Hindawi
Journal of Sensors
Volume 2020, Article ID 9592836, 13 pages
https://doi.org/10.1155/2020/9592836

Review Article

Wireless Sensor Network Design Methodologies: A Survey

Mohammed Sulaiman BenSaleh,[1] Raoudha Saida,[2,3] Yessine Hadj Kacem,[4] and Mohamed Abid[2,3]

[1]*National MEMS Technology Center, KACST, Riyadh, Saudi Arabia*
[2]*ENIS, CES Laboratory, University of Sfax, Sfax, Tunisia*
[3]*Digital Research Center of Sfax (CRNS), Safx, Tunisia*
[4]*College of Computer Science, King Khalid University, Abha, Saudi Arabia*

Correspondence should be addressed to Raoudha Saida; saidaraoudha@yahoo.fr

Received 29 August 2019; Revised 10 December 2019; Accepted 20 December 2019; Published 25 January 2020

Academic Editor: Matthew Brodie

Wireless sensor networks (WSNs) have grown considerably in recent years and have a significant potential in different applications including health, environment, and military. Despite their powerful capabilities, the successful development of WSN is still a challenging task. In current real-world WSN deployments, several programming approaches have been proposed, which focus on low-level system issues. In order to simplify the design of the WSN and abstract from technical low-level details, high-level approaches have been recognized and several solutions have been proposed. In particular, the model-driven engineering (MDE) approach is becoming a promising solution. In this paper, we present a survey of existing programming methodologies and model-based approaches for the development of sensor networks. We recall and classify existing related WSN development approaches. The main objective of our research is to investigate the feasibility and the application of high-level-based approaches to ease WSN design. We concentrate on a set of criteria to highlight the shortcomings of the relevant approaches. Finally, we present our future directions to cope with the limits of existing solutions.

1. Introduction

WSNs have become an integral part of diverse applications such as environmental monitoring [1], military surveillance [2], and medicine [3] by providing feasible communication, reliable inspection, and performing applications. WSNs are composed of a large number of sensor nodes which are densely deployed and wirelessly communicated to send and receive environmental information. Each sensor node is equipped at least with one or more sensors, a radio transceiver, a processor, and a power supply section. The development of WSNs becomes a very challenging task due to the complexity of such systems. Additionally, several important requirements need to be satisfied during the design of WSN such as the power consumption requirement, which represents the primary key. For this reason, many current researches focus on surveying WSNs. In [4, 5], authors described the concept of WSNs and their characteristics. They described the generations, routing, architecture, and storage management of WSN. Other surveys [6, 7] give an overview of existing routing protocols for sensor networks. In [8], authors presented several existing middleware for sensor networks. In [9], authors reviewed several sensor localization techniques and hierarchical taxonomy and their applications in different context. They presented new sensor localization schemes and their implementation for IoT infrastructure. In the same context [10], authors presented a survey on sensors' device-free localization for smart world. In [11], authors surveyed 9 WSN modeling techniques. They presented how each technique describes the node behavior and the network behavior. They presented also the modeling tool for each approach. However, few works surveyed and discussed the main state of the practice programming approaches and modeling techniques currently used to develop WSN. Motivated by this idea, we present in this work a study that aggregates and discusses existing

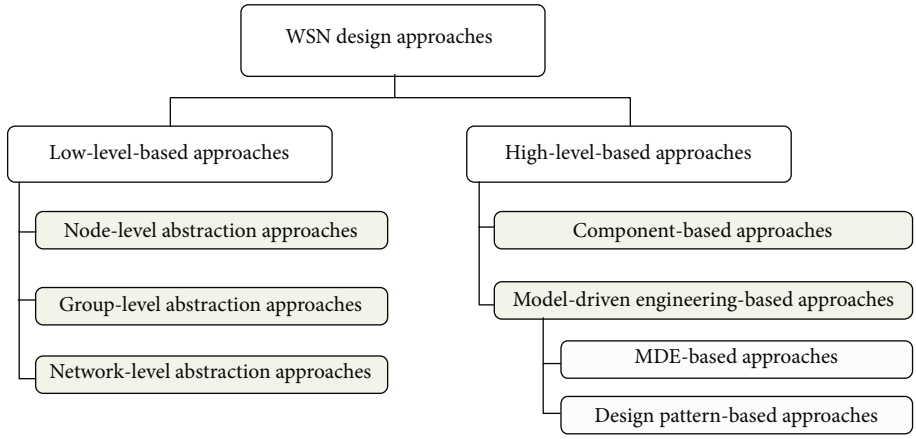

Figure 1: WSN design approaches.

WSN design methodologies. We survey low-level-based approaches that focus on the implementation level and high-level-based approaches that rely on model concept to design WSN systems. The main objective of our research is to investigate the feasibility and the application of high-level-based approaches that help decrease the complexity of the development of WSN systems and to improve maintainability and portability. At present, great attention has been attracted to high-level abstraction design based on MDE methodology [12] and especially using standard mechanisms such as the Modeling and Analysis of Real Time and Embedded Systems (MARTE) [13] profile and design patterns [14]. Representing the WSN system at higher abstraction levels permits reducing the system complexity and increasing the reusability and flexibility of models. It allows also automation and ameliorates model quality. Moreover, it provides the system at an early design stage, which permits revealing errors before the real network deployment.

In this paper, we recall several research works for WSN development. We begin by presenting low-level-based approaches to investigate then the need for appropriate high-level design methods. Figure 1 illustrates the classification of WSN design approaches. We considered a set of comparison criteria related to design environment, power supply design, reconfiguration scenario, and performance evaluation. In Section 2, we define the comparison criteria. In Section 3, we present low-level-based approaches for WSN development. In Section 4, we present high-level-based approaches for WSN development. Finally, we summarize our work and we give perspectives in Section 5.

2. Comparison Criteria

We focus on a set of criteria to compare existing related works (Figure 2). We divide these criteria into four groups: design environment, power supply design, reconfiguration scenario, and nonfunctional property (NFP) verification. In this study, we focus on modeling constraints and requirements typically imposed in terms of energy efficiency and reconfiguration in which we are interested for our future work.

Design environment: this group includes the design abstraction level. We are interested to define if the approach is MDE-based or developed at a low abstraction level. This group defines the modeling standards and techniques. We focus on the use of the UML/MARTE standard and definition and application of WSN design patterns

Power supply design: this group focuses on the modeling of the WSN power supply section. It is an important criterion to be considered since power resources in WSN applications present a primary concern. This group includes the support of a typical power supply section using a local battery. It includes also the modeling of an energy harvesting power supply unit while an energy harvesting alternative [15, 16] is a promising solution to meet the network energy requirements

Reconfiguration scenario: this group focuses on the level and structure of reconfiguration engines. It includes two reconfiguration scenarios and the MAPE (Monitor, Analyzer, Planner, and Executor) loop modules [17]. Indeed, two reconfiguration scenarios are adopted in WSN applications [18]. The first scenario is the node-level-based reconfiguration, which corresponds to hardware and software reconfiguration. The second scenario is the network-level-based reconfiguration, which consists in modifying the network topology. The general structure of the reconfiguration engine is based on the MAPE loop

NFP verification: this group focuses on the support of temporal verification and performance evaluation. It includes works that used transformations from system models to simulation or analysis tool models in order to perform early WSN analysis

3. Low-Level-Based Approaches for WSN Development

Several research works have been contributed in the literature to develop and design WSN systems and their requirements. In the present section, we discuss different low-level-based approaches for WSN. We classified those works into three categories: node-level abstraction, group-level abstraction, and network-level abstraction approaches.

