Computer Networks 129 (2017) 548–571

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Model driven flexible design of a wireless body sensor network for health monitoring

Ahmed Harbouche[a], Noureddine Djedi[b], Mohammed Erradi[c], Jalel Ben-Othman[d,*], Abdellatif Kobbane[c]

[a] Department of Computer Science, LME Lab., Hassiba BenBouali University, Chlef, Algeria
[b] Department of Computer Science, Mohamed Khider University, LESIA Lab., Biskra, Algeria
[c] ENSIAS, Mohammed V University of Rabat, Morocco
[d] L2TI lab., University of Paris 13, Paris, France

A R T I C L E I N F O

Article history:
Received 12 January 2017
Revised 1 April 2017
Accepted 15 June 2017
Available online 28 June 2017

Keywords:
WBSN
Health monitoring
Sensors
Mobile data collector
Behavior derivation
Model-driven engineering
Verification

A B S T R A C T

The Wireless Body Sensor Network (WBSN) is a wireless network that is designed to allow communication among sensor nodes that are attached to a human body to monitor the body's vital parameters and environment. The design and development of such WBSN systems for health monitoring have received a large amount of attention recently, in research studies and in industry. This attention is mainly motivated by costly health care and by recent advances in the development of miniature health monitoring devices as well as emerging technologies, such as the Internet of Things (IoT), which contribute to the main challenges of 5G. The existence of an explicit approach to address the required software design and verification should be very beneficial for the construction and maintenance of such systems. This paper presents a preventive health care system that has a flexible design. The proposed system is based on an architecture that has heterogeneous nodes and provides both daily continuous monitoring as well as specific controls. We defined a model to describe the WBSN's global behavior. An important aspect of this work is that we propose a model-driven engineering (MDE) approach to address the derivation of each node's behavior in the WBSN from the WBSN global behavior. This approach allows developers to obtain a system design from the global specification of its requirement. To ensure the conformance of this design to its specification, the derived behaviors should be validated and verified before their deployment. In fact, formal methods are powerful tools for software engineers to verify the logical correctness of concurrent software at different levels of its life cycle. Model checking is one of the most powerful formal methods for verifying the logical correctness of such concurrent systems. In this work, we make use of a model checking approach that is based on a model transformation to validate the automatically derived behavior of a WBSN for health monitoring. This model-driven approach will check whether the derived system behaves correctly according to its global specification, while the objective is to increase the system's performance and QoS. This approach allows the developer to reason about a model of the global system rather than about the system itself.

1. Introduction

Health care monitoring systems can monitor a patient's health status. The increasing demand for solutions in the health care industry is motivated by the high cost of health care, the growing and aging population and the increase in chronic disease patients around the world. Because of these factors, traditional health care cannot provide the desired scalability. Therefore, there is a need for high performance, low cost and effective health care solutions.

A Wireless Sensor Network (WSN) consists of a number of wireless sensors in addition to a communication infrastructure. The sensors measure and record the conditions at diverse locations. Commonly, such measures are environmental (e.g., air quality, temperature and humidity) and human vital functions (e.g., heart and brain signals). A WSN is capable of sensing, processing and communicating. The collected data are transmitted to a base station or a computer, to help it to analyze and react according to the conditions observed in an environment. Wireless Sensor Networks (WSNs) have paved the way for progress in various aspects of

* Corresponding author.
E-mail addresses: Ah.harbouche@gmail.com (A. Harbouche), djedi_nour@yahoo.fr (N. Djedi), mohamed.erradi@gmail.com (M. Erradi), jbo@univ-paris13.fr, jalel.ben-othman@univ-paris13.fr (J. Ben-Othman), abdellatif.kobbane@um5.ac.ma (A. Kobbane).

http://dx.doi.org/10.1016/j.comnet.2017.06.014
1389-1286/© 2017 Elsevier B.V. All rights reserved.

monitoring. Many WSN-based monitoring applications have been developed in different fields of application to gather data on the monitored environment. These applications include, but are not limited to, military applications [1], global climate monitoring applications [2], applications in underwater networks [3], and applications in structural health monitoring [4].

Health care monitoring systems using Wireless Sensor Networks have been designed to provide vital body function surveillance. Wireless body sensor networks (WBSN) for health monitoring enable constant monitoring of the health conditions of people. They can comprise various types of on-body and ambient sensors that are deployed around the patient. Commonly, the conditions monitored by the WBSN systems include diabetes, cardiac arrhythmia, sleep apnea, asthma and physiological signals or parameters (such as blood pressure, body temperature, heart activity, respiration rate, oxygen saturation, and brain activity). These parameters can be monitored remotely by doctors and caregivers without affecting the patients' activities. In fact, these systems have significantly minimized human errors, allowing better understanding of the origins of diseases.

A variety of Wireless Body Sensor Networks (WBSNs) has been developed in recent years. They aim to monitor a specific disease or a set of physiological signals of the patients at their homes [5–7], to perform daily physical activity recognition and heart monitoring [8]. These systems must satisfy certain criteria while operating under significant hardware resource limitations [9,10]:

1. The security and the privacy of the measured personal medical data must be guaranteed by the system, and
2. The energy consumption must be minimized to increase the system's operational lifetime.

The design of WBSN systems is a very complex task. Designers of such systems should account for these limitations when building increasingly reliable systems. However, a majority of studies in this area are focused on implementation issues, and they rarely address a methodology for the construction and maintenance of such systems. Most of these systems are developed by selecting the most appropriate target platform first and, then, its specific operating system (e.g., TinyOS) and programming language (e.g., NesC). The existence of an explicit architecture should be very beneficial for the construction and maintenance of large WBSN systems. Such architectures should provide techniques and approaches to address software design and verification.

The work presented in this paper contributes to the design and validation of WBSN-based preventive health care applications. We first propose an architecture that has heterogeneous nodes to build a preventive health monitoring system with a flexible design. The proposed health monitoring system is designed to perform daily health monitoring as well as specific controls for patients in a hospital. It is a WBSN architecture based on a mobile data collector. The mobile collector is used to save the resources (energy consumption) of the sensors. The sensor nodes can be awoken only when they expect the data collector to be present.

Second, a model-driven engineering (MDE) approach is described. This approach aims to derive the WBSN component behavior (sensors, data collector, and so on) from the WBSN global requirements. The requirements level and its transformation to the design level are the first step to generating a quality design of a complex system. The requirements specifications describe the system's global behavior. Usually, the global behavior of a distributed system is achieved by a set of collaborative components. Such global behavior can be decomposed into partial behaviors that are performed by the system's components. A manual decomposition of the behavior can lead to design errors. Therefore, an automatic transformation approach is needed to derive the behaviors of the components from the system's global requirements. The use of a software engineering approach to support such a derivation process can lead to a highly platform-independent design. Additionally, it promotes the reuse of software modules with which a derived behavior model can be reconsidered on different platforms.

Software designers know from experience that, most likely, any software system does not work well after the first successful compilation, nor the second or third. Sometimes, it takes a while to discover how seemingly correct software can fail in subtle ways. Small defaults can hide for years and appear at the most inconvenient moment when they are least expected. However, there are cases in which such defaults are not acceptable, such as is e-health applications or critical economical applications; defects in software can lead to a loss of life or can cause significant economic damage. Given Wireless Body Sensor Networks (WBSNs) that enable constant monitoring of the health conditions of people, the most important challenge is to find ways to make these applications more reliable. The verification of such a system involves design validation. This verification checks whether a system design satisfies the system's requirements. (If it does not, then it is desirable to find out early in the design process!) These tasks, system verification and design validation, can be accomplished using the techniques of simulation and testing.

The time-honored techniques of simulation and testing are very useful debugging tools for the early stages of system design and verification. They involve checking the system's behavior on a large set of expected inputs. However, while a system is being refined, the remaining bugs become fewer and more difficult to uncover. A major gap in the process of using simulation and/or testing for verification and validation is that there is no way to tell when all of the bugs in the system have been found. In other words, testing and simulation can be used to demonstrate the presence of bugs but not the absence of bugs. Quite simply, it has been proven that testing and simulation cannot be used to guarantee an ultra-high level of reliability within any realistic period of time [11]. For some systems, this limitation is an acceptable risk. For those systems, it is sufficient to reduce the bug level to be below a certain measurable tolerance. For other systems, such as life-support systems (e.g., WBSN), where the reliability is key because failure is potentially catastrophic and not tolerated, an absolute assurance is required that the system follows its specification via an examination of all possible behaviors, including those that are unexpected or unintended. This assurance can be accomplished reliably using formal methods.

Formal methods are powerful tools for software engineers, and are used to verify the logical correctness of concurrent software and to make the software work correctly. While there is a range of different techniques for formal verification, model checking is one of the most powerful formal methods and is especially well-suited for the automated verification of finite-state systems. Model checking requires sophisticated algorithms that are based upon automata theory and logic to check the models, but not programs [12]. The models are high-level descriptions of a system. These models represent faithfully the system, while remaining sufficiently concise to enable its correctness to be checked. Model checking is the preferable verification method where applicable for two reasons. First, it is significantly more efficient and cost-effective to perform verification as early as possible in the system design process, thereby avoiding the possibility of discovering an error that requires a redesign in the completed system. Second, it is simpler and easier to reason about a model of the system rather than about the system itself because the model includes only the relevant features of the larger system and because the model is easier to build and redesign as necessary [13].

The Model-Driven Engineering (MDE) approach provides models that define the requirements, the design, the deployment and the implementation of a given system. Each model describes the system at a given level of abstraction. Therefore, model transformations are needed between the different levels of abstraction to obtain final application code from the system requirements model. Each level of abstraction is described using a specific meta-model. Thus, applying an MDE approach requires the definition of the appropriate meta-models and the corresponding model transformations. This work focuses on the early development phases of a WBSN system, especially in the case of obtaining a system design from a global requirement specification. It proposes an MDE approach to derive the behavior of the WBSN components by transforming the WBSN global requirements model to the design model. The WBSN global requirements model describes the functional behavior of a given WBSN in an abstract way. The design model represents the local behavior of each component in the WBSN. These derived system components execute in a distributed environment. The concurrency aspects make these applications difficult to build and apply correctly. The most important challenge is to find ways to make these derived behaviors more reliable. This goal involves checking whether the derived system in question behaves as it was designed to behave. All of these reasons motivate our work for the use of an MDE approach that facilitates the derivation and verification of the WBSN system components behavior.

This paper presents an MDE approach to verify and validate the derivation process. It involves checking whether the collaboration of the derived behaviors satisfies the initial system's global specification. If it does not, then it is desirable to find out early in the design process. The MDE approach describes a process based on model-driven checking for verifying and analyzing the logical consistency of the derived WBSN system. The challenge for the designers is to develop models that faithfully represent the system, while in this case, these models are automatically derived from the system's global behavior. There is no time to spend during the design of these models to ensure that they sufficiently describe the system. The MDE verification process is mainly based on automatic model transformations of the derived models to specific models that are needed by a model checker to accomplish the verification. Afterward, model checking is used to analyze the derived behavior and therefore checks whether this behavior conforms to the global system specification.

We carefully consider each step in the end-to-end system architecture design, the processes of model-driven derivation and then verification (checking), while giving the big picture before the lower-level details of each step. Therefore, the structure of this paper is as follows. Section 2 reviews the related studies. Section 3 describes the system architecture design. Section 4 gives an overview of the WBSN derivation and the verification processes. Next, we present the derivation process and how the design models are obtained from the WBSN global specification in Section 5. Section 6 discusses the different steps of the verification process while specifying the behavior properties in linear temporal logic. The code generation process and the implementation are presented in Sections 7 and 8. Finally, we conclude with a discussion in Section 9.

2. Related work

The studies presented in [14,15] propose the use of the data-MULE system. The studies in [16,17] use the message ferrying scheme. In these two approaches, the data-MULE and the message ferries gather data from the sensor nodes while moving around the network area. These approaches are designed only to provide the service of message relaying in networks. At the same time, the mobile sink can consume the collected data for its own purposes or make the data available to the remote users.

In [18,19], one or multiple mobile sinks move throughout the WSN to gather data from static sensor nodes that are deployed in the network area. In the proposed architecture, we have made use of a mobile data collector to gather the data in the WBSN system. This solution was extended by a human presence (the nurse). A human presence can add value to this architecture by analyzing the collected data and making some urgent decisions. We also provide an approach to designing, deriving and verifying the system behavior. This approach derives the behavior of each WBSN component from the WBSN global behavior for various functioning scenarios of the hospital. The verification validates the derived system's design by checking whether it satisfies the system's requirements and making the system more reliable. It is very desirable to perform verification as early as possible in the system design process, thereby avoiding the possible discovery of an error in a completed system that would require a redesign.

WSNs are very efficient in supporting various day-to-day applications. WSN-based technologies have revolutionized home and elderly health care applications. Various publications in this research area have been proposed to monitor human vital body parameters at home, for elderly patients or in a health care center. These proposed methods and their implementations have facilitated health care systems. For continuous and real-time monitoring of health, the authors of [20] have developed a smart shirt that measures the ECG (Electro Cardio Gram) and acceleration signals. The shirt is composed of conductive fabrics to obtain the body's signal from electrodes and consists of sensors for online health data monitoring. The observed and measured data are transmitted in an adhoc network for remote monitoring. An Android device is used to analyze the ECG signals from a mobile monitoring terminal, as mentioned in [21]. In [22–25], applications of Wireless Sensor Networks for health-care monitoring at home are presented. They provide continuous health monitoring of patient in their home without restricting their activities and body movements. These studies are based on the use of a static data collector to gather the data. They are focused only on the implementation issues. In contrast to such work, our proposal was designed to allow the health monitoring of patients at a hospital. It also uses a mobile data collector and provides a development approach to derive and verify the behavior for the WBSN components, to make the system more reliable.

In [26], the authors proposed a Wireless Sensor Network personal health monitoring system for the collection and dissemination of medical sensor data. This system allows data storage correlation and dissemination as well as timely alerts, when the parameters are breached. The implementation and analysis of a WSN-based e-Health application has been described in [27]. The authors of [28] illustrate the design and implementation of an e-health monitoring networked system. The architecture for this system is based on smart devices and wireless sensor networks for real-time analysis of various parameters of patients. Although these studies present e-health monitoring networked systems, they are focused on the implementation issues. They do not address the design and maintenance of such systems. In addition to these studies, we provide a model-based approach to derive the WBSN behavior from its global specification and to verify whether the derived system behaves correctly according to its global specification, while the objective is to increase the system performance and QoS.

The studies in [29–32] present an MDE approach to WSN application development. They addressed only the design and development of static WSNs. These studies are focused only on model transformations and their results and do not provide the means to validate the transformations. The proposed model-driven approach derives the components' behaviors. It increases the system dynamicity by the automatic derivation of the behavior for the different functioning scenarios of the hospital. In addition to the behavior derivation, we propose a model-based method to validate and verify the results of the model's transformation. This verification approach is needed to check whether the obtained system

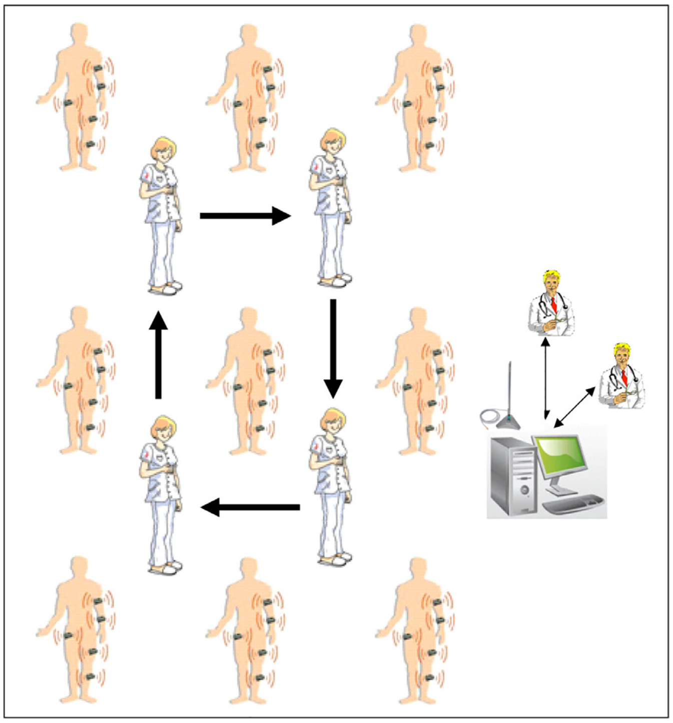


Fig. 1. WBSN floor architecture.

after transformation behaves as it was expected to behave. Model checking is one of the most powerful formal methods to be used in the verification of concurrent and distributed systems.

However, research thus far in system verification focuses mostly on verification and techniques that use an automaton to achieve the verification [13,33–37]. These investigators were not interested in the model creation, which is an important step in the verification process. In this proposed approach, the software designer spends no time on the system design. The design model is automatically obtained from the requirements specification. The requirements specification describes the WBSN global behavior at an abstract level. In our approach, we take advantage of the derivation process, which transforms the WBSN global behavior into a set of WBSN component behaviors. The whole system is then modeled by a set of communicating models. Thus, only such a sequence of models must be checked. This arrangement leads to reducing the state space of the WBSN to be checked by decomposing the global system model into multiple concurrent processes that provide the exact same behavior in the global WBSN. Therefore, it is better to model the whole WBSN with a set of communicating models that represent its subsystems than a single model.

