at different levels of abstraction (hardware, application, communication protocols, etc.) with the recurrent goals of code generation, communication overhead analysis, energy consumption.

For example, in [4], the authors address energy-aware system design of Wireless Sensor Networks (WSNs). Energy mode signalling and energy scheduling of nodes within a WSN are represented as SDL models, and then analysed. In [13], a framework for modeling, simulation, and code generation of WSNs is presented. The framework is based on Simulink, Stateflow and Embedded Coder, and allows engineers to simulate and automatically generate code with energy as one of the main issues. In [15], a model-driven process to enable a low-cost prototyping and optimization of WSN applications is provided. In this work, a set of modeling languages is the starting point for code generation and performance (with energy consumption) analysis.

The contribution of our approach is a clear separation amongst the software architecture of the application, the hardware and the WSN deployment topology. This promotes reuse of models across projects and organizations. The modeling language for the physical environment supports the analysis and development of WSNs.

VI. Conclusions

In this paper we propose an MDE framework for architecting WSNs. It relies on three modeling languages that allow developers to describe a WSN from three complementary viewpoints: software architecture, low-level hardware details of the nodes, and the physical environment of the WSN. These are linked together by using weaving models. In this work, a dedicated engine to assess the lifetime of the modelled WSN is also presented. Other kinds of code generation or analysis engines can be part of the framework.

As future work we are planning to extend our framework with other *code generation and analysis plugins* (e.g, for performance, security analysis, etc.). Also, we are working on an extension of the framework in which NODEML models are distributed via a *dedicated market*; it will be an on-line repository of nodes definitions with vendors-specific information. Moreover, in the current version of the SAML language conditions and application data definitions are defined as plain strings. As short-term future work, we are working on defining those conditions in a more disciplined manner, so that developers can validate those expressions at design-time. Finally, we are evaluating different solutions for refining and enhancing our (purposefully) simple ENVML modeling language. In this respect, we are doing a survey study for understanding how developers prefer to describe the *physical environment* of the WSN (e.g., 2D vs 3D environment) in order to provide them the solution that better fit with their needs.

Acknowledgment

The authors would like to thank Fabrizio Di Giuseppe and Lorenzo Salvatore, for their help and support in making this work possible.

References

[1] J. Blumenthal, M. Handy, F. Golatowski, M. Haase, and D. Timmermann, "Wireless sensor networks - new challenges in software engineering," in *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference*, vol. 1, sept. 2003, pp. 551 – 556 vol.1.

[2] K. Czarnecki and M. Antkiewicz, "Mapping features to models: A template approach based on superimposed variants," in *GPCE*, 2005, pp. 422–437.

[3] K. Gill, S.-H. Yang, F. Yao, and X. Lu, "A zigbee-based home automation system," *Consumer Electronics, IEEE Transactions on*, vol. 55, no. 2, pp. 422 –430, may 2009.

[4] R. Gotzhein, M. Krämer, L. Litz, and A. Chamaken, "Energy-aware system design with sdl," in *Proceedings of the 14th international SDL conference on Design for motes and mobiles*, ser. SDL'09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 19–33.

[5] D.-M. Han and J.-H. Lim, "Smart home energy management system using ieee 802.15.4 and zigbee," *Consumer Electronics, IEEE Transactions on*, vol. 56, no. 3, pp. 1403 –1410, aug. 2010.

[6] J. L. Hill, "System architecture for wireless sensor networks," Ph.D. dissertation, University of California, Berkeley, 2003, aAI3105239.

[7] ISO/IEC, "ISO/IEC/IEEE 42010:2011 Systems and software engineering – Architecture description," 2011.

[8] K.Pahlavan and P.Krishnamurthy, *Networking Fundamentals*. Chichester, UK: John Wiley and Sons, 2009.

[9] F. Losilla, C. Vicente-Chicote, B. lvarez, A. Iborra, and P. Snchez, "Wireless Sensor Network Application Development: An Architecture-Centric MDE Approach," in *ECSA*, ser. LNCS, F. Oquendo, Ed., vol. 4758. Springer, 2007, pp. 179–194.

[10] I. Malavolta, H. Muccini, P. Pelliccione, and D. Tamburri, "Providing architectural languages and tools interoperability through model transformation technologies," *Software Engineering, IEEE Transactions on*, vol. 36, no. 1, pp. 119 –140, jan.-feb. 2010.

[11] I. Malavolta, 2012, source code of the WSN Modeling Languages metamodels. [Online]. Available: http://www.di.univaq.it/malavolta/files/it.univaq.wsn.zip

[12] L. Mottola and G. P. Picco, "Programming wireless sensor networks: Fundamental concepts and state of the art," *ACM Comput. Surv.*, vol. 43, pp. 19:1–19:51, Apr. 2011.

[13] M. M. R. Mozumdar, F. Gregoretti, L. Lavagno, L. Vanzago, and S. Olivieri, "A framework for modeling, simulation and automatic code generation of sensor network application," in *SECON*, 2008, pp. 515–522.

[14] G. P. Picco, "Software engineering and wireless sensor networks: happy marriage or consensual divorce?" in *Proceedings of the FSE/SDP workshop on Future of software engineering research*, ser. FoSER, NY, USA, 2010.

[15] R. Shimizu, K. Tei, Y. Fukazawa, and S. Honiden, "Model driven development for rapid prototyping and optimization of wireless sensor network applications," in *Proceedings of SESENA '11*. New York, NY, USA: ACM, 2011, pp. 31–36.

[16] J. A. Stankovic, "Research challenges for wireless sensor networks," *SIGBED Rev.*, vol. 1, pp. 9–12, July 2004.