3.1. Node-Level Abstraction Approaches. In node-level abstraction, programmers decompose the global application

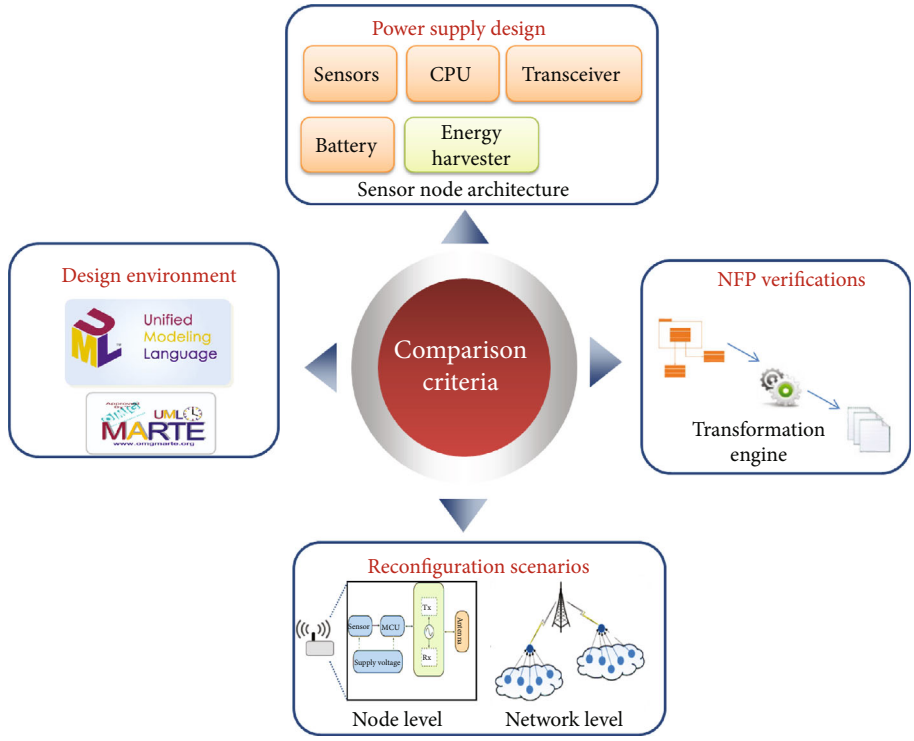

Figure 2: Comparison criteria.

into a set of local node behaviors where an explicit code is running on each node. Virtual machine approaches and embedded operating systems (OSs) are often used in this methodology. OS-based programming focuses on abstracting hardware allowing a flexible control of hardware resources of WSN application. TinyOS [19] (implemented with the programming language nesC [20]) and Contiki [21] are known OSs for WSNs used for the node-level abstraction. Both operating systems support WSN requirements, but there are differences. While TinyOS is better when resources are limited and power applications are low, Contiki is the best choice when flexibility is needed. The development of OSs for wireless sensor devices has been considered a key element for the IoT systems [22]. Several OSs are proposed in this context such as RIOT [23], Mbed OS [24], or FreeRTOS [25], which offer preemptive priority-based task schedulers, include full real-time support, and facilitate IoT application development. Mate [26] is a known virtual machine in WSNs. This project focuses on the need for new programming examples to overcome WSN constraints such as limited energy supply and limited bandwidth. A virtual machine approach provides the dynamic reprogrammability, but it suffers from the overhead that the instructions introduce. Several simulator environments are used for WSN research [27, 28]. The most cited is the NS-3 simulator which is a discrete-event simulator. NS-3 [29, 30] was developed in the C++ programming language, with an optional Python [31] scripting interface. Besides the scalability and performance improvements, simulation nodes have the ability to support multiple radio interfaces and multiple channels. Furthermore, NS-3 supports a real-time schedule that makes it possible to interact with a real system.

3.2. Group-Level Abstraction Approaches. Each node associated with a group-level programming entity is considered a neighbor node for other nodes in the same network. When nodes are grouped based on physical closeness (geographical distance, number of communication hops, etc.), the group is called a neighborhood-based group, such as in Hood [32] and Abstract Region [33] that are examples of neighborhood programming abstractions for WSNs. These algorithms provide local data processing within a neighborhood. Hood provides support to design distributed algorithms in terms of the neighborhood abstraction. It uses data sharing to support scalability and collaboration. It employs a caching technique to save energy, reduces communication failures between nodes, and uses mirror to reflect time synchronization. Abstract Region provides interfaces for identifying neighboring nodes, sharing data, and data reduction within local neighborhoods. Power consumption and scalability are supported through data sharing.

As for Hood, it provides a caching technique to reduce failures in the network. It provides also a way to adapt to different network requirements and conditions, to satisfy different levels of energy and bandwidth usage, and to attain the accuracy level of shared operations. When the group is constructed according to logical properties (node type, sensor input, etc.), it is called a logical group. An example of a logical-based group abstraction is EnviroTrack [34]. It is an application used specifically for target tracking where a set of nodes that detect the same event are grouped together. Like Hood and Abstract Regions, EnviroTrack provides the data sharing and aggregation facilities to satisfy WSN requirements. However, in a more dynamic situation, EnviroTrack provides the best support. In [5], authors

proposed the SPIDEY language, another example of a logical-based group, where a set of nodes are grouped based on their shared properties. The node is represented by both static (e.g., node type) and dynamic (e.g., sensor readings) attributes to determine the nodes' logical neighbors. As outlined programming methodologies, SPIDEY uses a data sharing mechanism to achieve several requirements and also provides a redundancy mechanism to avoid failures in the network. ZigBee technology [35] is considered one of the most deployed wireless technologies. It supports the mesh network topology that uses the most cost-effective path allowing multihop communication. Hence, mesh connection is secured, flexible, scalable, and reliable. It consists of three roles of nodes: a coordinator, several routers, and end devices connected. The mesh topology provides packets passing through multiple hops to reach destinations and communication between any source and destination in the network.

3.3. Network-Level Abstraction Approaches. In the network-level abstraction (called also macroprogramming), the whole sensor network is treated as a single system describing the global behavior. TinyDB [7] views the whole network as a database system. It allows users to issue queries in a declarative SQL-like language. For achieving energy efficiency, TinyDB focuses on when, where, and how to sample and deliver the data. TinyDB also optimizes the routing tree for disseminating a query and collecting the results. Moreover, there are limitations of database abstraction; for example, the use of only one table accessible on time is not favorable for heterogeneous sensors. Regiment [11] is a functional language specially designed for macroprogramming sensor networks that allows the direct use of a program state used to represent the finding of each individual node. Kairos [36], another example of programming abstraction, allows macroprogramming. It uses a caching technique to reduce power consumption and communications.