3. WBSN architecture design

A hospital is composed of many floors. Each floor consists of a number of patient rooms. It is structured as shown in Fig. 1. The sensors are embedded in and around the environment of the patient and in his body as well. They can be classified into two types: medical sensors and environmental sensors. The medical sensors monitor the vital parameters of the patient, whereas the environmental sensors monitor the parameters of the room, including the room temperature, oxygen levels and beyond. A caregiver equipped with a wireless node, for example, a personal digital assistant (PDA), a smart phone or a pocket PC gathers the data measured by the sensors. This mobile data collector can either consume the collected data for its own purpose (displays the physiological information on a user interface) or transmit it to a medical center. The doctors consult the medical center to display the patient status and to make appropriate care decisions.

There are four basic users in this architecture: patients, nurses (caregivers), doctors, and technicians.

1. Patients are equipped with sensors. These sensors are medical and environmental sensors. The medical sensors are attached to the patient to measure vital body parameters, whereas the environmental sensors are embedded in and around various locations in the patient's room. These sensors produce raw values of data that are transmitted to the data collector by wireless links. Taken together, these values present the real-time situation of the patient at all times. The sensor nodes with the data collector form a body sensor network (BSN) in the basic configuration of a star topology.
2. Nurses or caregivers are equipped with a mobile data collector. This mobile data collector gathers the measured data from the sensors and converts it into meaningful metadata. The sensor data contain only values of the measured parameters, with no patient information or measurement time. The mobile data collector adds values, such as a unique identifier, the type of parameter being monitored, the time and unit of measurement, to make the information meaningful. Then, it transmits the metadata to the medical center. The communication between the BSN and the medical center can be accomplished using WLAN, GSM, or other systems, which can offer wide coverage. The use of a smart device to collect the data can also be used for local processing of data to analyze the health condition of the patient.
3. Doctors consult the medical center to display the patient status. They prescribe medicines or provide suggestions to the patient, which correlate with the values of the parameters that are being received from the sensors. Based on the previously recorded data of the patient, the doctors offer advice by comparing the previous trends with the current trend in the sensor data.
4. Technicians consult the medical center to detect any sensor failures. They will respond to an alarm or malfunction of a sensor.

We have categorized the architecture of our system into three layers based on the functionality of the components being used. The figure (Fig. 2) illustrates the three layers used in our architecture:

1. The Perception Layer: The first layer at the bottom of the hierarchy consists of medical and environmental sensors that collect real-time data. The medical sensors are used to analyze the health of the patient by measuring various vital body parameters. These sensors include the heart rate monitor, blood pressure sensor, ECG module, and thermometer. The environmental sensors monitor the surroundings of the patient and ensure that the patient is in healthy condition. These sensors include gas detection sensors, temperature sensors, and so on. The data accumulated by the sensors are collected by the data collector, which converts the data into meaningful information. The data collector attaches some information to the measures to distinguish between the raw values; for example, it adds a unique patient identifier to distinguish which report is for which patient. Then, it relays the metadata to the medical center. Hence, the data collector acts as a gateway for the system between layer 1 and layer 2 of the architecture.
2. The Processing Layer (Medical center): This layer stores the medical history of the patient as well as the current records of the monitored parameters. These records store (i) medical reports, (ii) medical prescriptions and (iii) medicines to which the patient is allergic. This storage plays a central role in emergency responses and hospital monitoring system because it allows to compare data collected from the sensors with the stored

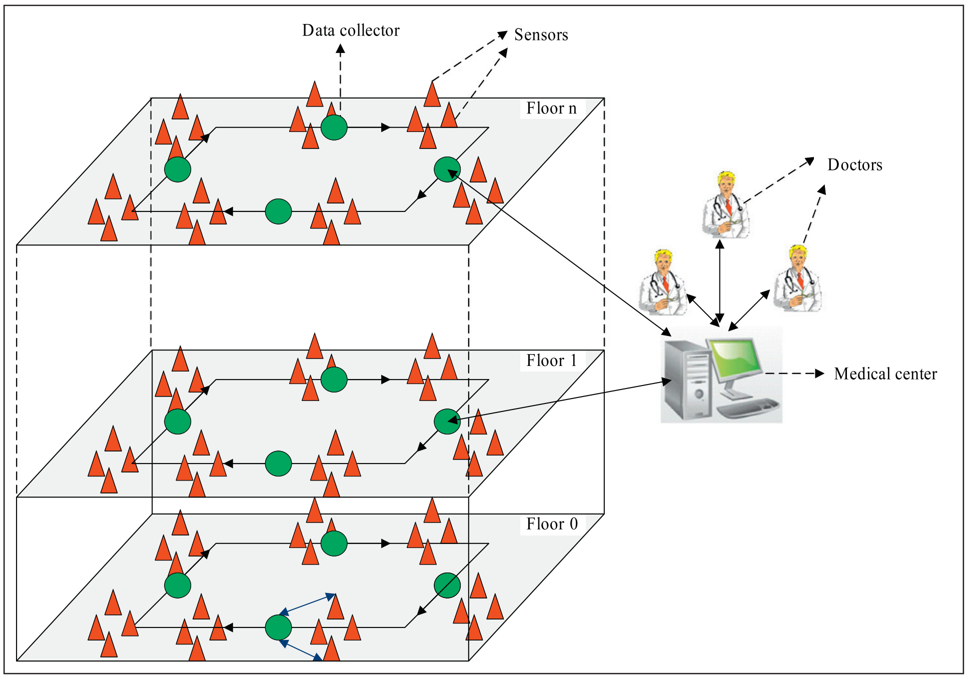


Fig. 2. WBSN for health monitoring architecture.

thresholds for the parametric values. The data are then relayed to this layer to be used by the doctors and service applications (third layer of the architecture).

3. The e-health Services (users and services): The third layer of the system is a layer that offers services for the use of the monitored data provided by the processing layer. It consists of nurses, doctors and technicians. This layer offers several services to the patient. It involves prescribing medicines or providing suggestions to the patient by the doctors. These medical prescriptions are made according to the values of the parameters that are being received from the sensors. There is also an immediate response of the doctors and nurses in the event of an emergency and technicians in the case of a sensor failure. The staff monitors these alarms for each of the patients and supplies the patient with the required medication to address the situation.

The proposed architecture minimizes the measurement errors by collecting data from a nurse equipped with a data collector. The presence of humans allows analysis of the measured data, notes on the patient's status and performing maintenance on the network if necessary (e.g., to correct a wrong position of a sensor). Assistance is directly requested to other caregivers when an anomaly is observed. On the other hand, in ordinary hospital functioning, the sensors are awakened only in the presence of the mobile data collector, which minimizes the energy consumption in the WBSN system. In this case, the nurse gathers data from sensors once a day and transmits the data to the medical center. Doctors can view the data to observe a patient's status. In other cases, the sensors can be awoken periodically to make measurements, store them in their private memories and send them to the collector. Thus, the ordinary functioning of the system can be enriched by new constraints, such as the treatment of alarms, critical cases and emergencies.

The WBSN system can give rise to several approaches to operating the hospital (ordinary hospital functioning, functioning with periodic measurements, functioning with alarm processing, functioning with handling emergencies, and so on). Each scenario describes a WBSN global behavior. Such global behavior can be decomposed into partial behaviors that are performed by the different system components or roles (Sensor, Data collector and Doctor). A manual decomposition of the behavior can lead to design errors. Therefore, an automatic transformation approach is needed to derive the behavior of these components from the system's global behavior.

The distributed nature of WBSNs and the different scenarios that they could have makes it difficult to correctly design and develop WBSN systems. Many proposals for WBSN systems for health care at home have been developed. They are mainly focused on implementation issues, and they rarely address the development process. The development of such a system with various scenarios is an increasingly complex task. The use of a software engineering approach to support the design and verification of WBSN systems could lead to a highly platform-independent design and promotes the reuse of software on various platforms. The Model-driven Engineering (MDE) approach can contribute to solving this problem by allowing designers to build and validate WBSN applications.

This paper presents an MDE approach to derive the behavior of the WBSN system components by transforming the WBSN global requirements model to the behavior model of each component. The WBSN global requirements model describes the functional behavior of a system in an abstract way. This approach will allow developers to build more flexible and reusable designs. A formal model checking method is then used to verify that the derived behaviors satisfy the system specification.

4. The model-driven derivation and verification approach overview

The Wireless Body Sensors Network systems development is an increasingly complex task. The WBSN systems global behavior is not achieved by a single component but by a set of collaborative components. The development of such complex systems requires the decomposition of the WBSN global behavior into partial behaviors that are performed by the different system components. The construction and maintenance of such systems require the existence of explicitly stated development methodologies. Such methodologies should provide techniques and approaches that ad-

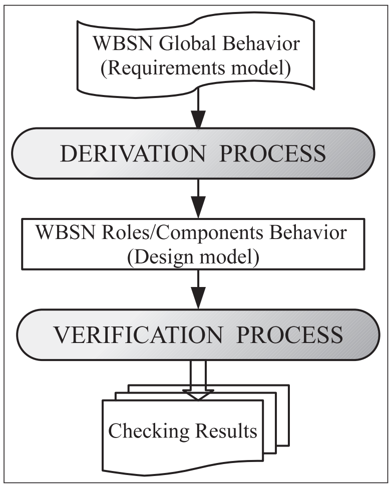

WBSN Global Behavior (Requirements model)
DERIVATION PROCESS
WBSN Roles/Components Behavior (Design model)
VERIFICATION PROCESS
Checking Results

Fig. 3. The model driven derivation and verification approach.

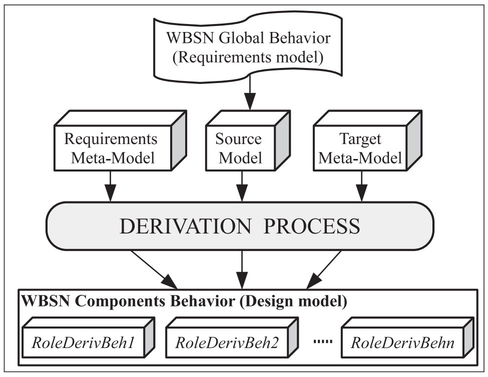


Fig. 4. WBSN derivation process.

dress the automatic derivation of the system components' behaviors. The use of a software engineering approach to support such a derivation process could lead to highly platform-independent designs. Furthermore, the most important challenge is to find ways to ensure that this derived software runs without any errors. These derived WBSN components execute in a distributed environment. The concurrency aspects make these applications difficult to get right. All of these reasons motivate our work for the use of an MDE approach first to derive the behavior of each system component and then to validate this derivation. The validation achieves the formal verification of the system components' behaviors. Fig. 3 shows an overview of the model-driven derivation and verification approach.

The behavior of the system components is obtained by transforming the system global requirements model, the WBSN global behavior, to the design model, the behavior of the system components, in which each component is described by its role in the system. The WBSN global requirements model describes the functional behavior of a given system in an abstract way. The design model represents the local behavior of each system role. The verification process can ensure that the collaboration of the derived behaviors should satisfy the initial requirements model.

Model checking is one of the most powerful formal methods used in the verification of concurrent and distributed systems. Therefore, the first step in model checking is to create a system model to be verified. The challenge is to write a model that describes only the relevant features of the larger system, and a further challenge is that the model is sufficiently detailed to represent the system faithfully. However, the model is created, this model can cause the creation of an astronomical number of states during model checking (the state explosion problem), which cannot be supported by the computer memory. Fortunately, real computations cannot have an infinite number of distinct states, for the simple reason that the computer memory is finite. The state explosion problem leads to halting the model checking. This problem can be mitigated during the modeling phase by reducing the state space to be as small as possible. Reducing the state space is achieved by modeling only the relevant aspects of the system [34].

In the proposed approach, the designer spends no time on the system design. The WBSN design models are automatically obtained from the requirements specification. The requirements specification describes the WBSN's global behavior. To represent the system's global behavior, we suggest the use of UML collaborations as the main activities in UML activity diagrams for the construction of the requirements models. Unified Modeling Language (UML) is a modeling language which provides system architects, software engineers, and software developers with tools for analysis, design, and implementation of software based systems as well as for modeling business and similar processes [38]. A given collaboration describes the structure of the collaborating components. Each component performs specific processing, and all of the system components realize the desired functionality (WBSN global behavior). The collaborations are very appropriate to model the requirements because they provide a structural framework for such requirements, which embody both the behavior of each component and the interactions between the components. This arrangement allows describing a given WBSN global requirement at an abstract level.

The reduction in the state space during the modeling step can sometimes be insufficient and cause the state explosion problem. To resolve this problem, it is better to use another technique to reduce the state space. This technique consists of representing concurrent system components independently to take maximum advantage of the reduction in the system model's size. This reduction technique consists of decomposing the WBSN global model into multiple concurrent models that have the exact same behavior as the global system. Therefore, it is better to model the whole system with a set of models of different independent and communicating subsystems than a single model. In our approach, we take advantage of the proposed derivation process, which transforms the requirements model into a set of WBSN component behaviors. Synchronization messages are included in the derived models to ensure the system's synchronization. The whole system is then modeled by a set of communicating models. Thus, only one such sequence of models must be checked.

The final stage of the approach is to proceed with the code generation for a specific platform once the results of the verification show that the system is reliable.

5. The WBSN derivation process overview

We have defined a model-driven approach to derive the behavior of the WBSN components. This MDE approach derives the behavior of the WBSN components by transforming the WBSN global requirements model to the WBSN design model. The appropriate meta-models have been defined at each level of abstraction with the corresponding model transformations. This approach allows designers to describe the WBSN global behavior using the requirements model. The requirements model is then automatically transformed to the behavior of the WBSN components. A set of rules is defined to manage the model transformations during the derivation process. The derivation process (Fig. 4) is achieved by the following:

1. The definition of the requirements meta-model (source): This meta-model describes the WBSN global behavior at a high level

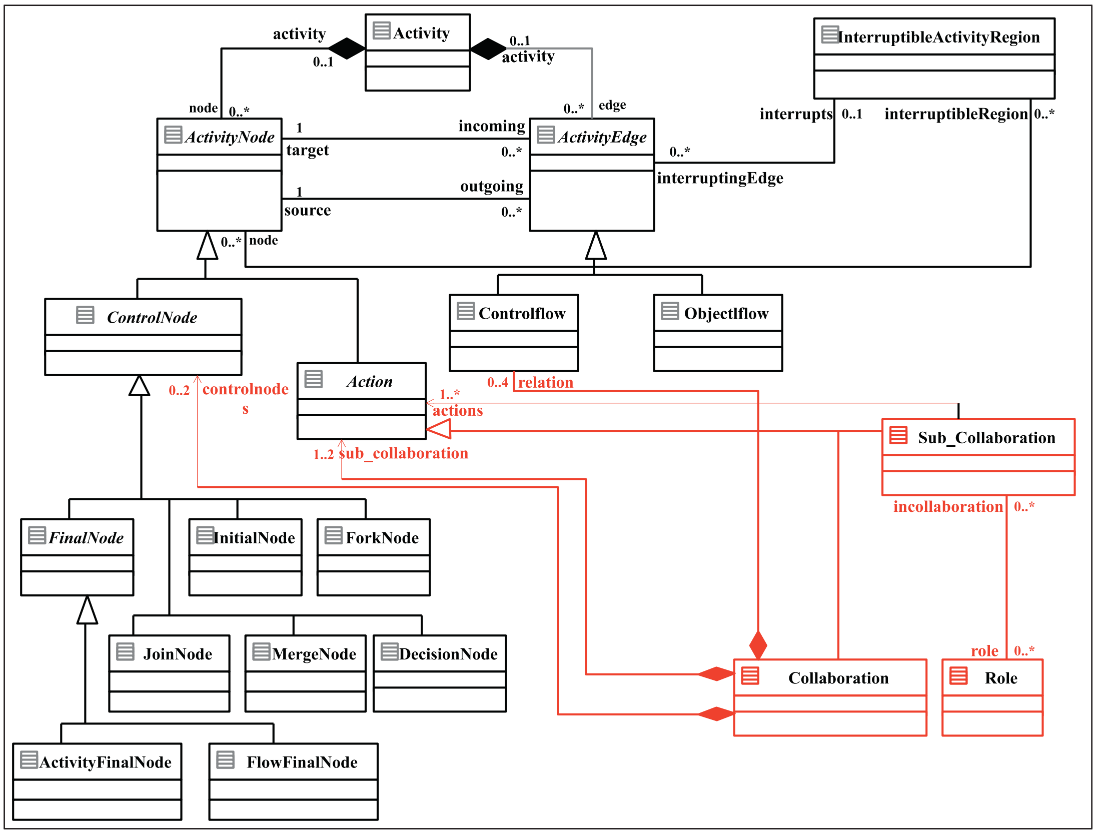


Fig. 5. The requirements meta-model.

of abstraction. It is defined by a UML activity diagram extended with collaborations.

2. The description of the design meta-model (target): This meta-model describes the local behavior of each system role or component. It is the UML state machine meta-model.
3. The definition of the model-to-model (M2M) transformation rules: These rules map the concepts from the requirements meta-model to those of the design meta-model. These transformation rules will govern the derivation process.

5.1. The basic meta-models

The meta-models used in the derivation process are the requirements meta-model and the design meta-model. The suggested requirements meta-model is defined by a UML activity diagram that is extended with collaborations. The design meta-model is the UML state machine meta-model [38]. The definition of the requirements meta-model describes the WBSN global behavior at a high level of abstraction. The requirements meta-model provides the needed concepts to describe the system without considering any design concepts or information about the target platform. At this stage, the models must give a clear picture of the WBSN global behavior and the collaborations between the components.

The proposed requirements meta-model (Fig. 5) provides the appropriate classes and their associations that are used to describe the behavior of a collaborative WBSN system. Such a requirements meta-model is considered to be the source meta-model of the derivation process. The WBSN global behavior is then described by a UML activity diagram that is extended with collaborations concepts. The collaborations are used as the basic activities in the activity diagram meta-model. They involve multiple collaborative roles. Each Collaboration is composed of one or two collaborations (sub-collaborations). A Sub-collaboration can consist of some actions that are accomplished by the collaborating roles. The roles represent the different system components. The UML activity diagrams are well suited to represent the choreography of collaborations and sub-collaborations when describing the system's global behavior. They can express sequential behavior, alternative behavior, competing behavior (parallel composition), or repetitive behavior, as well as interruptions. The WBSN global behavior model is defined as an activity diagram in which the actions are structured collaborations.

In the WBSN system for health monitoring, each functioning scenario is expressed with collaborations among the system components (sensor, data collector, doctor, medical center). The following collaborations are distinguished in the ordinary functioning scenario, where each collaboration describes the interactions among the system's components.

- Sensor activation by the data collector,
- Measure of the medical and environmental parameters,
- Data collection from sensors,
- Transmission of collected data to the medical center,
- The doctors consult the medical center to display the patient's status,
- The doctors provide an adequate decision and suggestions to address the patient situation,
- Collaboration between doctors and caregivers to respond to an emergency.

The behavior of the various components of a given WBSN that results from the derivation process are modeled using UML state machines. The UML state machine meta-model was selected as the design meta-model. It is a platform-independent modeling language. Thus, the generated models provide a clear description of the WBSN component behaviors without any information about the implementation platform.

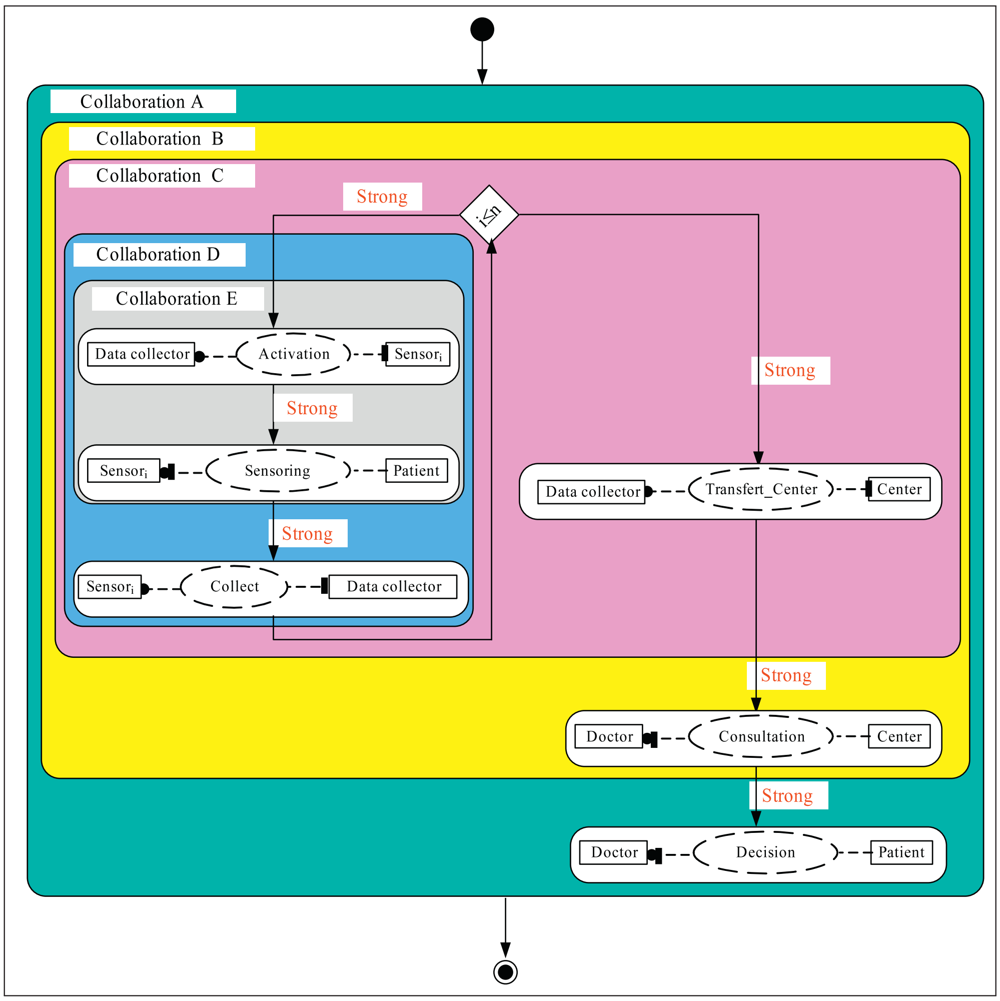


Fig. 6. WBSN system global behavior.

5.2. The derivation process

The designer specifies, in the first step of the derivation process, the global requirements model that describes the WBSN's global behavior. The Figure (Fig. 6) represents the WBSN global behavior for ordinary hospital functioning. This behavior is modeled by an activity diagram whose core activities are collaborations and sub-collaborations.

The second step of the derivation process achieves the model-to-model transformation. The model-to-model transformation (M2M) transforms automatically the WBSN requirements model into design models. We have defined a set of transformation rules to govern this transformation. The derivation process includes synchronization messages in the WBSN derived behaviors to ensure coordination between the WBSN components.

A derivation algorithm has been designed to achieve the derivation process. It derives the behavior for a given component in the WBSN. This algorithm applies the appropriate transformation rule for each concept described in the WBSN global behavior model (Mg) and generates the corresponding concept in the target design model (Md). This design model conforms to the UML state machines' meta-model (MMd). Each collaboration concept is transformed into a composite state with the same name as the transformed collaboration. This composite state consists of a UML state machine that is the result of the transformation of the different concepts embedded within the corresponding collaboration. During the transformation of a sub-collaboration, the algorithm transforms the component actions. Coordination messages are included in the different states to ensure synchronization between the derived behaviors. Transitions and control states are generated to interconnect the generated states.

The transformation rules are defined according to the different cases of choreography expressed in the UML activity diagrams. We have identified sequential, choice composition, parallel composition and repetition behaviors. For each case, we have designed a set of M2M transformation rules to achieve the WBSN derivation.

To define the transformation rules, we must to distinguish between the roles (components) in a collaboration:

Definition 1. A starting role is a role that accomplishes an initial action in a collaboration or in one of its sub-collaborations [39].

Definition 2. A terminating role is a role that accomplishes a final action in a collaboration or in one of its sub-collaborations [39].

The sets of starting (SR), terminating (TR) and participating roles (PR) are calculated as in [39]. These sets are determined for a given collaboration according to the type of sequencing operator used in the WBSN requirements model. Two types of sequencing are distinguished [40,41]: Strong and Weak sequencing.

Definition 3. Strong sequencing implies that all sub-activities of an activity A1 are finished before an activity A2 can begin.

Definition 4. Weak sequencing specifies that an activity A2 will be executed after another activity A1. Weak sequencing provides only

Algorithm 1 The derivation algorithm.

```
Init r = component_name;
for all Concept= collaboration C in Mg do
    Generate a composite State S in Md conform to MMd;
    S.Name= C.Name; Generate (Initial State) in S;
    if type_collaboration C = sub-collaboration then
        for each action a in C do
            Transform (a, a')
        end for
    else
        for each C' in C do
            Transform(C', S', Messages) ;
        end for
    end if
    if Messages ≠ Nil then
        Integrate the coordination messages in S
    end if
    Generate the transitions to connect the states in S;
    Generate (Final State) in S;
end for
for all Concept= Controlflow or ControlNode in Mg do
    Transform(controlflow, Transition);
    Transform(controlNode, ControlState);
end for
```

a local order of activities for each system component and does not imply a global order.

Two coordination messages are used in the derivation process to ensure synchronization among the WBSN components. We use the same type of coordination messages as introduced in [39]. Each coordinating message contains the following parameters: (a) Source component (Sr), (b) Destination component (Dr), and (c) name of the state it belongs to (St).

1. Flow message for coordinating strong sequencing, named Flowm(Sr, Dr, St).
2. Choice indication message for propagating the choice to a component that does not participate in the selected alternative in the choice composition structure, named Choicem(Sr, Dr, St).

Next, we present the rules needed to derive the behavior of the different components that are involved in a WBSN global requirements specification for the cases of strong and weak sequencing. Each rule is provided to perform a model-to-model transformation for a participating component r in a collaboration C_i. The transformation of the collaboration concept triggers the transformation of its sub-collaborations. This property requires the definition of recursive transformation rules.

Case 1: Strong sequencing between two collaborations.

The strong sequencing consists of two collaborations (C_1 and C_2) linked by a control flow labeled Strong. Each collaboration can be composed of other collaborations or sub-collaborations.

Rule 1: The source collaboration C_1 is transformed into the composite state S_1, for a terminating role r in C_1. This state will hold the actions performed by the role r after the transformation and include the actions of sending the coordination messages *Flowm*. The coordination message *Flowm* is sent by the role r to the starting roles of the target collaboration C_2 (except not to itself if it belongs to the set of roles).

If r ∈ TR(C_1) then Transform(C_1, S_1, Send(Flowm(r, r', S_2))) ∀ r'∈ (SR(C_2) - r);

Rule 2: The target collaboration C_2 is transformed into the composite state S_2, for a starting role r in C_2. This state consists of the actions of receiving coordination messages *Flowm* from the terminating roles of C_1 (except from itself) and includes the actions performed by r.

If r ∈ SR(C_2) then Transform(C_2, S_2, Receive (Flowm(r',r,S_2))) ∀r' ∈ (TR(C_1) - r);

Rule 3: The collaboration concept is transformed into a composite state, for a participating role r, not terminating in the source collaboration or not starting in the target collaboration. This state holds the actions performed by the role r.

If r∈ (PR(C_1)-TR(C_1)) or r∈ (PR(C_2)-SR(C_2)) then Transform(C_i, S_i) ∀ i=1,2;

Rule 4: The ControlFlow concept is transformed into a transition, for a participating role r in the source and the target collaborations. This transition connects the states S_1 and S_2 obtained from the transformation of C_1 and C_2.

If r∈ PR(C_1) and r∈ PR(C_2) then Transform (Controlflow, Transition);

Case 2: Weak sequencing between two collaborations.

The weak sequencing consists of two collaborations (C_1 and C_2) linked by a control flow labeled Weak. Each collaboration can be composed of other collaborations or sub-collaborations. To transform the concept ControlFlow, Rule 4 is applied.

Rule 5: The collaboration C_i is transformed into a composite state S_i, for a participating role r in the collaboration C_i. This state holds the actions performed by r.

If r∈ PR(C_i) then Transform(C_i, S_i) ∀ i=1,2;

The result of the derivation process is a UML state machine for each role that represents a component's behavior. Figs. 7 and 8 describe the data collector and the sensor behaviors.

6. The verification process

The derived behaviors of the WBSN components operate in a distributed environment and collaborate to achieve the system's global behavior. This cooperation requires a formal method to ensure that the collaboration of these behaviors should satisfy the initial requirements model. Model checking is the formal process through which a desired behavioral property (the specification) is verified to hold for a given system (the model). This verification should examine all of the behaviors that cause the system transitions between the reachable system states. Once the system model and specification have been determined, the model checking step is accomplished. The model checking returns an error when the specification is found to not hold in all system executions. This error specifies that a faulty system behavior is detected. A counterexample is produced, which consists of a trace of the model from a start state to an error state in which the specification is violated. This trace of the model provides the necessary diagnostic feedback.

As specified previously, the most important aspect for the software designers is to develop a model that faithfully represents the system in sufficient detail to enable it to faithfully reproduce reality, but at a sufficiently high level of abstraction that it is easy to understand and tractable to verify that the system functions properly in its environment. In the suggested approach, these models are automatically derived from the WBSN global behavior. There is no time to spend on the design of these models and to ensure that they sufficiently describe the system. This automatic decomposition reduces the system complexity for verification. It generates a set of models that describe multiple different interleaving of concurrent behaviors in the system that have the exact same effect on the WBSN global behavior, thus only such a sequence of models needs to be checked. Thus, it only remains to translate these models into some form of models for verification, which can easily be converted into the Promela language, and then the model checking can be performed. The Promela language is used for writing these models to be checked by SPIN [12]; SPIN is a software tool for verifying the models of physical systems; then, behavioral properties

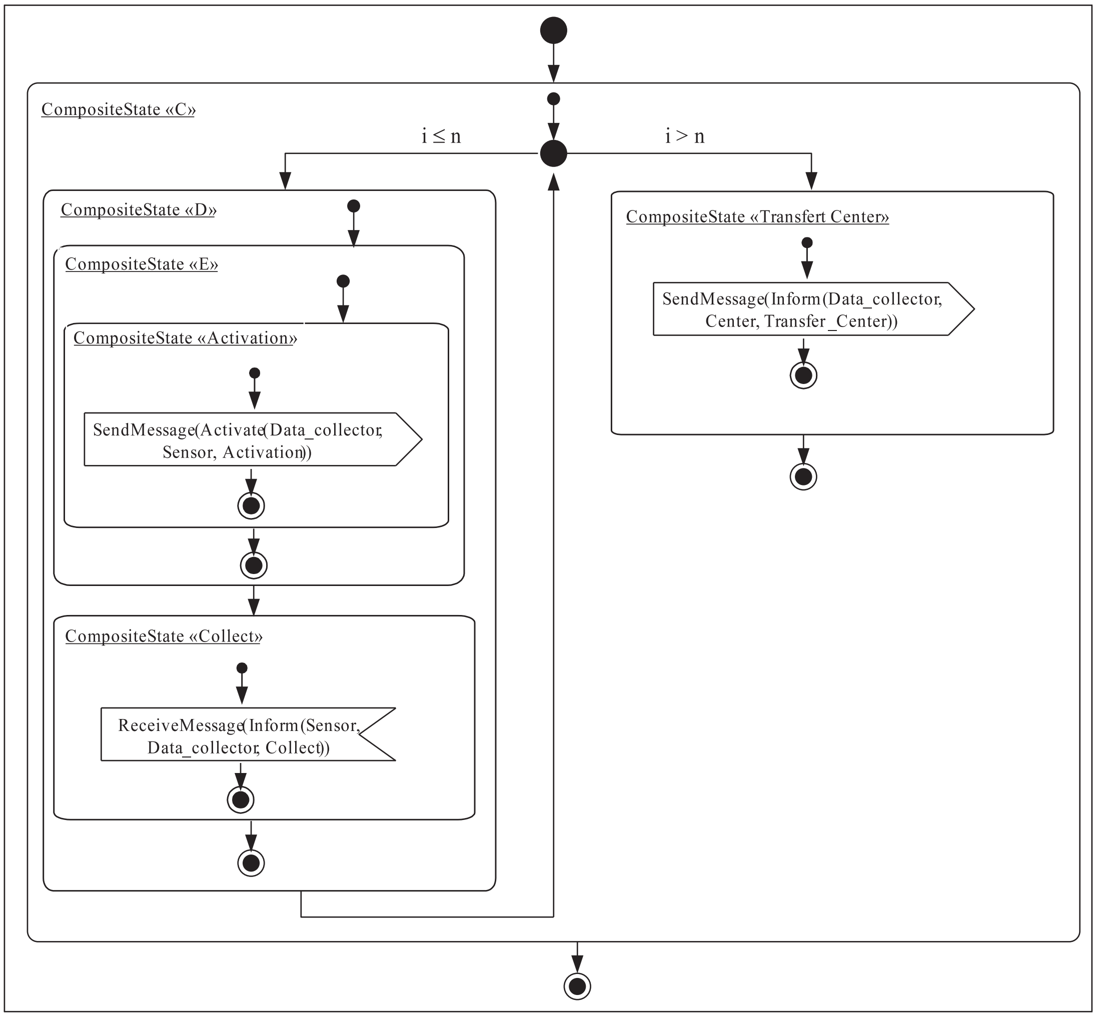

CompositeState «C»
i ≤ n
i > n
CompositeState «D»
CompositeState «E»
CompositeState «Activation»
SendMessage(Activate(Data_collector, Sensor, Activation))
CompositeState «Collect»
ReceiveMessage(Inform(Sensor, Data_collector, Collect))
CompositeState «Transfert Center»
SendMessage(Inform(Data_collector, Center, Transfer_Center))

Fig. 7. The derived behavior of the data collector.