3.4. Synthesis. In the previous sections, we concentrated on low-level-based approaches that have been investigated for WSN development. We derive from this study that the presented approaches offer rich support for WSN development. They considered several WSN features such as OS, power supply, communication capabilities, and reconfiguration issues. Moreover, they capture WSN characteristics at three different levels including node, group, and network levels. The current design of WSN occurs at the implementation level. This leads to increasing the complexity of such systems since they are platform-specific dependent. Raising the abstraction level enables designers to cope with the challenging increasing complexity. In this context, we detailed in the next section high-level-based approaches for WSN development.

4. High-Level-Based Approaches for WSN Development

The use of modeling techniques and languages involves the WSN design at higher abstraction levels, facilitates analysis steps, and resolves problems before deployment. The main motivation for applying high-level-based approaches is that new applications can be created with less effort than in traditional approaches. In the present section, we present firstly some existing component-based approaches for modeling WSN. Then, we recall WSN approaches based on the MDE methodology.

4.1. Component-Based Approaches. Many of modeling techniques support components to model the system because component-based modeling offers great abstraction by providing interface-based interaction between elements. We examine for each approach its modeling language, scopes, and elements.

4.1.1. High-Level SDL Models (HL-SDL). In [37], authors represented the HL-SDL modeling language that uses the Specification and Description Language (SDL) [38] to model, simulate, and verify communication protocols. SDL is used to model the component-based architecture of TinyOS using the SDL process. The system is modeled as a collection of processes and channels. The SDL approach provides the modeling of node behaviors and does not permit the modeling of the network architecture. Moreover, the generated code is created manually by the designer.

4.1.2. Insense. Authors proposed the Insense language in [39] for supporting a component-based model for WSN applications. Components can be hardware or software entities of sensor nodes, and they communicate synchronously via directional channels which abstract over communication and synchronization. This technique does not support the modeling of a WSN system architecture.

4.1.3. SensorML. Authors introduced the SensorML [40] language in order to provide support for modeling sensor specifications such as physical location, hardware, and sensor type. In the SensorML model, components are represented as processes linked together through inputs and outputs. The model includes physical elements such as sensors and actuators and nonphysical components such as mathematical equations. SensorML supports the modeling of different elements of the network. But it lacks support for testing and validating the design before its real deployment.

4.1.4. UM-RTCOM. The UM-RTCOM model, which is developed in [41], presents a component framework for developing wireless sensor and actor network applications. This model contains sensors, actors, and a coordinator. The communication between sensors and actors and between actors and a coordinator is via channels which are modeled as a tuple that allows one-to-many and many-to-one communication. The UM-RTCOM model provides three types of components: generic components, active components, and passive components. The UM-RTCOM model performs different kinds of analysis such as liveness property, real-time, and deadlock freedom.

4.1.5. MathWorks Modeling Approach. Authors proposed in [42] a framework for modeling, simulation, and generation of code for WSN applications based on MathWorks tools. Sensor nodes are modeled by using Stateflow and Simulink

blocks. Connectivity between sensor nodes is represented by a communication medium block which is implemented on the C language. MathWorks provides analysis tools such as animated state charts, chart displays, and scopes to perform WSN analysis. After modeling and simulation, the code application is generated using the Target Language Compiler, which generates the nesC code for TinyOS and the C code for MANTIS [43].

4.1.6. SystemC. In [44], authors introduced a model of wireless sensor network nodes using SystemC-AMS which is a C++-based language. Each node is represented as a module which represents a model for the sensor, the A/D converter, the microcontroller, and the RF transceiver. SystemC-AMS models the system behavior using a data flow diagram. However, this technique does not support the network architecture modeling. Another related work was proposed in [45]. Authors presented the XRM modeling language, which is an extension language of Reactive Modules (RMs), to model the sensor network. Each node is captured as a module which contains methods and variables describing the node behavior, power consumption, memory, and communication capabilities. XRM supports also the modeling of the WSN system architecture. In [46], authors utilized the PROMELA model to check ad hoc WSN system communication. The network is modeled by using communicating finite-state machines. The PROMELA modeling language is the input of the model checker SPIN. The SPIN model checker is the tool used for verifying the WSN specifications, specifically the network connectivity.

4.1.7. Middleware. A component-based architecture for adaptive WSN was developed in [47]. Authors used middleware development to enable adaptability in service-oriented WSNs. They followed the MAPE loop to attend to self-adaptive WSN requirements. The reference architecture is created based on layer architectural style, component-based and service-oriented architectural style, and decorator pattern. In this work, verification and test phases are absent.

4.1.8. Synthesis. The reviewed component-based techniques enable designers to represent the WSN design at higher abstraction levels, which facilitates testing and verification of errors before the real deployment. They used different modeling languages to capture WSN features. They tackle either the node behavior or the WSN system architecture. Despite its benefits, none of the presented techniques have presented standard modeling languages to address the behavior and the structure of the WSN design at a much higher level of abstraction. In this context, MDE design methodologies are relatively leveraged. MDE affords many of the benefits to software engineering. An overview of the MDE paradigms will be accurately described in the following section.

4.2. Model-Driven Engineering for WSN Development. In this section, we first give an overview about the existing MDE-based approaches for WSN development. We focus on UML/MARTE-based approaches. Then, we provide a view on design pattern-based approaches.

4.2.1. MDE-Based Approaches for WSN. The MDE [12] paradigm is a software engineering approach based on exploiting models in order to address the complexity of embedded systems. This software development methodology deals with the shortcomings of complex system development and reduces the system development costs and time. The MDE has widely contributed to supporting the development life cycle of embedded systems in many fields such as WSN design and development [48–50], energy supply designs for WSN [51], and self-adaptive system development [52, 53]. The MDE has several advantages. First, this approach is aimed at increasing the abstraction level of development and at discarding the low-level details. Second, MDE permits being less prone to error, because it enables the system analysis at an early design stage, which permits revealing errors before the real network deployment. Third, MDE permits a set of model transformations and refinements in order to generate codes or analyze systems. Therefore, the code generation task becomes easier than in traditional software techniques.

The MDE is based on several basics and concepts [54], namely, the model, the metamodel, the model transformation concept, and the UML extension mechanisms.