(correctness properties) that express requirements on the system's behavior are specified; finally, the model checker is run to check whether the behavioral properties hold for the model.

To accomplish this goal, we suggest the use of a model-based approach to achieve the verification process of a given WBSN system. This MDE approach will allow developers to obtain automatically the models that will be verified by the SPIN model checker. It transforms the WBSN components' behaviors (design models) to the verification-specific models. A set of rules is required to manage the model transformations. Finally, a Promela program is obtained from these specific models through a model-to-text transformation. It is combined with the correctness properties to realize the system analysis and verification. Thus, applying this model-driven verification process requires the definition of the appropriate meta-models at each level of abstraction with the corresponding model transformations. This process consists of the steps shown in Fig. 9:

1. The definition of the verification-specific meta-model.
2. The definition of the model-to-model transformation that maps the concepts from the design meta-model to those of the verification-specific meta-model.
3. The definition of the model-to-text transformation that transforms the verification specific-models to a Promela program.
4. The definition of the behavioral properties (correctness properties).
5. The verification of the WBSN system, specified in Promela, with the SPIN model checker.

6.1. The verification-specific meta-model

All of the systems can be modeled using many strategies for modeling to optimize the clarity and provide an abstract view of the system. The model is then used to achieve formal verification via model checking. Regardless of the original form of the system model, we eventually translate it into some form of a Communicating Finite State Machine (FSM), which is simply a type of graph called an automaton. As the name implies, this finite state machine is a model of behavior using system states and transitions between states. A transition is a state change that is triggered by an event, i.e., transitions map some state-event pairs to other states. As indicated in the name, the set of states should be finite. Additionally, it is assumed that there is a finite set of distinct events. Subsequently, the set of transitions is finite as well.

To translate the derived WBSN components' behaviors to a set of communicating finite state machines, we define a state for each collaboration in which the component participates. Therefore, each state in the graph is labeled by the collaboration name. We consider a component's behavior transition from one state to another when the component events can be performed. Thus, each component's behavior is modeled by a finite state machine (FSM). This FSM consists of a set of states (set Q), a set of events (set X), a

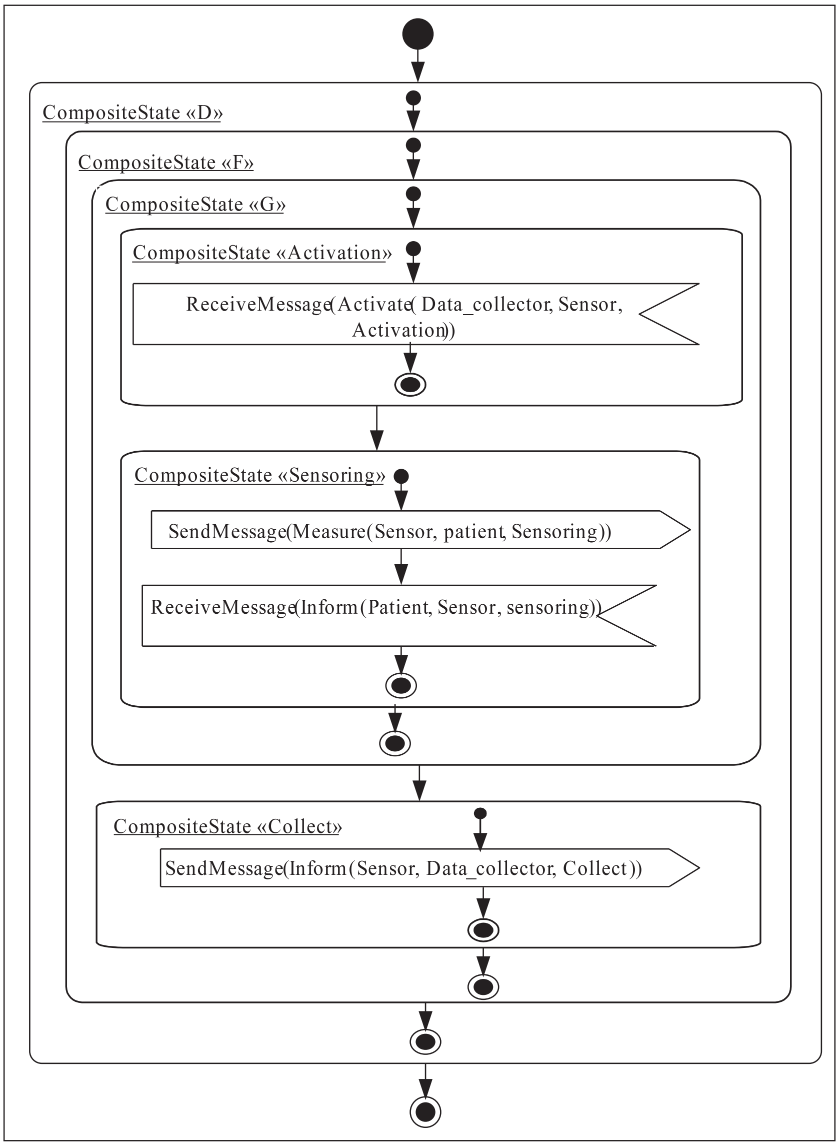

CompositeState «D»
CompositeState «F»
CompositeState «G»
CompositeState «Activation»
ReceiveMessage(Activate(Data_collector, Sensor, Activation))
CompositeState «Sensoring»
SendMessage(Measure(Sensor, patient, Sensoring))
ReceiveMessage(Inform(Patient, Sensor, sensoring))
CompositeState «Collect»
SendMessage(Inform(Sensor, Data_collector, Collect))

Fig. 8. The derived behavior of the sensor.

set of final states (set F) and a state transition function. Each state has rules that describe the action of the machine for every input or output event, which is represented in the state transition mapping function. The state transition function takes the current state and an event or more events and returns the next state. This finite state machine is formally defined as a quintuple(X, Q, q_0, δ, F) such that

- X = a finite non-empty set of events (input and output events: receive messages and send messages)
- Q = a finite non-empty set of states
- q_0 = an initial state, q_0 is an element of Q
- δ = mapping of Q $\times$ X into Q called the state transition function, i.e., $Q \times X \rightarrow Q$
- F = set of final states where F is a subset of Q

From the mathematical interpretation above, it can be said that a finite-state machine contains a finite number of states. Each state accepts a finite number of input and output events, and each state has rules that describe the action of the FSM for every input or output event, which is represented in the state transition mapping function. At the same time, an input or output event can cause the machine to change states. For every input or output event, there is exactly one transition out of each state.

The definition of the verification-specific meta-model aims to describe the WBSN behavior. Models at this level must provide a clear picture of the system behavior. The communicating finite state machine meta-model (Fig. 10) has been designed to provide the appropriate concepts and their relationships, which describe a role behavior. Such a meta-model is considered to be the target meta-model of the model-to-model transformation during the verification process. It defines the FSM meta-model with its main classes and associations. An FSM consists of several *States* and *Transitions*. The *State* class models a situation or a significant phase in the life cycle of the system component. A *Transition* class allows specifying the control flow (connections) between the *State* classes. It models a relationship between a source state and a target state, which represents how the system component can respond to the occurrence of an event when the component is in the source state. The behavior of a WBSN component is defined as an automaton whose language consists of the set of all of the possible behaviors of the system's component. A component's behavior is essentially a finite sequence of WBSN component states.

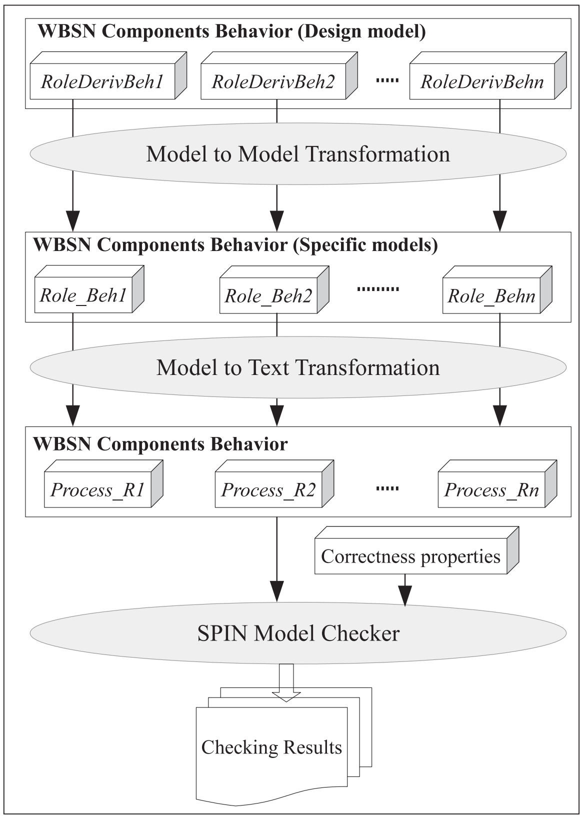


Fig. 9. The verification process.

6.2. The model-to-model transformation

The model-to-model transformation (M2M) transforms automatically the design models, which describe the behavior of each WBSN component, into finite state machines models. This transformation governs the translation process. It is based on a set of transformation rules. A rule consists of transforming a concept outlined in the design meta-model to a corresponding concept in the target specific meta-model. For this purpose, we have defined the function named *Translate(Design_Concept, FSM_Concept, Event)*. The Event parameter in this function represents the actions that will be associated with the transitions that cause the system to change state. This M2M transformation is designed to generate the different concepts of a finite state machine for each WBSN component behavior. These models are used in the model-to-text transformation for the automatic generation of code. The figure (Fig. 11) illustrates the communicating finite state machines that are generated for the WBSN application (Sensor and Data collector).

For example, the automaton in Fig. 11 specifies the data collector behavior. The states are labeled with the names of the collaborations in which it participates. The transitions are labeled by actions that cause the system to change state. The data collector behavior starts at the *Initial_state*. When the data collector sends an *Activate_Sensor* message, it moves to the *Activation* state. At this stage, it waits for an *Information_data* from the sensor to move to *Collect* state. Then, it decides to activate another sensor and moves to *Activation* state if necessary. Once the data are collected, it transfers the information to the data center and moves to the *Transfer_center* state and finally to the *Final_state*.

6.3. The model-to-text transformation

The model-to-text (M2T) transformation aims to generate automatically a Promela program. This activity takes as input the communicating finite state machine models, describing the WBSN be-

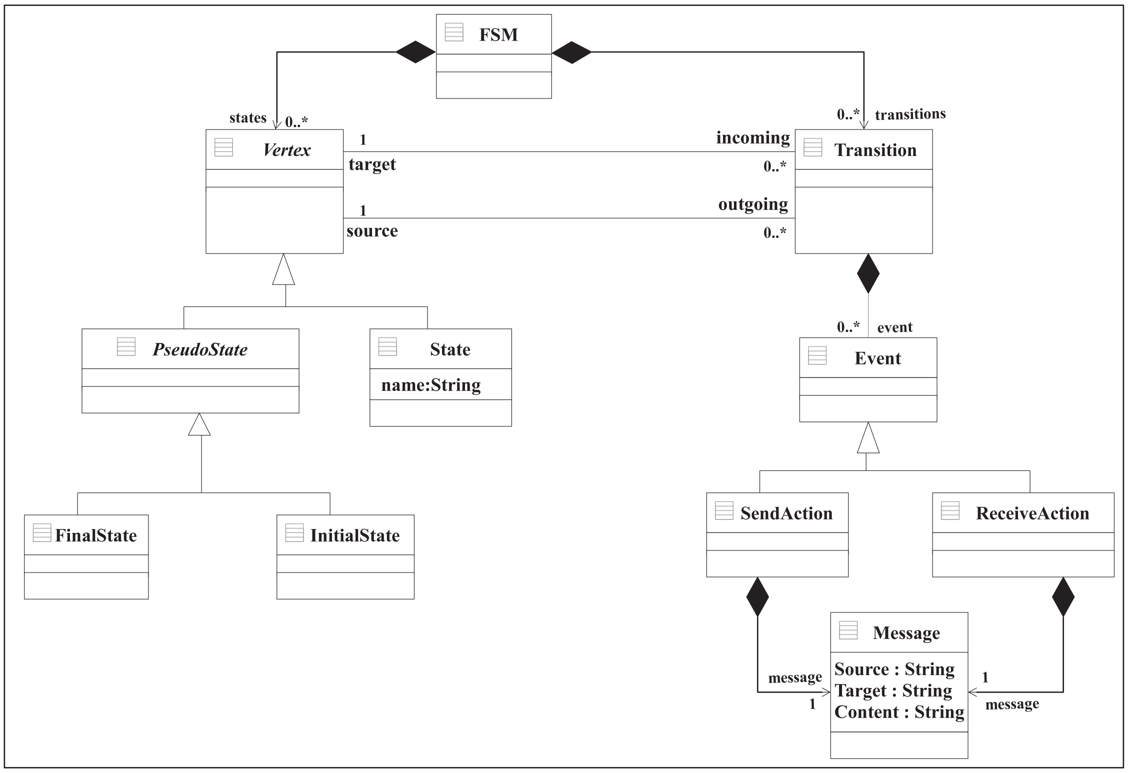


Fig. 10. The finite state machine (FSM) meta-model.

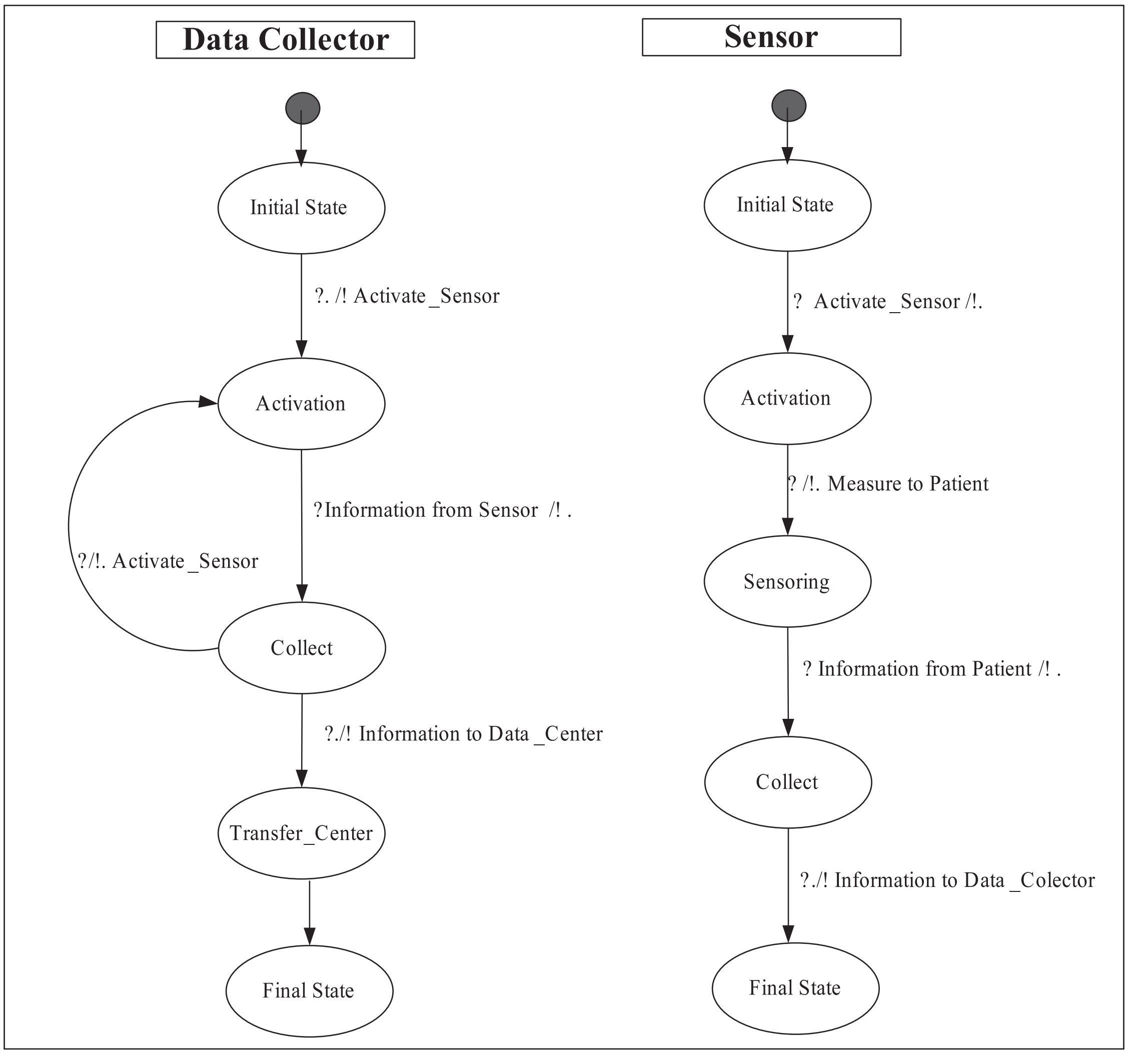


Fig. 11. The FSM of the data collector and the sensor.

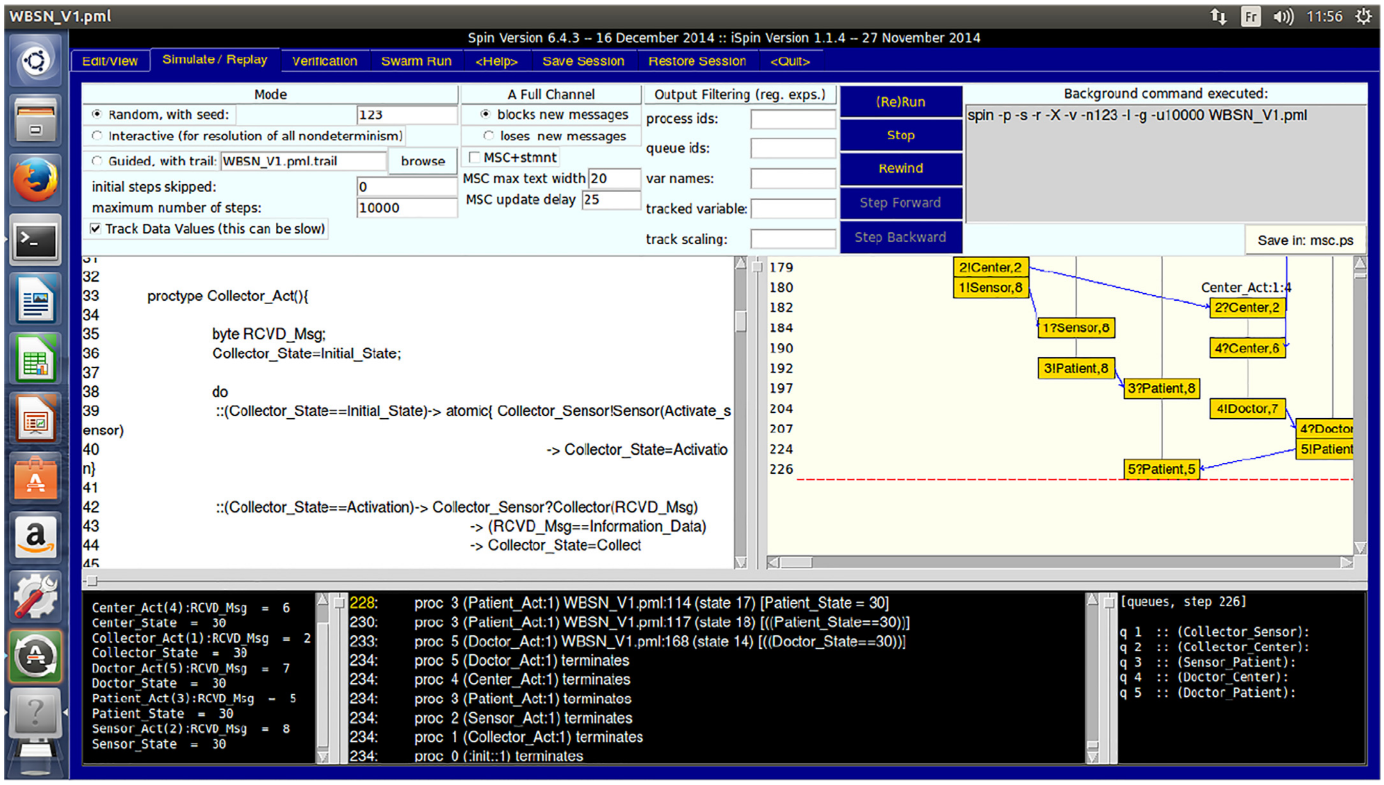
haviors, and generates as output the application source code to be validated by the SPIN model checker. SPIN is a software tool for verifying models of physical systems described in the Promela language. Promela (Process Meta Language) is a validation modeling language. This language allows the dynamic creation of concurrent processes. Communication via message channels can be defined to be synchronous or asynchronous. Promela is a language that ignores the details of distributed systems that are not part of the interaction process. The SPIN tool is used to validate fractions of the system behaviors that are considered to be suspect.

The generated Promela program consists of *processes*, message *channels, variables* and a *main* section that defines how the actions of the processes may be interleaved in time. Processes are global objects. They specify behaviors, and in the WBSN application, each process represents the behavior of a system component. Message channels and variables can be declared either globally or locally within a process. Channels and global variables define the environment in which the processes run. Thus, we obtain, after the M2T transformation, a process that specifies the behavior of each WBSN component. Listing 1 lists the necessary global variables and message channels, and Listing 2 lists the code of the data collector process and the main section.

The SPIN tool executes the program in a stepwise manner. In each step, one executable basic statement is selected at random. The SPIN tool examines its corresponding executability clause to determine if this statement is executable or not. When this condition is evaluated to true, the effect clause from the statement is then applied, and the current state of the process is updated. The SPIN continues executing statements until no executable statements remain, which will occur when either (1) the number of active processes goes to zero and all of the processes terminate without errors, or (2) if an invalid end-state is reached.

6.4. Behavioral properties (correctness properties)

Distributed systems such WBSNs are routinely developed from their specifications. However, once these systems are developed, there is no way to check whether the developed system follows its specification. An exhaustive checking method is impractical to verify the correctness of the distributed systems. The problem with such a method is that it generates an astronomical number of possible computations. The most appropriate method to check that a system follows its specifications must be based on a formal, very accurate, clear notion about the statement of the specifications. Therefore, the logic is a good means to express the correctness properties. Logic has the ability to express system specifications in a concise and clear way and thereby enables automation of the verification process. Concurrent systems necessarily involve the no-



Mode
● Random, with seed: 123
○ Interactive (for resolution of all nondeterminism)
○ Guided, with trail: WBSN_V1.pml.trail browse
initial steps skipped: 0
maximum number of steps: 10000
☑ Track Data Values (this can be slow)

A Full Channel
● blocks new messages
○ loses new messages
☐ MSC+stmnt
MSC max text width 20
MSC update delay 25

Output Filtering (reg. exps.)
process ids:
queue ids:
var names:
tracked variable:
track scaling:

Background command executed:
spin -p -s -r -X -v -n123 -l -g -u10000 WBSN_V1.pml

Save in: msc.ps

```
proctype Collector_Act(){

        byte RCVD_Msg;
        Collector_State=Initial_State;

        do
        ::(Collector_State==Initial_State)-> atomic{ Collector_Sensor!Sensor(Activate_sensor)
                                                        -> Collector_State=Activation}

        ::(Collector_State==Activation)-> Collector_Sensor?Collector(RCVD_Msg)
                                          -> (RCVD_Msg==Information_Data)
                                          -> Collector_State=Collect
```

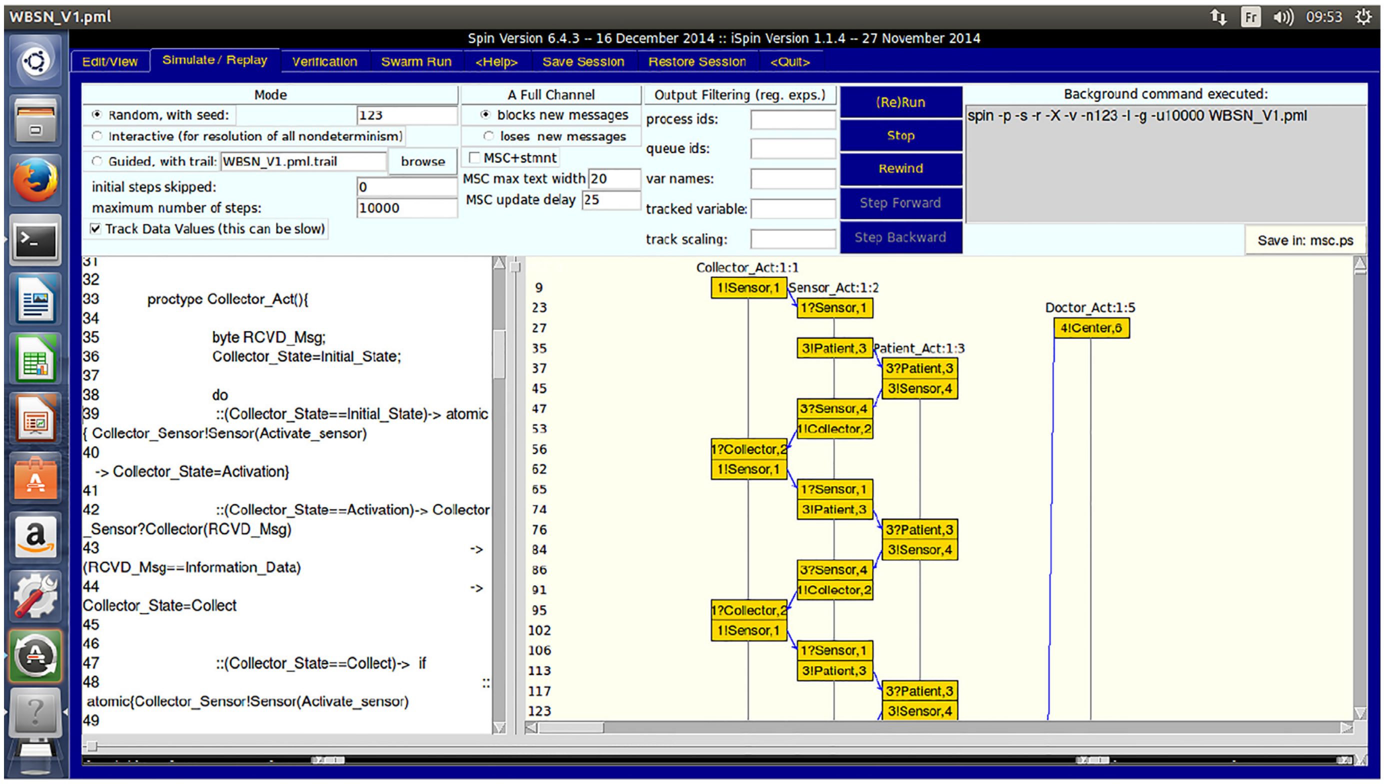
```
Center_Act(4):RCVD_Msg  =  6
Center_State  =  30
Collector_Act(1):RCVD_Msg  =  2
Collector_State  =  30
Doctor_Act(5):RCVD_Msg  =  7
Doctor_State  =  30
Patient_Act(3):RCVD_Msg  =  5
Patient_State  =  30
Sensor_Act(2):RCVD_Msg  =  8
Sensor_State  =  30
```

```
228:   proc 3 (Patient_Act:1) WBSN_V1.pml:114 (state 17) [Patient_State = 30]
230:   proc 3 (Patient_Act:1) WBSN_V1.pml:117 (state 18) [((Patient_State==30))]
233:   proc 5 (Doctor_Act:1) WBSN_V1.pml:168 (state 14) [((Doctor_State==30))]
234:   proc 5 (Doctor_Act:1) terminates
234:   proc 4 (Center_Act:1) terminates
234:   proc 3 (Patient_Act:1) terminates
234:   proc 2 (Sensor_Act:1) terminates
234:   proc 1 (Collector_Act:1) terminates
234:   proc 0 (:init::1) terminates
```

```
[queues, step 226]

q 1  :: (Collector_Sensor):
q 2  :: (Collector_Center):
q 3  :: (Sensor_Patient):
q 4  :: (Doctor_Center):
q 5  :: (Doctor_Patient):
```

Fig. 12. The simulation results.



Mode
● Random, with seed: 123
○ Interactive (for resolution of all nondeterminism)
○ Guided, with trail: WBSN_V1.pml.trail browse
initial steps skipped: 0
maximum number of steps: 10000
☑ Track Data Values (this can be slow)

A Full Channel
● blocks new messages
○ loses new messages
☐ MSC+stmnt
MSC max text width 20
MSC update delay 25

Output Filtering (reg. exps.)
process ids:
queue ids:
var names:
tracked variable:
track scaling:

Background command executed:
spin -p -s -r -X -v -n123 -l -g -u10000 WBSN_V1.pml

Save in: msc.ps

```
proctype Collector_Act(){

        byte RCVD_Msg;
        Collector_State=Initial_State;

        do
        ::(Collector_State==Initial_State)-> atomic
{ Collector_Sensor!Sensor(Activate_sensor)

 -> Collector_State=Activation}

        ::(Collector_State==Activation)-> Collector
_Sensor?Collector(RCVD_Msg)
                                                ->
(RCVD_Msg==Information_Data)
                                                ->
Collector_State=Collect


        ::(Collector_State==Collect)-> if
                                                ::
 atomic{Collector_Sensor!Sensor(Activate_sensor)
```

Fig. 13. A scenario of the random simulation for the WBSN system.

tion of time. Therefore, the propositional logic is not sufficiently expressive to describe such real systems.

Linear Temporal Logic (LTL) was introduced as a vehicle for reasoning about concurrent programs, by Pnueli [42]. Temporal logics are modal logics that are oriented at describing the temporal ordering of events. The notion of time is not explicitly introduced in the description of ordering of events. Temporal logics are especially effective for describing concurrent systems [33]. LTL model checkers follow the automata-theoretic approach [43]. In such model checkers, the system model and the negation of its correctness property (expressed in temporal logic) are transformed into finite automata and are executed simultaneously. A model checker searches for an accepted path by both automata. If it finds one, then this accepting path represents a case where the system allows a behavior that violates the correctness specification; therefore, the model is not correct, and the behavior can be reported as a counterexample to the correctness property. If no such behavior exists (i.e., a correctness property holds on the system), then there is no behavior that both is a legal execution of the model and also satisfies the negation of the correctness property.

Linear Temporal Logic is used to specify and verify most correctness properties of models. It is the formal logic used for verification in SPIN [12,44]. Linear Temporal Logic (LTL) reasons over linear traces through time. At each time instant, there is only one real future timeline that will occur. Traditionally, that timeline is defined as starting now, in the current time step, and progressing infinitely into the future. LTL extends propositional logic with temporal operators. The formulas of propositional calculus are composed from atomic propositions and the operators (not= ! ,and= &&, or= $\|$, implies= $\rightarrow$ and equivalent = $\leftrightarrow$). LTL formulas are composed of a finite set of atomic propositions and of operators that include the operators of propositional calculus as well as temporal operators. The temporal operators are (always= $\square$, eventually= $\diamond$, until= $\mathcal{U}$).

The behavioral properties (specifications) are the temporal logic formulas that we define to describe the desirable behaviors that the WBSN system must have. To accomplish that goal, we design a set of specifications to describe a coherent WBSN behavior. They are supplied to a model checker to verify that the system has the list of emergent behaviors described by our specification set. For the WBSN application, we have described different types of behaviors, which specify behavioral properties (correctness properties). There are two fundamental properties: safety and liveness properties. These categories were originally introduced by Lamport [45]. A Safety property expresses the sentiment that something bad never happens. The system always had a good behavior. The safety properties reason about reaching specific states in the WBSN behavior. A Liveness property expresses the sentiment that something good must eventually happen. The system must have eventually a good behavior. The liveness properties reason about the control flow between states. Liveness is loosely defined as a program's ability to execute in a timely manner.

The correctness properties for the WBSN application that describe the desirable behaviors that the system must have are

- The passage of the data collector activates the sensor to perform the measurement of physiological or environmental signs
- Once the sensor is activated, it must perform the measurement of a sign or parameter
- The data collector must collect information before transferring it to the data center
- If the sensor is activated, then the measured data will be transmitted to the data collector
- If the sensor is activated, then the measured data will be transmitted after a finite time to the data center
- The doctor cannot provide treatment unless the measured data were transferred to the data center
- The patient can receive treatment only if the doctor has consulted the data center
- Sensors should be already alerted before measuring a physiological or environmental sign
- The measured data should be already transferred to the data center before the doctors make a decision
- The measured data should be already collected before the data's eventual transfer to the data center
- At the end of the system execution, the communication channels should be empty

Some of these properties expressed in Linear Temporal Logic formulas are listed in Listing 3.