Researchers on WSN systems have proven the effectiveness of the MDE paradigm in software development. It helps in reducing the design complexity of WSN applications through its principles of abstraction, separation of concerns, reuse, and automation. The present section recalls several modeling approaches based on the MDE methodology for WSN design. We are interested in power supply modeling, on the one hand, and in reconfiguration modeling for WSN, on the other hand. Energy efficiency represents a primary key for most research studies, so does the need for abstracting energy details. In this context, an MDE-based framework methodology is proposed in [49]. It is aimed at defining an architecture for WSN and focuses on energy consumption analysis. This framework proposed three modeling languages that allow modeling separately the software architecture of the WSN application (Software Architecture Modeling Language), the low-level details of each type of nodes used in the network (Node Modeling Language), and the physical environment where the WSN nodes are deployed (Environment Modeling Language). The proposed metamodel of the Node Modeling Language is composed of stereotypes permitting defining the node features. It uses *EnergySource* and *HarvestedEnergySource* stereotypes to model energy information, the *RadioCommunicationDevice* stereotype to present the transceiver unit, the *Sensor* stereotype to model sensors, and the *Node* stereotype and the *NodeSpecification* stereotype to represent sensor nodes. An analysis step is also performed through automatic code generation. Another high-level application model was introduced in [55], which is composed of a feature model and a class diagram annotated with the WiSeN profile to support WSN modeling. First, the model defines different features related to application, network, programming language, hardware, and communication in the WSN. Then, authors proposed the WiSeN profile which is an extension of the UML/MARTE profile for supporting communication, sensing, and synchronization in WSN design. This profile has

extended the MARTE profile by a set of stereotypes in order to allow the specification of complementary information for WSN features, such as the *Sensor* stereotype extending «HwSensor» to map sensors and their characteristics and the *Synchronize* stereotype extending «SynchronizationResource» used to model synchronization between sensor nodes. In addition, it supports the *Communication* stereotype extending «SaCommStep» to transmit/receive messages. To represent nodes, the WiSeN profile uses the *Node* stereotype extending «ResourceUsage». It uses the *MsgPackage* stereotype extending «MessageComResource» to give information about the structure of message. The WiSeN profile contains other information that does not exist in the MARTE standard. It uses the *Gateway* stereotype to assist in the communication with the external system. The WiSeN profile can be refined and extended to address the power supply section where the limited energy is the most important constraint in WSNs. In addition, this profile can provide new extension of MARTE to support the reconfigurable aspect of the WSN system since MARTE contains concepts related to reconfiguration. Authors proposed in [51] a high-level methodology based on the MARTE profile for designing the power section for a WSN node.

Four main elements are defined in the design: energy scavenging device, energy accumulating device, consumption of the node, and recharging energy. This methodology provides an extension of the «HW_PowerSupply» with «HW_Harvesting» to describe the harvester, and the «HW_Harvesting» is also extended with «HW_PV» to describe the harvesting done by a solar panel. In addition, to describe the details of the energy accumulating device, authors added modifications to the «HW_Battery», extending «HW_PowerSupply». This methodology provides an extension to support harvesting done by a solar panel so we can extend the MARTE profile to support more energy harvesting such as vibration. In addition, the WSN requirements (e.g., characteristics of sensors, communication links, and characteristics of nodes) must be considered in the design of the power section. To address the design of adaptive WSN, multiple works have been proposed. Authors provided in [56] a model-driven approach for designing and verifying autonomic network behaviors. They used a generic control loop based on the decision-making element for establishing the autoconfiguration in the network. They proposed a model-driven methodology based on metamodeling, structural modeling, and behavioral modeling methods. Then, they provided an evolvable and holistic model for autonomic networking and autoconfiguration. But still the autonomic code generation is not defined for validation and verification purposes. Another work was proposed in [57], which deals with adaptive sensor networks. Authors proposed a model-driven development (MDD) framework for modeling and executing WSN systems. They defined a UML profile, which describes low-level details of WSN and provides high-level design presentation. Moreover, they used the Matilda UML virtual machine for executing and validating WSN application. Matilda permits the automatic code generation through M2T transformation from the proposed UML profile. The proposed MDD framework is designed based on BiSNET architecture that addresses the dynamic adaptability required in nodes and the network. The BiSNET consists of agents and middleware platforms in order to achieve adaptability requirements. In [58], authors proposed an MDE approach to develop WSN application, which allows the flexibility and reusability of their designs. They considered three levels of abstraction: domain-specific models, platform-independent models, and platform-specific models. Then, automatic model transformations are refined until the final code is produced. M2M and M2T transformation engines are considered using eclipse plugins to obtain a NesC code for the TinyOS-based node. The MindCPS approach is proposed by [59] to design and develop Cyber-Physical Systems (CPS). It is a model-driven development (MDD) solution that defines a DSL for describing primitives of the autonomic behavior of CPS and produces model-to-code transformations. To achieve the autonomic adaptation, authors used the MAPE loop model [17]. MAPE loop modules are used to monitor the different sensor states in order to enable the system functionalities.

(1) Synthesis. To deal with the increasing complexity of WSN systems, several works have proposed the use of high-level methodologies to fulfill the strict WSN constraints and facilitate the development process. Most reviewed approaches used high-level modeling concepts to specify several WSN basics. In some works, authors used UML and MARTE annotations. Others proposed new UML/MARTE extensions to support relevant WSN information specifically in terms of power consumption and reconfiguration concerns. In other cases, some approaches proposed a whole development cycle that starts with a modeling step to lead to a validation step through a set of model transformation rules. Therefore, we can assume that the MDE methodology is the right tool to deal with WSN complexity. Nonetheless, the relevant existing studies still present some limitations. There is a lack of adaptation process that designs and validates WSN models. In addition, most adaptive works address the node-level-based reconfiguration scenario. They deal with hardware reconfiguration such as sensor node adaptation and software reprogramming such as OS programming. However, architectural reconfiguration is not well tackled in high-level modeling. In addition, MAPE loop modules are not considered in the modeling phase. On the other hand, explicit support for power requirement modeling is absent. Accordingly, there is a lack of high level abstraction modeling of energy harvesting modules. Moreover, NFP verification and validation steps are not well performed. The use of high-level modeling languages and methodologies helps designers to cope with the growing complexity of WSN systems. Nevertheless, there is still a great need for generic and reusable models to help designers to easily model their systems. In this regard, design patterns [14] represent a promising solution since they promote generic models used to propose solutions for recurrent problems. We give in the following some basic definitions on design patterns, and we recall existing pattern-based works for WSN development.

4.2.2. Design Patterns for WSN Development. Design patterns [14] are widely affirmed as a potential approach towards

software design. A design pattern is used to capture the application flow and the design components at a higher abstraction view that guarantees the reusability of the design. Indeed, a design pattern is a general and reusable solution to a recurrent problem in software design. It has been proven to be highly effective in modeling and representing complex systems. In addition, it facilitates the reuse of software models and improves the quality of software. Each design pattern is described by essential elements following the pattern template proposed in [14]. A pattern *name* is a handle that helps in identifying the pattern, its intent, and its solution. The pattern *intent* describes the goal behind the design pattern. The pattern *problem* consists in defining the context of pattern application. Lastly, the pattern *structure* is a graphical representation of the pattern where class and sequence diagrams can be used.