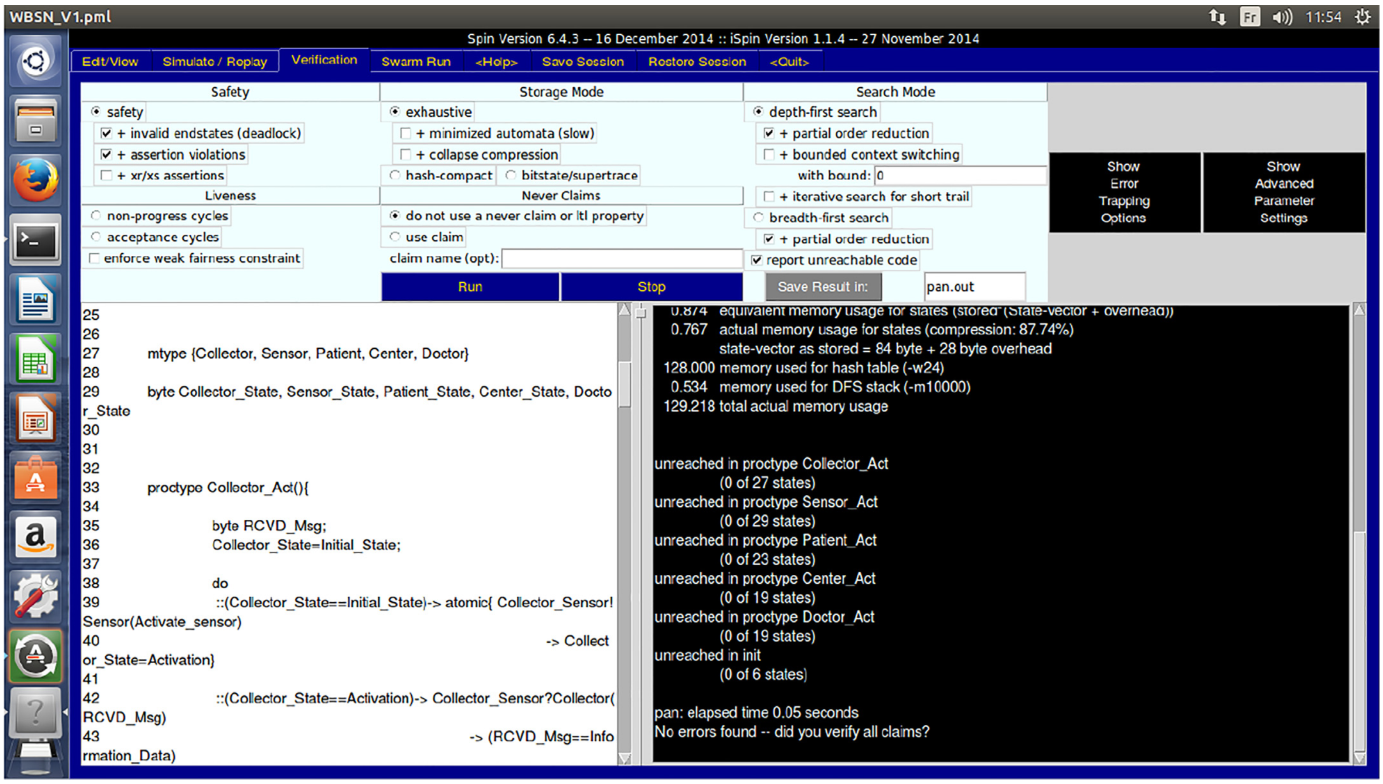
6.5. Model checking (SPIN)

Exhaustive checking is an impractical method for verifying the correctness of concurrent programs. It generates a large number of possible computations, which cannot be achieved with a finite memory space. Therefore, such exhaustive verifications must be performed by an efficient software tool that uses a minimal amount of memory. This tool must establish with certainty whether a given behavior is error-free.

SPIN is perhaps the leading example of such a tool. It is a model checker that has been developed over many years by Holzmann [12,44]. Originally designed for verifying communications protocols, it has since become one of the most powerful model checkers. It is applied in industrial practices to solve real problems in the construction of large-scale distributed software systems. This tool has been widely used for the verification of systems in many applications, including operating systems software and communications protocols. SPIN is particularly well-suited for checking concurrent and distributed systems that are based upon the interleaving of atomic instructions. SPIN is a software tool for analyzing and verifying the logical consistency of concurrent systems. The system to be checked is described in a modeling language called Promela (Process Meta Language).

The Promela program that describes the WBSN system is generated automatically by the model-to-text transformation process. It consists of processes in which each process specifies the behavior of a WBSN component. SPIN performs random simulations of the WBSN system's execution. During the simulation and verification, SPIN checks for the absence of deadlocks, unspecified receptions, and unexecutable code. The simulation of the WBSN execution in a concurrent environment by SPIN has found no errors. All of the processes terminate their execution without errors, deadlocks or indefinite cycles. The simulation results are shown in Fig. 12. Fig. 13 shows one scenario of a random execution for the WBSN system, which is expressed with a sequence diagram. SPIN has also found no errors during the safe verification of the WBSN components. The verification results are shown in Fig. 14.

We performed several random simulations of the WBSN application by varying each time the number of sensors attached to a patient. The results of these simulations are listed in Table 1. These results show that regardless of the number of sensors attributed to a patient, the simulation ends without errors in a finite amount of time. The results showed that the execution of the WBSN components' behaviors in a distributed environment generates no unexpected message and all of the transmitted messages are received. These results are represented by the curves in Fig. 15. Additionally, the results shows that the variation in the number of messages according to the number of sensors is stable. The number of messages does not change during the different executions for a given number of sensors. It increases with the increase of the number of sensors in a constant way.


WBSN_V1.pml
Spin Version 6.4.3 -- 16 December 2014 :: iSpin Version 1.1.4 -- 27 November 2014
Edit/View
Simulate / Replay
Verification
Swarm Run
<Help>
Save Session
Restore Session
<Quit>
Run
Stop
Save Result in:
Show Error Trapping Options
Show Advanced Parameter Settings

Safety
- [✓ safety]
- [✓] + invalid endstates (deadlock)
- [✓] + assertion violations
- [ ] + xr/xs assertions

Liveness
- [ ] non-progress cycles
- [ ] acceptance cycles
- [ ] enforce weak fairness constraint

Storage Mode
- [✓ exhaustive]
- [ ] + minimized automata (slow)
- [ ] + collapse compression
- [ ] hash-compact
- [ ] bitstate/supertrace

Never Claims
- [✓ do not use a never claim or ltl property]
- [ ] use claim
- claim name (opt):

Search Mode
- [✓ depth-first search]
- [✓] + partial order reduction
- [ ] + bounded context switching
- with bound: 0
- [ ] + iterative search for short trail
- [ ] breadth-first search
- [✓] + partial order reduction
- [✓] report unreachable code

Save Result in: pan.out

```
mtype {Collector, Sensor, Patient, Center, Doctor}

byte Collector_State, Sensor_State, Patient_State, Center_State, Doctor_State



proctype Collector_Act(){

          byte RCVD_Msg;
          Collector_State=Initial_State;

          do
           ::(Collector_State==Initial_State)-> atomic{ Collector_Sensor!Sensor(Activate_sensor)
                                                        -> Collector_State=Activation}

           ::(Collector_State==Activation)-> Collector_Sensor?Collector(RCVD_Msg)
                                                        -> (RCVD_Msg==Information_Data)
```

```
  0.874   equivalent memory usage for states (stored*(State-vector + overhead))
  0.767   actual memory usage for states (compression: 87.74%)
          state-vector as stored = 84 byte + 28 byte overhead
128.000 memory used for hash table (-w24)
  0.534   memory used for DFS stack (-m10000)
129.218 total actual memory usage


unreached in proctype Collector_Act
          (0 of 27 states)
unreached in proctype Sensor_Act
          (0 of 29 states)
unreached in proctype Patient_Act
          (0 of 23 states)
unreached in proctype Center_Act
          (0 of 19 states)
unreached in proctype Doctor_Act
          (0 of 19 states)
unreached in init
          (0 of 6 states)

pan: elapsed time 0.05 seconds
No errors found -- did you verify all claims?
```

Fig. 14. The verification results.

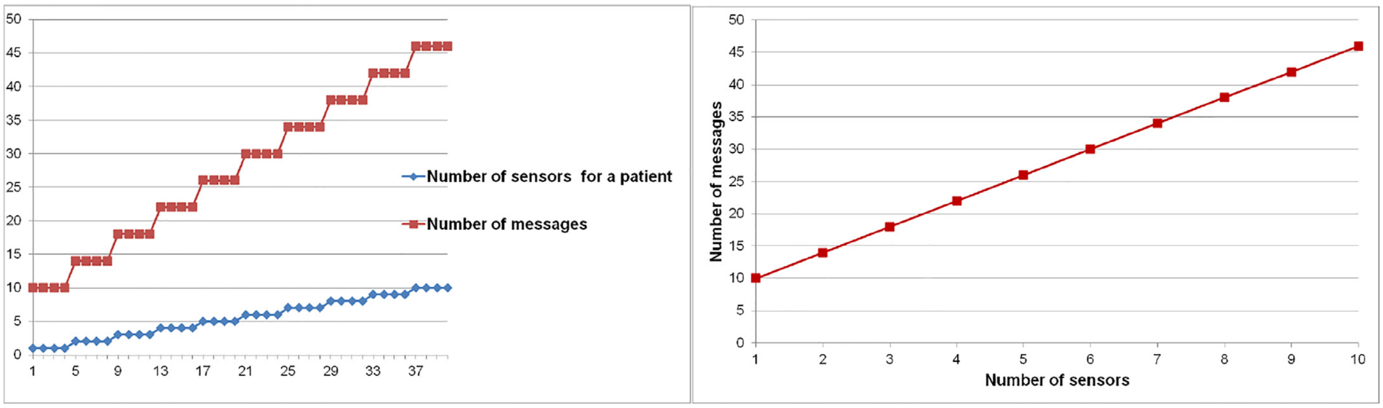


Fig. 15. Simulation results.

Table 1
Simulation results.

Simulation	Number of Sensors	Number of messages	Simulation	Number of Sensors	Number of messages	Simulation	Number of Sensors	Number of messages	Simulation	Number of Sensors	Number of messages	Simulation	Number of Sensors	Number of messages
1	1	10	9	3	18	17	5	26	25	7	34	33	9	42
2	1	10	10	3	18	18	5	26	26	7	34	34	9	42
3	1	10	11	3	18	19	5	26	27	7	34	35	9	42
4	1	10	12	3	18	20	5	26	28	7	34	36	9	42
5	2	14	13	4	22	21	6	30	29	8	38	37	10	46
6	2	14	14	4	22	22	6	30	30	8	38	38	10	46
7	2	14	15	4	22	23	6	30	31	8	38	39	10	46
8	2	14	16	4	22	24	6	30	32	8	38	40	10	46

Table 2 shows the variation in the simulation time according to the number of sensors assigned to a patient. These results show that the simulation time is finite and stable for a given number of sensors during all of the executions (Fig. 16).

SPIN can also perform an efficient, exhaustive verification of the system's correctness properties. During the verification, SPIN checks whether the correctness properties expressed in linear temporal logic (LTL) formulas hold for the model. For the WBSN application, we combine the Promela program generated by the M2T process with the correctness properties expressed in LTL formulas. Then, we perform verification of the correctness properties described in paragraph 6.4, and the SPIN model checker shows

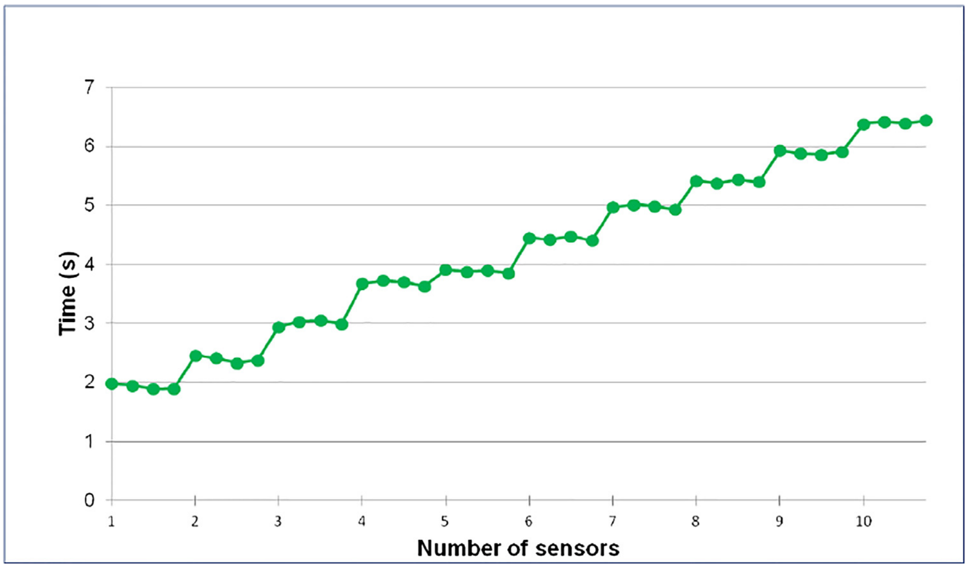
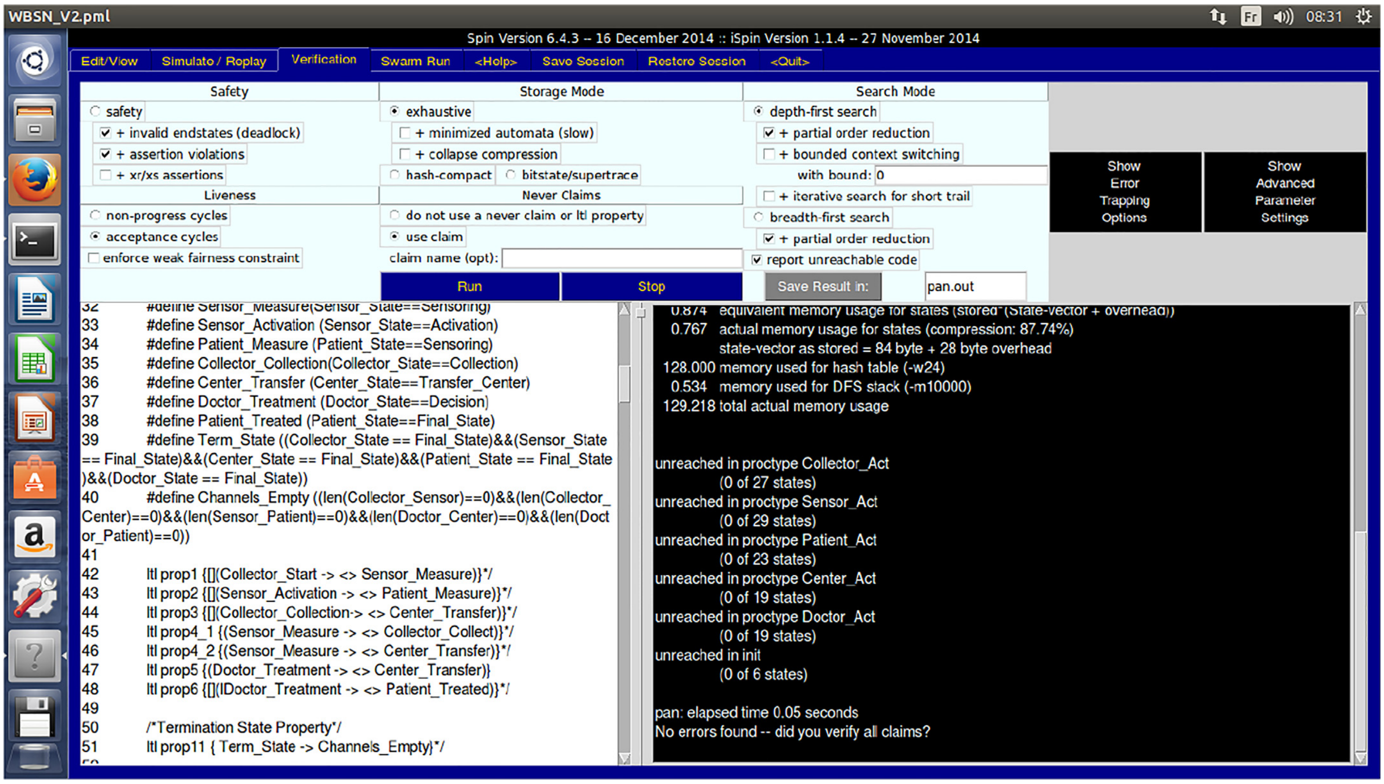


Fig. 16. Variation of the simulation time.


WBSN_V2.pml
Fr
08:31
Spin Version 6.4.3 -- 16 December 2014 :: iSpin Version 1.1.4 -- 27 November 2014
Edit/View
Simulate / Replay
Verification
Swarm Run
<Help>
Save Session
Restore Session
<Quit>
Run
Stop
Save Result in:
Show Error Trapping Options
Show Advanced Parameter Settings

Safety
safety
[✓ + invalid endstates (deadlock)]
[✓ + assertion violations]
+ xr/xs assertions

Liveness
non-progress cycles
[✓ acceptance cycles]
enforce weak fairness constraint

Storage Mode
[✓ exhaustive]
+ minimized automata (slow)
+ collapse compression
hash-compact
bitstate/supertrace

Never Claims
do not use a never claim or ltl property
[✓ use claim]
claim name (opt):

Search Mode
[✓ depth-first search]
[✓ + partial order reduction]
+ bounded context switching
with bound: 0
+ iterative search for short trail
breadth-first search
[✓ + partial order reduction]
[✓ report unreachable code]

Save Result in: pan.out

```
        #define Sensor_Measure(Sensor_State==Sensoring)
        #define Sensor_Activation (Sensor_State==Activation)
        #define Patient_Measure (Patient_State==Sensoring)
        #define Collector_Collection(Collector_State==Collection)
        #define Center_Transfer (Center_State==Transfer_Center)
        #define Doctor_Treatment (Doctor_State==Decision)
        #define Patient_Treated (Patient_State==Final_State)
        #define Term_State ((Collector_State == Final_State)&&(Sensor_State
== Final_State)&&(Center_State == Final_State)&&(Patient_State == Final_State
)&&(Doctor_State == Final_State))
        #define Channels_Empty ((len(Collector_Sensor)==0)&&(len(Collector_
Center)==0)&&(len(Sensor_Patient)==0)&&(len(Doctor_Center)==0)&&(len(Doct
or_Patient)==0))

        ltl prop1 {[](Collector_Start -> <> Sensor_Measure)}*/
        ltl prop2 {[](Sensor_Activation -> <> Patient_Measure)}*/
        ltl prop3 {[](Collector_Collection-> <> Center_Transfer)}*/
        ltl prop4_1 {(Sensor_Measure -> <> Collector_Collect)}*/
        ltl prop4_2 {(Sensor_Measure -> <> Center_Transfer)}*/
        ltl prop5 {(Doctor_Treatment -> <> Center_Transfer)}
        ltl prop6 {[](!Doctor_Treatment -> <> Patient_Treated)}*/

        /*Termination State Property*/
        ltl prop11 { Term_State -> Channels_Empty}*/
```

```
  0.874  equivalent memory usage for states (stored*(State-vector + overhead))
  0.767  actual memory usage for states (compression: 87.74%)
         state-vector as stored = 84 byte + 28 byte overhead
128.000  memory used for hash table (-w24)
  0.534  memory used for DFS stack (-m10000)
129.218  total actual memory usage


unreached in proctype Collector_Act
        (0 of 27 states)
unreached in proctype Sensor_Act
        (0 of 29 states)
unreached in proctype Patient_Act
        (0 of 23 states)
unreached in proctype Center_Act
        (0 of 19 states)
unreached in proctype Doctor_Act
        (0 of 19 states)
unreached in init
        (0 of 6 states)

pan: elapsed time 0.05 seconds
No errors found -- did you verify all claims?
```

Fig. 17. Verification of the correctness properties.

Table 2
Variation of the simulation time.

Sensors	time (s)	Sensors	time (s)	Sensors	time (s)	Sensors	time (s)	Sensors	time (s)
1	1,98	3	2,94	5	3,91	7	4,97	9	5,93
1	1,95	3	3,03	5	3,88	7	5,01	9	5,88
1	1,89	3	3,05	5	3,9	7	4,99	9	5,86
1	1,90	3	2,99	5	3,85	7	4,93	9	5,91
2	2,46	4	3,68	6	4,45	8	5,42	10	6,38
2	2,41	4	3,73	6	4,42	8	5,38	10	6,42
2	2,33	4	3,70	6	4,48	8	5,44	10	6,39
2	2,38	4	3,63	6	4,41	8	5,4	10	6,44

```
#define  Activate_sensor     1
#define  Information_Data    2
#define  Request_Measure     3
#define  Measure_Data        4
#define  Treatment           5
#define  Request_Data        6
#define  Data                7

#define  Initial_State       12
#define  Activation          13
#define  Sensoring           14
#define  Collect             15
#define  Transfer_Center     16
#define  Consultation        17
#define  Decision            18
#define  Final_State         30

chan  Collector_Sensor  = [1]  of  {  mtype , byte  };
chan  Collector_Center  = [1]  of  {  mtype , byte  };
chan  Sensor_Patient    = [1]  of  {  mtype , byte  };
chan  Doctor_Center     = [1]  of  {  mtype , byte  };

mtype  {Collector ,  Sensor ,  Center ,  Doctor}

byte  Collector_State ,  Sensor_State ,  Center_State
```

Listing 1. The global variables and message channels.

that the specified properties hold on the derived behaviors of the WBSN. The verification results are illustrated in Fig. 17.

7. The code generation process

The code generation process is the final step of the system development methodology. It is conducted once the verification and validation of the system has been completed (no specification error has been detected). This step consists of the following:

1. The description of the platform-specific meta-model (e.g., the NesC meta-model for sensors).
2. The definition of the model-to-model (M2M) transformation, which maps the concepts in the design meta-model to those in the NesC platform-specific meta-model.
3. The definition of the model-to-text (M2T) transformation, which automatically generates code from the NesC platform-specific models.

The model-to-model (M2M) transformation transforms the UML states' machine models that describe the behavior of each system component into a platform-specific model. This platform-specific model represents the application in the chosen platform (e.g., NesC (TinyOS) [46] for the sensors and Java [47] for data collector and doctors). It is refined by the network expert to augment the model with information about the network specificities related to the target WBSN platform. The model-to-text (M2T) transformation is then accomplished. This transformation focuses on the generation of textual code from the platform-specific model. The generated application code is refined to add some improvements such as application-specific functions or protocol parameters before it is deployed in the WBSN components.

8. Implementation

The meta-models and model transformations described previously have been developed using the Eclipse Modeling Framework (EMF) [48]. Eclipse Modeling Framework (EMF) is an implementation of the OMG standard Meta Object Facility (MOF) [49]. EMOF (Essential MOF) allows designers to create, manipulate and store both models and meta-models. The meta-models and models in this approach are created using EMOF. The compliance of the design model with its meta-model is achieved by the derivation process. The compliance of the FSM models in the verification process with their meta-model is guaranteed by the model-to-model transformation.

The Model-to-model (M2M) transformation is a crucial step in the MDE process. There exist several languages for model-to-model transformations. The ATL language (ATLAS Transformation Language) [50–52] and the QVT language (Query View Transformation) of the OMG [53,54] are two transformation languages that offer a layered architecture. They also share some common characteristics because they initially shared the same set of requirements defined in QVT RFP [55]. They have a similar operational context [56,57]. The M2M transformation rules used in the different steps of this paper are expressed in the ATL language (ATLAS Transformation Language), which provides the standard Eclipse solution for the model-to-model transformations.

According to the derivation process adopted (Fig. 4), the implementation of this process requires the following:

1. The representation of the requirements meta-model, the target meta-model and the WBSN global requirements model instance of the requirements meta-model.

```
proctype Collector_Act(){

byte RCVD_Msg;
Collector_State=Initial_State;

do
::(Collector_State==Initial_State)
      -> atomic{ Collector_Sensor!Sensor(Activate_sensor)
      -> Collector_State=Activation}
::(Collector_State==Activation)
     -> Collector_Sensor?Collector(RCVD_Msg)
     -> (RCVD_Msg==Information_Data)
     -> Collector_State=Collect
::(Collector_State==Collect)
     -> if
         :: atomic{ Collector_Sensor!Sensor(Activate_sensor)
       -> Collector_State=Activation}
         :: atomic{ Collector_Center!Center(Information_Data)
          ->Collector_State=Transfer_Center}
                                        fi
::(Collector_State==Transfer_Center)
     -> Collector_State=Final_State
::(Collector_State==Final_State)-> break
od
}

init {
         run Collector_Act();
         run Sensor_Act();
         run Patient_Act();
         run Center_Act();
         run Doctor_Act();
}
```

Listing 2. The data collector process and the main section.

2. Applying the rules of model transformations specified in the ATL language to the source model. This process generates a UML finite state machine that describes the behavior of each WBSN component.

In the following, we present the ATL code (Listing 4) of rule 1 specified in Section 5.2. This rule is applied for a sub-collaboration only if the helpers IsinTermine() and CibleIsStrong() are evaluated to true. These helpers determine whether a role is a terminating role in the source collaboration. Similar rules are used during the collaboration transformation, but in this case, the rules call other rules to achieve the collaboration transformation. This arrangement is due to the composition of the collaboration according to the requirements meta-model (Fig. 5).

9. Conclusions

The work presented in this paper suggests a Wireless Body Sensor Network (WBSN) architecture. This architecture has been designed to ensure monitoring vital body functions in many functioning scenarios. The WBSN architecture is based on using sensor nodes, to measure vital body parameters, in combination with a mobile data collector, to gather the measured data and transmit it to the medical center. We have defined a model-driven process to derive the appropriate behavior of each node in the WBSN system from the system global behavior. Coordination messages are included in the system-derived behaviors to realize the collaboration and to ensure global synchronization and coordination among the different WBSN components. This derivation process can play an important role in the WBSN application development by increasing the abstraction level and increasing the reuse of the derived behavior on multiple platforms. It is also possible to treat the different scenarios without the need for a new system design.

This paper suggests also an approach to verify and validate the WBSN derived behaviors. The verification and validation of the derived behaviors can ensure that the collaboration of these behaviors satisfy the initial requirements. This approach complements the derivation approach, and both provide a reliable development methodology for the construction and maintenance of WBSN applications. Furthermore, the use of a software engineering approach to support such a derivation and verification process can lead to highly platform-independent models. This model-driven checking

```
#define  Collector_Start  (Collector_State==Initial_State)
#define  Sensor_Measure(Sensor_State==Sensoring)
#define  Sensor_Activation  (Sensor_State==Activation)
#define  Patient_Measure  (Patient_State==Sensoring)
#define  Collector_Collection(Collector_State==Collection)
#define  Center_Transfer  (Center_State==Transfer_Center)
#define  Doctor_Treatment  (Doctor_State==Decision)
#define  Patient_Treated  (Patient_State==Final_State)


 ltl  prop1  {[](Collector_Start -> <> Sensor_Measure)}
 ltl  prop2  {[](Sensor_Activation -> <> Patient_Measure)}
 ltl  prop3  {[](Collector_Collection-> <> Center_Transfer)}
 ltl  prop4_1  {(Sensor_Measure -> <> Collector_Collect)}
 ltl  prop4_2  {(Sensor_Measure -> <> Center_Transfer)}
 ltl  prop5  {(Doctor_Treatment -> <> Center_Transfer)}
 ltl  prop6  {[](lDoctor_Treatment -> <> Patient_Treated)}
 ltl  prop7  { Term_State -> Channels_Empty}
 ltl  prop8  { Sensor_Measure -> (Sensor_Alert == Alerted)}
 ltl  prop9  { Patient_Measure -> (Patient_Alert == Alerted)}
 ltl  prop10 {Doctor_Decision -> (Data_Rec==Recorded)}
```

Listing 3. The WBSN behavioral properties.

process is mainly based on automatic model transformations of the derived WBSN models to the specific models that are required by a model checker to accomplish the verification. A Promela program is then obtained from these specific models through a model-to-text transformation. The issued Promela program is combined with the correctness properties to realize the system analysis and verification. We checked the safety and liveness properties. These properties are defined on the WBSN global behavior. They are expressed in linear temporal logic (LTL) formulas.

The random simulation of the WBSN derived behaviors performed by the SPIN model checker has allowed us to explore several scenarios of system execution. The results showed that the executions of these behaviors in a distributed environment generate no unexpected message and all of the transmitted messages are received. In addition, it leads to no deadlock, it always ends in a safe manner and it does not generate a run time error. The verification of the correctness properties by SPIN has showed the absence of errors and that the derived behaviors satisfy the WBSN global specification. All of the correctness properties expressed in the WBSN global behavior hold for the derived behaviors. This verification allowed us to validate the WBSN derivation process. Therefore, the derivation process and its verification constitute a robust and reliable approach for the construction and maintenance of WBSN systems.

The growing complexity of WBSN systems as well as changing conditions in the operating environment demand that we accommodate to the environment changes with the existing designed WBSN system. A possible solution that we envisage is the use of mechanisms for effecting behavioral enhancements or changes in the running systems. This solution has been called dynamic adaptation. Usually, the changes are introduced at a high level (requirements specification level) before being propagated to the existing system components. In future work, we plan to extend this work by including the dynamic adaptation of the derived behavior to the various functioning scenarios of the hospital. We propose to use a model-driven approach to address the dynamic changes in the WBSN specifications and to propagate the changes to the existing system components. This approach describes a process for change detection, behavior modification and the automatic derivation of the corresponding components behaviors. It is mainly concerned with the ability of software systems to withstand changes in their global requirements specifications. This new approach consists of a process architecture, a set of components to ensure the structural and behavioral conformance of the derived WBSN behaviors after the changes, and an adaptation algorithm to allow the dynamic adaptation. It addresses the dynamic evolution of a WBSN global specification and the automatic derivation of the system behavior components during the system's life cycle.

In the proposed WBSN architecture, the patients are monitored using a data collector that accumulates and processes the data from a set of medical and environmental sensors. These measured data are relayed to the medical center. This architecture is designed to facilitate an accelerated diagnosis of diseases and also to provide increased efficiency and accuracy in the process. We plan to extend this architecture to create a security aspect in patient monitoring at all times by providing instantaneous attention for emergencies. The system will monitor the deployed alarms for any anomalies that are observed in the system for emergencies that are overlooked by attendees and caregivers. The existence of several anomalies allows them to work in a cooperative way. This approach reduces the workload that is assigned to each of them and hence minimizes the duration of their intervention. In this way, we want to address the cooperation between the actors of the WBSN for the management and monitoring of alarms in emergency responses. To accomplish this goal, we will consider the constraints of time, space and availability of actors during an intervention. In this direction, we plan to minimize the time needed by the doctors, caregivers and technicians to address these situations. The problem consists in determining the best strategy for the participants to cooperate with one another in real time to monitor the anomalies. The problem can be considered to be a game theory problem with the doctors, caregivers and technicians being cooperative players.

```
rule CollaborationS2StateT {
from s:MMg!Sub_collaboration (s.IsinTermine()
                               and s.CibleIsStrong())
to r:    MMd!CompositeState(Name<-s.Name,
                            outgoing<-s.Getoutgoing(),
                            incoming<-s.Getincoming(),
                            states<-Si,transition<-ti,
                            states<-st,states<-sc,
                            transition <-t,
                            transition <-tf,
                            states<-Sf ),
   Si: MMd!InitialState(outgoing<-ti),
   ti: MMd!Transition (source<-Si, target<-sc),
   sc: MMd!State (action<-act, incoming<-ti  ,
                            outgoing <-t),
   act: MMd!DomainAction(Name<-s.Name),
   t  : MMd!Transition (source<-sc,target<-st),
   st: MMd!State (incoming<-t,outgoing<-tf),
   tf  : MMd!Transition (source<-st,target<-Sf),
   Sf:MMd!FinalState (incoming<-tf)
do { thisModule.i<-thisModule.i+1;
     sc.Name<-'S'+thisModule.i.toString();
     thisModule.i<-thisModule.i+1;
     st.Name<-'S'+thisModule.i.toString();
     thisModule.j<-thisModule.j+1;
     ti.Name<-'T'+ thisMo-dule.j.toString();
     thisModule.j<-thisModule.j+1;
     t.Name<-'T'+ thisModule.j.toString();
     thisModule.NameCol <- s.Name;
    st.action<-s.GetRolesInit() ->select(e|
                         e.Name <> thisModule.role)
          ->collect(x| thisModule.SendFlowm(x));
    thisModule.j<-thisModule.j+1;
    tf.Name<-'T'+thisModule.j.toString()
    }}
```

Listing 4. The strong sequencing rule 1 expressed in ATL.

We expect also to check the scalability of this approach by testing more WSN systems. We must take advantage of these applications to make this model-driven approach for derivation and verification more efficient, in terms of the models that it can reason about and the time and space that it requires.

References

[1] M. Winkler, M. Street, K. Tuchs, K. Wrona, Wireless sensor networks for military purposes, Auton. Sens. Netw. Springer Ser. Chem. Sens. Biosens. 13 (2013) 365–394.

[2] R. Pahuja, H.K. Verma, M. Uddin, A wireless sensor network for greenhouse climate control, Pervasive Comput. IEEE 12 (2) (2013) 49–58.

[3] A. Mansour, I. Leblond, D. Hamad, F. Artigas, Sensor networks for underwater ecosystem monitoring and port surveillance systems, in: M. Illias (Ed.), Sensor Networks for Sustainable Development, 2013, pp. 1–25.

[4] M. Reyer, S. Hurlebaus, J. Mander, O.E. Ozbulut, Design of a wireless sensor network for structural health monitoring of bridges, Wireless Sens. Netw. Ecol. Monit. Smart Sens. Meas. Instrum. 3 (2013) 195–216.

[5] P. Khan, A. Hussain, K. Kwak, Medical applications of wireless body area networks, Int. J. Digital Content Technol. Appl. 3 (3) (2009) 185–193.

[6] P. Neves, M. Stachyra, J. Rodrigues, Application of wireless sensor networks to healthcare promotion, J. Commun. Softw. Syst. 4 (3) (2006) 181–190.

[7] B. Wang, L. Wang, S.J. Lin, D. Wu, B.Y. Huang, Y.T. Zhang, Q. Yin, W. Chen, A body sensor networks development platform for pervasive healthcare, in: Proceedings of the 3rd International Conference on Bioinformatics and Biomedical Engineering (ICBBE 2009), 2009, pp. 1–4.

[8] C. Seeger, A. Buchmann, K. VanLaerhovenmy, Healthassistant: a phone-based body sensor network that captures the wearer's exercises throughout the day, in: Proceedings of the 6th International Conference on Body Area Networks (BodyNets), Beijing, China (2011), 2011.

[9] D. Raskovic, T. Martin, E. Javanov, Medical monitoring applications for wearable computing, Comput. J. 47 (4) (2004) 495–504.

[10] U. Anliker, J. Beutel, M. Dyer, P. Lukowicz, G. Troster, A systematic approach to the distributed wearable systems, Comput. J. IEEE Trans. 53 (8) (2004) 1017–1033.