With the growing development of WSNs and the programming challenges of sensor nodes, software designers have shown a significant interest in representing design patterns for WSN development. In this context, a design pattern for a sensor node was developed in [60]. The proposed pattern described the architecture of a sensor node which includes sensors, a power source, communication channels, and memory. Authors presented the static and dynamic aspects of the proposed pattern using, respectively, the UML class diagram and UML sequence diagram. Another related work is proposed in [61] that used the previous pattern to design the structure of the network and connections between devices in order to achieve performance objectives. Authors presented dynamic diagrams of several use cases which include network reconfiguration, data gathering, and others. Another software design pattern is described in [62] for the TinyOS operating system which is well described in WSN applications. They presented two behavioral design patterns, dispatcher and decorator, and three structural design patterns. In [63], authors presented a set of design patterns using UML diagrams that help designers in defining a software design of middleware and hardware modules for a WSN system in order to optimize analysis for power consumption at an early stage of development. Several design patterns are defined in [64] for unifying different abstractions and middleware such that users can manipulate various WSNs using different programming languages. In [65], authors proposed a programming pattern named sMapReduce for enabling sensor network applications targeting applications with complex data aggregation. A set of design patterns is developed in [66] towards the self-adaptability for RTES. In this work, authors suggested four design patterns as solutions for four MAPE loop modules: Monitor, Analyzer, DecisionMaker, and Actor. The proposed patterns are described following the pattern template [14]. Authors used the UML diagrams annotated with the UML/MARTE profile stereotypes. They used class diagrams to present the structural views of the patterns and sequence diagrams to explain the behavioral views. The authors combined the four patterns based on integration rules in order to form the design of the MAPE adaptation loop and enable its application. The RTE Monitor pattern allows continuous monitoring to reflect the current state of the system in order to detect trigger events and relevant changes. It considers the system stability problem by minimizing the events triggered through the choice of the significant context variations. It also deals with concurrency and real-time specifications associated with the control tasks. The RTE Analyzer pattern enables the verification of the constraints which fit an RTE system in order to request adaptation if needed. It handles concurrency and real-time specifications associated with the control tasks. The RTE DecisionMaker pattern decides an adaptation plan once an adaptation request is sent. It provides adaptation policies. It also defines what elements to modify and how to meet constraints and requirements. The adaptation strategy can be based on changeable parameters or the modification of the structure of the RTE system. The RTE Actor pattern permits the final adaptation plan within a set of adaptation actions, which are related to the relative changeable element of a system. It specifies the effector responsible for an adaptation action.

(1) Synthesis. According to the previous study, research works based on design patterns are still limited in the WSN domains. Most existing approaches focus on either sensor node components or network architecture. Few works dealt with reconfiguration scenarios. Nevertheless, the studied works do not cope with WSN requirements such as power consumption or real-time constraints. Moreover, they do not offer explicit support for architectural reconfiguration in WNS. They offer limited support for modeling of WSN systems and most research does not explore high-level modeling languages and specific standards.

5. Summary and Synthesis

In this paper, we presented a study about programming methodologies and modeling techniques for WSN development. We classified those works into two categories according to the abstraction level of their design. The first category concerned low-level techniques for WSN and particularly programming models. The second category dealt with high-level-based approaches including component-based modeling techniques and MDE-based approaches and particularly those that used UML and MARTE standards and pattern-based concepts. Table 1 summarizes the previous sections by illustrating a comparison of the discussed related works.

We conclude from our study that the development of WSN can be investigated at different abstraction levels. Various development techniques have been presented at low abstraction level to address either the node behavior or the network architecture [7, 19, 21]. These approaches have proven their effectiveness for developing WSNs; however, there is a lack of standard mechanisms that fit with WSN complexity. Therefore, raising the abstraction level is a promising solution to handle the shortcomings of low-level-based approaches. In fact, MDE and specifically the MARTE profile have received enormous attention in WSN development. The UML/MARTE profile offers the modeling of hardware and software elements of the WSN system. Table 2 illustrates how existing high-level approaches have applied UML concepts and MARTE stereotypes to model WSN elements.

Table 1: Comparison of development approaches for WSN.

Related works	Design environment			Power supply design		Reconfiguration scenario			NFP verification	
	MDE	UML/MARTE	DesignPatternsforWSN	EnergySourceConepts	Energyharvestingconcepts	Node-levelbasedreconfiguration	Network-levelbasedreconfiguration	MAPEloopmodules	Classicalanalysis	Automaticanalysis
[39, 42]	-	-	-	-	-	-	-	-	-	-
[21, 26]	-	-	-	X	-	X	-	-	-	-
[5, 7, 11, 32–34, 36, 37, 44]	-	-	-	X	-	-	-	-	-	-
[40]	-	-	-	-	-	-	-	-	X	
[41, 45]	-	-	-	X	-	-	-	-	-	X
[57]	-	-	-	-	-	X	-	X	-	-
[46]	X	-	-	X	X	-	-	-	-	X
[47, 58]	X	-	-	-	-	X	-	-	-	X
[49]	-	X	-	-	-	-	-	-	X	-
[55]	-	X	-	X	X	-	-	-	X	-
[51]	X	-	-	-	-	-	X	-	-	-
[56]	X	-	-	X	-	-	-	-	-	X
[59]	X	-	-	X	-	X	-	X	-	X
[60–62]	-	-	X	-	-	-	-	-	-	-
[63]	-	-	X	X	-	-	-	-	-	X
[64, 65]	-	-	X	X	-	-	-	-	-	-
[66]	-	X	X	-	-	X	-	X	-	-
Proposed approach	X	X	X	X	X	-	X	X	-	X

Legend: X: supported; -: unsupported.

Table 2: Existing UML concepts and MARTE stereotypes to model WSN elements.

Approach Modeling scope	[49]	[55]	[51]	[57]	[59]	[60, 61]	[66]
Node	«Node»	«MARTE::GRM::ResourceUsage::Node»	Not supported	«SensorNode»	Not supported	«Node»	Not supported
Gateway	Not supported	«Gateway»	Not supported	Not supported	Not supported	Not supported	Not supported
Actuator	«Actuator»	«MARTE::HRM::HWActuator»	Not supported	Not supported	«Actuator»	Not supported	Not supported
Communication	«RadioCommunicationDevice»	«MARTE::SAM::SaCommStep::Communication»	Not supported	«Wirelesslink»	Not supported	«Transeiver»	Not supported
Sensor	«Sensor»	«MARTE::HRM::HwSensor»	Not supported	Not supported	«Sensor»	«Sensor»	Not supported
Energy source	«Energy-Source»	«MARTE::HRM::HWPowerSupply»	«MARTE::HRM::HWBattery»	Not supported	Not supported	«Power-Supply»	Not supported
Energy harvesting	«Harvested-EnergySource»	Not supported	«MARTE::HRM::Hw_PV»	Not supported	Not supported	Not supported	Not supported
Reconfiguration	Not supported	Not supported	Not supported	Not supported	«RealTime-Symptom» «Action» «Plan»	Not supported	«MARTE::GRM::RtUnit::Monitor» «MARTE::GRM::RtUnit::Analyzer» «MARTE::GRM::RtUnit::DecisionMaker» «MARTE::GRM::RtUnit::Actor»

Table 3: Evaluation of the proposed MDE-based approach.