[11] R.W. Butler, G.B. Finelli, The infeasibility of quantifying the reliability of life–critical real-time software, J. IEEE TSE 19 (1) (1993) 3–12.

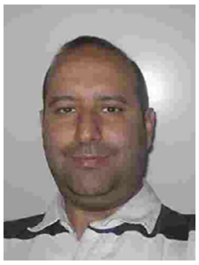
[12] G.J. Holzmann, The Spin Model Checker: Primer and Reference Manual, Addison Wesley, Boston MA, 2004.

[13] K.Y. Rozier, Survey: linear temporal logic symbolic model checking, J. Comput. Sci. Rev. 5 (2) (2011) 163–203.
[14] R. Shah, S. Roy, S. Jain, W. Brunette, Data mules: modeling a three-tier architecture for sparse sensor networks, in: Proceedings of the 2nd ACM International Workshop on Wireless Sensor Networks and Applications (SNPA 2003), 2003, pp. 30–41.
[15] S. Jain, R. Shahand, W. Brunette, G. Borriello, S. Roy, Exploiting mobility for energy efficient data collection in wireless sensor networks, ACM/Springer Mobile Netw. Appl. 11 (3) (2006) 327–339.
[16] H. Jun, W. Zhao, M. Ammar, E. Zegura, C. Lee, Trading latency for energy in wireless ad hoc networks using message ferrying, in: Proceedings of the 1st IEEE Workshop on Pervasive Wireless Networking (PWN 2005), 2005, pp. 220–225.
[17] W. Zhao, M. Ammar, E. Zegura, A message ferrying approach for data delivery in sparse mobile ad hoc networks, in: Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2004), 2004, pp. 187–198.
[18] G. Wang, G. Cao, G. LaPorta, W. Zhang, Sensor relocation in mobile sensor networks, in: Proceedings of the 24th IEEE Conference on Computer Communications (INFOCOM 2005), 4, 2005, pp. 2302–2312.
[19] J. Rao, S. Biswas, Joint routing and navigation protocols for data harvesting in sensor networks, in: Proceedings of the 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2008), 2008, pp. 143–152.
[20] Y.D. Lee, W.Y. Chung, Wireless sensor network based wearable smart shirt for ubiquitous health and activity monitoring, Sens. Actuators B 140 (2009) 390–395.
[21] P. Hii, W. Chung, A comprehensive ubiquitous healthcare solution on an android mobile device, Sensors 11 (2011) 6799–6815.
[22] Y. Zatout, E. Campo, J.-F. Llibre, Toward hybrid wsn architectures for monitoring people at home, in: Proceedings of the International ACM Conference on Management of Emergent Digital EcoSystems (ACM MEDES 2009), 2009, pp. 308–314.
[23] Y. Zatout, R. Kacimi, J. Llibre, E. Campo, Using wireless technologies for health–care monitoring at home, in: Proceedings of the 4th Joint IFIP Wireless and Mobile Networking Conference (WMNC 2011), 2011, pp. 1–6.
[24] L. Xuemei, J.L.L. Jincheng, Home healthcare platform based on wireless sensor networks, in: Proceedings of the Fifth International Conference on Information Technology and Application in Biomedicine, 2008, pp. 263–266.
[25] Y. Hairong, X. Youzhi, M. Gidlund, Experimental e-health applications in wireless sensor networks, in: WRI International Conference on Communications and Mobile Computing, 1, 2009, pp. 563–567.
[26] E. Lawrence, K.F. Navarro, D. Hoang, Y. Lim, Data collection, correlation and dissemination of medical sensor information in a wsn, in: Fifth International Conference on Networking and Services, 2009, pp. 402–408.
[27] H. Yan, Y. Xu, M. Gidlund, Experimental e-health applications in wireless sensor networks, in: International Conference on Communications and Mobile Computing, 2009, pp. 563–567.
[28] S. Mukherjee, K. Dolui, S.K. Datta, Patient health management system using e-health monitoring architecture, in: Advance Computing Conference (IACC), 2014 IEEE International, 2014, pp. 400–405.
[29] B. Akbal-Delibas, P. Boonma, J. Suzuki, Extensible and precise modeling for wireless sensor networks, in: Information Systems: Modeling, Development, and Integration, Lecture Notes in Business Information Processing (LNBIP), 20, 2009, pp. 551–562.
[30] F. Losilla, C. Vicente-Chicote, B. Alvarez, A. Iborra, P. Sanchez, Wireless sensor network application development: an architecture-centric mde approach, in: Proceedings of the ECSA 2007, Lecture Notes in Computer Science, 4758, 2007, pp. 179–194.
[31] C. Vicente-Chicote, F. Losilla, B. Alvarez, A. Iborra, P. Sanchez, Applying mde to the development of flexible and reusable wireless sensor networks, Int. J. Coop. Inf. Syst. 16 (3/4) (2007) 393–412.
[32] W. Kurschl, S. Mitsch, J. Schoenboeck, Modeling distributed signal processing applications, in: Sixth International Workshop on Wearable and Implantable Body Sensor Networks, BSN 2009, 2009, pp. 103–108.
[33] E.M. Clarke, E.A. Emerson, Design and synthesis of synchronization skeletons using branching time temporal logic, in: Proceedings of the Workshop on Logic of Programs, LNCS, 131, 1981, pp. 52–71.
[34] E.M. Clarke, O. Grumberg, D.E. Long, Model checking and abstraction, in: POPL, Proceedings of the 19th ACM Symposium, 1992, pp. 343–354.
[35] E.M. Clarke, O. Grumberg, D.E. Long, Verification tools for finite-state concurrent systems, in: J.W. de Bakker, W.-P. de Roever, G. Rozenberg (Eds.), Decade of Concurrency Reflections and Perspectives (Proceedings of REX School), 803, LNCS, 1993, pp. 124–175.
[36] J.P. Queille, J. Sifakis, Specification and verification of concurrent systems, in: Cesar, Proceedings of the 5th International Symposium on Programming, LNCS, 137, 1982, pp. 337–351.
[37] M.Y. Vardi, Automata-theoretic model checking, in: VMCAI, Proceedings of the 7th International Conference, LNCS, 4339, 2007, pp. 137–150.
[38] Object–Management–Group, Uml 2.0 superstructure specification, OMG Adopted Specification, at http://www.omg.org (2007) (2007).
[39] G.v. Bochmann, Deriving component designs from global requirements, International Workshop on Model Based Architecting and Construction of Embedded Systems (ACES) (2008), 2008.
[40] H. Castejon, G.v. Bochmann, R. Brack, Realizability of collaboration-based service specifications, in: The Asia-Pacific Software Engineering Conference (APSEC) (2007), 2007.
[41] H. Castejon, G.v. Bochmann, R. Brack, Using Collaborations in the Development of Distributed Services, AVANTEL Technical Report, Department of Telematics, 2008. 2008, 37 s.
[42] A. Pnueli, The temporal logic of programs, in: FOCS, Proceedings of the 18th IEEE Symposium, 1977, pp. 46–57.
[43] M.Y. Vardi, P. Wolper, An automata-theoretic approach to automatic program verification, in: Proceedings of the 1st LICS, Cambridge, 1986, pp. 332–344.
[44] G.J. Holzmann, The model checker spin, Form. Methods Softw. Pract. IEEE TSE 23 (5) (1997) 279–295.
[45] L. Lamport, Proving the correctness of multiprocess programs, IEEE TSE 3 (2) (1977) 125–143.
[46] P. Levis, D. Gay, Tinyos Programming, Cambridge University Press, Cambridge, 2009.
[47] J. Gosling, B. Joy, G. Steele, The Java Language Specification, Addison-Wesley Publication, 1996.
[48] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, T.J. Grose, Eclipse Modeling Framework, Addison-Wesley Professional, 2003.
[49] Object–Management–Group, Meta-object facility (mof) 2.0 core specification, at http://www.omg.org/docs/ptc/04-10-15.pdf (2004).
[50] F. Jouault, I. Kurtev, Transforming models with atl, in: Proceedings of MODELS 2005 Workshops, LNCS, 3844, 2006, pp. 128–138.
[51] F. Jouault, F. Allilaire, J. Bezivin, I. Kurtev, Atl: a model transformation tool, J. Sci. Comput. Program. 72 (1–2) (2008) 31–39.
[52] Eclipse, The atlas transformation language (atl) project, at http://www.eclipse.org/m2m/atl/.
[53] Object–Management–Group, Mof query/ view/ transformations (qvt) adopted specification, at http://www.omg.org (2005).
[54] Object–Management–Group, Mof query/ view/ transformations (qvt) adopted specification, at http://www.omg.org (2008).
[55] Object–Management–Group, mof 2.0 query/ views/ transformations rfp, OMG Document ad/2002-04-10, at http://www.omg.org/cgi-bin/doc?ad/2002-04-10 (2002).
[56] F. Jouault, F. Allilaire, J. Bezivin, I. Kurtev, P. Valduriez, Atl: a qvt-like transformation language, in: Proceedings of Companion to the 21st ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications, ACM Press, 2006, pp. 719–720.
[57] F. Jouault, I. Kurtev, On the architectural alignment of atl and qvt, in: Proceedings of the 2006 ACM Symposium on Applied Computing, ACM Press, 2006, pp. 1188–1195.

Ahmed Harbouche has been an Assistant professor in Computer Science at Hassiba Benbouali University, Chlef, Algeria. He obtained his Magister in 1993 at Houari Boumedienne University, Algiers, Algeria in the area of Artificial intelligence. He is member of the research team “models engineering” of the department of computer science at Hassiba Benbouali University. His research interests include multi-agents systems, designing evolving distributed systems, adaptive wireless sensor networks, collaborative distributed applications and systems, formal Methods for the specification, design and verification of distributed systems and distributed collaborative E-health applications.

Noureddine Djedi has been a Professor in Computer Science at Mohamed Khider, Biskra, Algeria. His recent main research interests include Image Synthesis, Virtual Worlds, Knowledge Based Systems, Artificial Life, Modular robotics and Artificial Ecosystems. He obtained his Ph.D. in 1991 at Paul Sabatier University of Toulouse. Professor Djedi has published more than 80 papers in international conferences and journals. He has been a member of the program committee in multiple international conferences.

Mohammed Erradi has been a professor in Computer Science since 1986. He has been leading the Networking and Distributed Systems Research Group since 1994 at ENSIAS (Ecole Nationale d’Informatique et d’Analyse des Systèmes) of Mohammed V-Souissi University (Rabat Morocco). Before joining ENSIAS, Professor Erradi has been affiliated with the University of Sherbrooke and the University of Quebec (Rouyn) in Canada. His recent main research interests include Security Policies and Access Control, Privacy in Big Data, Communication Software Engineering, Distributed Applications and Systems, Formal description techniques, Reflection and Meta-level Architectures. He obtained his Ph.D. in 1993 at University of Montreal in the area of Communicating Software Engineering under Professor Gregor Von Bochmann. He is leading, at the present time, many research projects in collaboration with national and international partners. He is currently the Principal Investigator of a number of research projects grants. Professor Erradi has published more than 100 papers in international conferences and journals. He has organized and chaired many international scientific events (METIS, NETYS) and has been a member of the scientific program committee in multiple international conferences.



Ben-Othman received his B.Sc. and M.Sc. degrees both in Computer Science from the University of Pierre et Marie Curie, (Paris 6) France in 1992, and 1994 respectively. He received his Ph.D. degree from the University of Versailles, France, in 1998. He was an Assistant Professor at the University of Orsay (Paris 11) and University of Pierre et Marie Curie (Paris 6), in 1998 and 1999 respectively. He was an Associate Professor at the University of Versailles from 2000 to 2011. He is currently full professor at the University of Paris 13 since 2011. Dr. Ben-Othman’s research interests are in the area of wireless ad hoc and sensor networks, Broadband Wireless Networks, multi-services bandwidth management in WLAN (IEEE 802.11), WMAN (IEEE 802.16), WWAN (LTE), VANETS, Sensor and Ad Hoc Networks, security in wireless networks in general and wireless sensor ad hoc networks and vehicular ad hoc Networks. His work appears in highly respected international journals and conferences, including, IEEE ICC, Globecom, LCN, MSWIM, VTC, PIMRC etc. He has supervised and co-supervised several graduate students in these areas. He is widely known for his work on wireless ad hoc and sensor Networks, in particular, security. He gave several talks on these topics, as Keynote in conferences Road Transportation System Strategy and Standardization (Korea), WCCCS’13, NSERC DIVA Distinguished Lecture Series (Canada), P2MNET’10, PEDISWESA’09, and as invited talks in GIST (Korea), Seoul National University, KRRI (Korea), USTHB (Algeria), Fes University (Marocco), Hanoi Science and Technology University (Vietnam), Reims (France), Martinique (France), University of Ottawa (Canada), INRS (Canada), Gliwice (Pologne)……. He is an editorial board member of IEEE Communications letters, Wiley Wireless Communications and Mobile Computing (WCMC), Wiley Security and Communication Networks (SCN), Inderscience Int. J. of Satellite Communications Policy and Management, IEEE comsoc Journal of Communications and Networks (JCN) and International Journal On Advances in Networks and Services IJANS. He is also an Associate Editor of Wiley International Journal of Communication Systems (IJCS). He is also editor of Elsevier ICTexpress. He has served as a member of Technical Committees of more than 80 international IEEE/ACM conferences and workshops including ICC, Globecom, MSWIM, LCN. He is a member of IEEE and ACM. He has served as General co-chair of international conference on wireless networks and mobile communications (WINCOM’15), and program chair of IEEE New technologies mobility and security (NTMS’15). He has served as TPC Co-Chair for IEEE Globecom Ad hoc and Sensor and Mesh Networking (Globecom, 2011, 2014). He will serve as TPC Co-Chair for IEEE Globecom Wireless and Mobile Networks symposium (Globecom, 2016), and as a TPC Co-Chair of IEEE Globecom Wireless Communications Symposium (Globecom 2010). He has served as TPC Co-Chair IEEE International Conference on Communications Ad hoc and Sensor and Mesh Networking (ICC 2012, ICC 2014). He is serving as TPC Co-Chair IEEE International Conference on Communications Wireless and Mobile Networks symposium (ICC 2016, 2017). He also has served as TPC Co-Chair Wireless Networking Symposium of The IEEE International Wireless Communications and Mobile Computing Conference (IWCMC 2011, 2012, 2013, 2014, 2015, 2016), ACM International Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet 2010, 2011, 2012), and other conferences as for ICNC, WSCP, CNIT. He has served as Workshop chair for 9th international Workshop on Wireless local Networks (WLN09) and 10th international Workshop on Wireless local Networks (WLN10). He served as a publicity chair of several conferences such as the 12th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM 09), IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WOWMOM 2010), 25th Biennial Symposium on Communications. He has also served as Tutorial chair for Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2014). He served as Local Arrangement Chair for the 13th IEEE International Symposium on Computer Communication (ISCC 09). As IEEE Comsoc Member, he has actively participated to the activities. He was the secretary, Vice chair and currently chair of the IEEE Ad Hoc and sensor networks technical committee since January 2012. He is IEEE Comsoc distinguished lecturer since January 2015. He is member of the IEEE comsoc technical service board since January 2016. He is an active member of IEEE Communication software, CIS-TC, and WTC.

Abdellatif Kobbane is currently an Associate Professor (Professeur Habilité) at the Ecole Nationale Suprieure d'Informatique et d'Analyse des Systemes (ENSIAS), Mohammed V University of Rabat, Morocco since 2009. He received his Ph.D. degree in computer science from the Mohammed V-Agdal University (Morocco) and the University of Avignon (France) in September 2008. He received his research MS degree in computer science, Telecommunication and Multimedia from the Mohammed V-Agdal University (Morocco) in 2003. Doctor Kobbane is Adjunct professor at L2TI laboratory, Paris 13 University, France. His research interests lie with the field of wireless networking, performance evaluation in wireless network and SDN, NGN, Ad-hoc networks, DTNs and Sensors, M2M, Mesh networks, cognitive radio, Mobile computing, Mobile Social networks, Caching and backhaul problem, Internet of things, Beyond 4G and 5G 5G Mobile networks, Future and emerging technologies. He has +10 years of computer sciences and Telecom experience, in Europe (France) and in Morocco, in the areas of Performances evaluation in wireless Mobile networks, Mobile Cloud Networking, Cognitive radio, Ad-hoc networks and Future network. He has co-authored several scientific publications in top IEEE conferences and journals. He has served as General co-chair of the International Conference on Wireless Networks and Mobile Communications (WINCOM 2015), served as program co-chair for IEEE International Wireless Communications and Mobile Computing Conference (IWCMC 16, 15, 14) and and Executive Chair of WINCOM 2017. As an active IEEE Comsoc member, he has participated to the activities of this organizational unit. He is an active member of IEEE Communication software, AHSN, CIS-TC, and WTC. Dr. Kobbane served as a reviewer for many international journals and conferences such COMCOM, IJCS, WCMC, SN, IEEE ICC, IEEE Globecom, IWCMC, ICNC, IEEE WCNC. Dr Kobbane is a Founder & the President of Moroccan Association of Research in Mobile Wireless networks and embedded systems (MobiTic).