	Design environment	Power supply design	Reconfiguration scenario	NFP verification
Proposed approach	(i) New UML/MARTE extensions (ii) MAPE loop design patterns (iii) M-2-T transformation	(i) It supports energy source (battery) (ii) It proposes new energy harvesting source design	(i) It proposes a new network-level-based reconfiguration strategy based on the MAPE loop	(i) Automatic generated analysis scripts (ii) Simulation and real deployment steps

The studied approaches [49, 55] used MARTE to deal with the modeling of the power supply section of WSN. In the same context, [51] defined new extensions to MARTE to support only solar energy harvesting modeling. The use of high-level mechanisms and languages facilitates the designer's tasks, decreases the complexity of WSN systems, and minimizes cost and time to market. However, the majority of studies tackle only node-level-based reconfiguration [58, 59]. They addressed hardware and software reconfiguration in a sensor node whereas network-level-based reconfiguration modeling is absent. According to our study, network-level-based reconfiguration and energy harvesting modeling are not well considered by model approaches based on the MARTE profile. We therefore need to add extensions to MARTE to allow a generic specification and modeling of energy harvesting modules. We need also to create a new package extending the MARTE standard for supporting architectural reconfiguration for WSN. Additionally, the automatic analysis of WSN regarding power efficiency and reconfiguration concerns is not well supported. Design patterns have been also investigated to cope with RTES complexity. They offer generic and reusable models that are used to solve a recurrent problem. In addition, they facilitate the reuse of software models and improve the quality of software. We derived from our study that design patterns are still not well tackled. Most of existing works described either node components or network architecture [60, 61]. Only one work [66] dealt with patterns devoted for adaptive systems. This work offers new MARTE extensions for presenting MAPE loop modules. We cited in Table 2 the relative MARTE extension: «MARTE::GRM::RtUnit::Monitor», «MARTE::GRM::RtUnit::Analyzer», «MARTE::GRM::RtUnit::DecisionMaker», and «MARTE::GRM::RtUnit::Actor». Yet no attention was given to real-time constraints in the development of WSN patterns.

We can conclude from our study that existing approaches on model-based WSN design present shortcomings regarding energy and reconfiguration requirements. In addition, there is still a lack of high-level modeling standards and reusable models that support the specification of requirements related to WSN development. Moreover, there are still open issues regarding the verification of the system's nonfunctional properties (NFPs).

In this regard, we plan, as a future work, to propose an MDE-based approach for developing an energy-aware reconfigurable WSN. The proposed framework used MDE concepts, UML/MARTE profile, and design patterns to support high-level specification and automatic analysis of WSN. A first study was devoted to the energy sources in WSN. As we mentioned, existing studies lacks explicit support for power requirement modeling. Accordingly, high abstraction modeling of energy harvesting modules is absent. In this context, we proposed well-structured support for energy harvesting specification based on the MARTE profile. Given the fact that WSN can be analyzed similar to a real-time system, MARTE can be suitably adopted to support the modeling of such systems. We extended this profile with seven new energy harvesting devices including vibration, thermal, kinetic, acoustic noise, RF, biochemical, and hybrid energy harvesting types. A second study was devoted to the reconfiguration scenarios in WSN. Our study shows that existing works focus on low-level specifications. They addressed the node-level-based reconfiguration scenario and dealt with hardware and software reconfiguration. Moreover, they do not offer explicit support for network-level-based reconfiguration. To solve all these problems, we defined an MDE-based process for supporting the architectural reconfiguration in WSN applications, which we named EARN-(Energy-Aware Reconfigurable Node-) MDE-based process. It allows the automatic generation of a high-level reconfigurable WSN model in an energy harvesting environment. It is based on the detection, instantiation, and integration of design patterns. It starts by annotating the system model with reconfiguration semantics. Then, the pattern instances are automatically generated and integrated into the initial system model based on a set of instantiation and integration rules.

Finally, verification and simulation steps are realized to check the system constraints. For this end, model-to-text (M2T) transformations are performed to generate scripts for simulation purposes and NFP verification. Moreover, it is important to mention that our research work is initiated in the EARN project [67]. We will thus test and evaluate our proposed process in terms of resource efficiency on a real demonstrator platform. Table 3 summarizes how the proposed approach achieves the design criteria explained in Section 2. The development of WSN using high-level techniques and reusable models is a promising solution to decrease the growing complexity of heterogeneous systems such as the IoT (Internet of Things) systems. The use of MDE and models has been proven as an enabling and promising solution [68, 69] through its principles of abstraction, separation of concerns, reuse, and automation. Indeed, in MDE, models represent the core concept and are considered in abstraction of the system under development. In addition to abstraction, automation is performed in terms of model manipulation and refinement through model transformations. Additionally, MDE solves the challenge heterogeneity management of software and hardware thanks to the use of modeling languages, more specifically domain-specific ones. Models defined through these languages are meant to

be much more human-oriented than common code artifacts that enhance reusability. In this direction, an interesting perspective thus is to extend the proposed EARN-MDE process for designing complex and critical reconfigurable applications. In fact, the EARN-MDE process offers powerful support for the management of heterogeneity of software and hardware by using the MARTE profile that permits the modeling of hardware components as well as allocations of software to hardware. Additionally, it supports the MAPE loop as a reconfigurable managing system that optimizes the management of the system even in complex situations.

Conflicts of Interest

The authors declare that they have no conflicts of interest

Acknowledgments

This work was supported by King Abdulaziz City for Science and Technology (KACST) and Digital Research Center of Sfax (CRNS).

References

[1] N. K. Suryadevara, S. C. Mukhopadhyay, S. D. T. Kelly, and S. P. S. Gill, "WSN-based smart sensors and actuator for power management in intelligent buildings," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 564–571, 2015.

[2] M. P. Durisic, Z. Tafa, G. Dimic, and V. Milutinovic, "A survey of military applications of wireless sensor networks," in *2012 Mediterranean Conference on Embedded Computing (MECO)*, pp. 196–199, Bar, Montenegro, June 2012.

[3] H. Furtado and R. Trobec, "Applications of wireless sensors in medicine," in *2011 Proceedings of the 34th International Convention MIPRO*, pp. 257–261, Opatija, Croatia, May 2011.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[5] L. Mottola and G. P. Picco, "Logical neighborhoods: a programming abstraction for wireless sensor networks," in *Distributed Computing in Sensor Systems. DCOSS 2006. Lecture Notes in Computer Science, vol 4026*, P. B. Gibbons, T. Abdelzaher, J. Aspnes, and R. Rao, Eds., pp. 150–168, Springer, Berlin, Heidelberg, 2006.

[6] Q. Jiang and D. Manivannan, "Routing protocols for sensor networks," in *First IEEE Consumer Communications and Networking Conference, 2004. CCNC 2004*, pp. 93–98, Las Vegas, NV, USA, January 2004.

[7] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tinydb: An acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 122–173, 2005.

[8] M. M. Molla and S. I. Ahamed, "A survey of middleware for sensor network and challenges," in *2006 International Conference on Parallel Processing Workshops (ICPPW'06)*, pp. 6–228, Columbus, OH, USA, August 2006.

[9] R. C. Shit, S. Sharma, D. Puthal, and A. Y. Zomaya, "Location of things (lot): a review and taxonomy of sensors localization in Iot infrastructure," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2028–2061, 2018.

[10] R. C. Shit, S. Sharma, D. Puthal et al., "Ubiquitous localization (UbiLoc): a survey and taxonomy on device free localization for smart world," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3532–3564, 2019.

[11] R. Newton, G. Morrisett, and M. Welsh, "The regiment macroprogramming system," in *2007 6th International Symposium on Information Processing in Sensor Networks*, pp. 489–498, Cambridge, MA, USA, April 2007.

[12] D. C. Schmidt, "Guest editor's introduction: model-driven engineering," *Computer*, vol. 39, no. 2, pp. 25–31, 2006.

[13] OMG Object Management Group, *A UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems, ptc/2011-06-02*, Object Management Group, 2011.

[14] J. Vlissides, R. Helm, R. Johnson, and E. Gamma, *Design patterns: elements of reusable object-oriented software*, Reading: Addison-Wesley, 1995.

[15] F. K. Shaikh and S. Zeadally, "Energy harvesting in wireless sensor networks: a comprehensive review," *Renewable and Sustainable Energy Reviews*, vol. 55, pp. 1041–1054, 2016.

[16] S. Basagni, M. Y. Naderi, C. Petrioli, and D. Spenza, "Wireless sensor networks with energy harvesting," in *Mobile Ad Hoc Networking, Cutting Edge Directions*, S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, Eds., pp. 701–736, John Wiley & Sons, 2013.

[17] Autonomic Computing, *An architectural blueprint for autonomic computing*, IBM White Paper, 31, 2006.

[18] C. Rajasekaran, R. Jeyabharath, and P. Veena, "Hardware-software reconfigurable techniques for wireless sensor network," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 8, no. 17, pp. 1855–1862, 2014.

[19] P. Levis, S. Madden, J. Polastre et al., "TinyOS: an operating system for sensor networks," in *Ambient Intelligence*, pp. 115–148, Springer, Berlin, Heidelberg, 2005.

[20] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language," *ACM SIGPLAN Notices*, vol. 49, no. 4, pp. 41–51, 2014.

[21] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *29th Annual IEEE International Conference on Local Computer Networks*, pp. 455–462, Tampa, FL, USA, November 2004.

[22] R. Rodriguez-Zurrunero, R. Utrilla, A. Rozas, and A. Araujo, "Process management in Iot operating systems: cross-influence between processing and communication tasks in end-devices," *Sensors*, vol. 19, no. 4, p. 805, 2019.

[23] E. Baccelli, O. Hahm, M. GÃijnes, M. WÃđhlisch, and T. C. Schmidt, "RIOT OS: towards an OS for the Internet of things," in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 79-80, Turin, Italy, April 2013.

[24] Arm Mbed, "Mbed OS," https://www.mbed.com/en/platform/mbed-os/.

[25] FreeRTOS, "The FreeRTOS™ reference manual," https://www.freertos.org/Documentation/FreeRTOS_Reference_Manual_V9.0.0.pdf.

[26] P. Levis and D. E. Culler, "Maté: a tiny virtual machine for sensor networks," in *ASPLOS X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pp. 85–95, San Jose, CA, USA, October 2002.

[27] M. Živković, B. Nikolić, J. Protić, and R. Popović, "A survey and classification of wireless sensor networks simulators based

on the domain of use," *Adhoc and Sensor Wireless Networks*, vol. 20, 2014.

[28] A. S. Toor and A. K. Jain, "A survey on wireless network simulators," *Bulletin of Electrical Engineering and Informatics*, vol. 6, no. 1, pp. 62–69, 2017.

[29] "Ns-3 overview," August 2010, http://www.nsnam.org.

[30] M. Lacage, "Experimentation with ns-3," *Trilogy Summer School*, vol. 14, 2009.

[31] A. L. S. Saabith, M. M. M. Fareez, and T. Vinothraj, "Python current trend applications-an overview," *International Journal of Advance Engineering and Research Development*, vol. 6, no. 10, 2019.

[32] K. Whitehouse, C. Sharp, D. E. Culler, and E. A. Brewer, "Hood: a neighborhood abstraction for sensor networks," in *MobiSys '04, Proceedings of the Second International Conference on Mobile Systems, Applications, and Services*, pp. 99–110, Hyatt Harborside, Boston, MA, USA, June 2004.

[33] M. Welsh and G. Mainland, "Programming sensor networks using abstract regions," in *1st Symposium on Networked Systems Design and Implementation (NSDI 2004)*, pp. 29–42, San Francisco, CA, USA, March 2004.

[34] T. Abdelzaher, B. Blum, Q. Cao et al., "Envirotrack: towards an environmental computing paradigm for distributed sensor networks," in *24th International Conference on Distributed Computing Systems, 2004. Proceedings*, pp. 582–589, Tokyo, Japan, March 2004.

[35] G. Omojokun, "A survey of Zigbee wireless sensor network technology: topology, applications and challenges," *International Journal of Computer Applications*, vol. 130, no. 9, pp. 47–55, 2015.

[36] R. Gummadi, O. Gnawali, and R. Govindan, "Macro-programming wireless sensor networks using *Kairos*," in *Distributed Computing in Sensor Systems. DCOSS 2005. Lecture Notes in Computer Science, vol 3560*, V. K. Prasanna, S. S. Iyengar, P. G. Spirakis, and M. Welsh, Eds., pp. 126–140, Springer, Berlin, Heidelberg, 2005.

[37] D. Dietterle, J. Ryman, K. Dombrowski, and R. Kraemer, "Mapping of high-level SDL models to efficient implementations for TinyOS," in *Euromicro Symposium on Digital System Design, 2004. DSD 2004*, pp. 402–406, Rennes, France, August-September 2004.

[38] K. K. Sandhu, "Specification and description language (SDL)," in *IEE Tutorial Colloquium on Formal Methods and Notations Applicable to Telecommunications*, London, UK, March 1992.

[39] A. Dearle, D. Balasubramaniam, J. Lewis, and R. Morrison, "A component-based model and language for wireless sensor network applications," in *2008 32nd Annual IEEE International Computer Software and Applications Conference*, pp. 1303–1308, Turku, Finland, July 2008.

[40] M. Botts and A. Robin, "OpenGIS sensor model language (SensorML) implementation specification," in *Open Geospatial Consortium (OGC, 07-000)*, Wayland, MA, USA, 2007.

[41] M. Diaz, D. Garrido, L. Llopis, B. Rubio, and J. M. Troya, "A component framework for wireless sensor and actor networks," in *2006 IEEE Conference on Emerging Technologies and Factory Automation*, pp. 300–307, Prague, Czech Republic, September 2006.

[42] 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 *2008 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pp. 515–522, San Francisco, CA, USA, June 2008.

[43] S. Bhatti, J. Carlson, H. Dai et al., "MANTIS OS: an embedded multithreaded operating system for wireless micro sensor platforms," *Mobile Networks and Applications*, vol. 10, no. 4, pp. 563–579, 2005.

[44] M. Vasilevski, N. Beilleau, H. Aboushady, and F. Pecheux, "Efficient and refined modeling of wireless sensor network nodes using SystemC-AMS," in *2008 Ph.D. Research in Microelectronics and Electronics*, pp. 81–84, Istanbul, Turkey, June 2008.

[45] A. Demaille, S. Peyronnet, and B. Sigoure, "Modeling of sensor networks using XRM," in *Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (isola 2006)*, pp. 271–276, Paphos, Cyprus, November 2006.

[46] V. A. Oleshchuk, "Ad-hoc sensor networks: modeling, specification and verification," in *Second IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2003. Proceedings*, pp. 76–79, Lviv, Ukraine, September 2003.

[47] J. M. T. Portocarrero, F. C. Delicato, P. F. Pires, and T. V. Batista, "Reference architecture for self-adaptive management in wireless sensor networks," in *Adaptive and Intelligent Systems. ICAIS 2014. Lecture Notes in Computer Science, vol 8779*, A. Bouchachia, Ed., Springer, Cham, 2014.

[48] A. Hac, *Wireless Sensor Network Designs*, John Wiley & Sons Ltd, 2003.

[49] K. Doddapaneni, E. Ever, O. Gemikonakli, I. Malavolta, L. Mostarda, and H. Muccini, "A model-driven engineering framework for architecting and analysing wireless sensor networks," in *2012 Third International Workshop on Software Engineering for Sensor Network Applications (SESENA)*, pp. 1–7, Zurich, Switzerland, June 2012.

[50] P. Boonma, Y. Somchit, and J. Natwichai, "A model-driven engineering platform for wireless sensor networks," in *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pp. 671–676, Compiegne, France, October 2013.

[51] I. Argyris, M. Mura, and M. Prevostini, "Using MARTE for designing power supply section of WSNs," in *M-BED 2010: Proceeding of the 1st Workshop on Model Based Engineering for Embedded Systems Design 2010*, Germany, 2010.

[52] F. Krichen, B. Hamid, B. Zalila, and M. Jmaiel, "Towards a Model-Based Approach for Reconfigurable Dre Systems," in *Software Architecture*, Springer, 2011.

[53] M. Ben Said, Y. H. Kacem, N. Ben Amor, M. Kerboeuf, and M. Abid, "Fine-grain adaptation for real time embedded systems using UML/MARTE profile," in *Proceedings of the 2013 Forum on specification and Design Languages (FDL)*, pp. 1–8, Paris, France, September 2013.

[54] A. R. da Silva, "Model-driven engineering: a survey supported by the unified conceptual model," *Computer Languages, Systems & Structures*, vol. 43, pp. 139–155, 2015.

[55] A. R. Paulon, A. A. Frohlich, L. B. Becker, and F. P. Basso, "Wireless sensor network UML profile to support model-driven development," in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pp. 227–232, Porto Alegre, Brazil, July 2014.

[56] A. Prakash, R. Chaparadza, and A. Starschenko, "A model-driven approach to design and verify autonomic

network behaviors," in *2011 IEEE GLOBECOM Workshops (GC Wkshps)*, pp. 701–706, Houston, TX, USA, Decemebr 2011.

[57] H. Wada, P. Boonma, J. Suzuki, and K. Oba, "Modeling and executing adaptive sensor network applications with the Matilda UML virtual machine," in *Proceedings of the 11th IASTED International Conference on Software Engineering and Applications*, pp. 216–225, ACTA Press, 2007.

[58] C. Vicente-Chicote, F. Losilla, B. Álvarez, A. Iborra, and P. Sánchez, "Applying MDE to the development of flexible and reusable wireless sensor networks," *International Journal of Cooperative Information Systems*, vol. 16, pp. 393–412, 2007.

[59] C. Vidal, C. Fernández-Sánchez, J. Díaz, and J. Pérez, "A model-driven engineering process for autonomic sensor-actuator networks," *International Journal of Distributed Sensor Networks*, vol. 11, no. 3, Article ID 684892, 2015.

[60] A. Sahu, E. B. Fernandez, M. Cardei, and M. Vanhilst, "A pattern for a sensor node," in *Proceedings of the 17th Conference on Pattern Languages of Programs - PLOP '10*, pp. 7:1–7:7, ACM Press, 2010.

[61] M. Cardei, E. B. Fernandez, A. Sahu, and I. Cardei, "A pattern for sensor network architectures," in *Proceedings of the 2Nd Asian Conference on Pattern Languages of Programs - AsianPLoP '11*, pp. 10:1–10:8, ACM Press, 2011.

[62] D. Gay, P. Levis, and D. Culler, "Software design patterns for TinyOS," *ACM Transactions on Embedded Computing Systems*, vol. 6, no. 4, 2007.

[63] J. K. Jacoub, R. Liscano, J. S. Bradbury, and J. Fisher, "UML modelling of design patterns for wireless sensor networks," in *Proceedings of the 2nd International Conference on Sensor Networks - Volume 1: SENSORNETS*, pp. 89–93, Barcelona, Spain, 2013.

[64] K. Tei, Y. Fukazawa, and S. Honiden, "Applying design patterns to wireless sensor network programming," in *2007 16th International Conference on Computer Communications and Networks*, pp. 1099–1104, Honolulu, HI, USA, August 2007.

[65] V. Gupta, E. Tovar, L. M. Pinho, J. Kim, K. Lakshmanan, and R.(. R.). Rajkumar, "sMapReduce: a programming pattern for wireless sensor networks," in *Proceeding of the 2nd workshop on Software engineering for sensor network applications - SESENA '11*, pp. 37–42, ACM Press, 2011.

[66] M. B. Said, Y. H. Kacem, M. Kerboeuf, N. B. Amor, and M. Abid, "Design patterns for self-adaptive RTE systems specification," *International Journal of Reconfigurable Computing*, vol. 2014, Article ID 536362, 21 pages, 2014.

[67] EARN project2015, http://www.crns.rnrt.tn/event/earn-2015.

[68] F. Ciccozzi and R. Spalazzese, "MDE4Iot: supporting the Internet of things with model-driven engineering," in *Intelligent Distributed Computing X. IDC 2016. Studies in Computational Intelligence, vol 678*, C. Badica, A. El Fallah Seghrouchni, A. Beynier, D. Camacho, C. Herpson, K. Hindriks, and P. Novais, Eds., pp. 67–76, Springer, Cham, 2016.

[69] F. Ciccozzi, I. Crnkovic, D. Di Ruscio, I. Malavolta, P. Pelliccione, and R. Spalazzese, "Model-driven engineering for mission-critical Iot systems," *IEEE Software*, vol. 34, no. 1, pp. 46–53, 2017.