Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 35 (2014) 203 – 212

Procedia Computer Science

18th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems - KES2014

A fault fuzzy-ontology for large scale fault-tolerant wireless sensor networks

Yazid Benazzouz, Oum-El-keir Aktouf*, Ioannis Parissis

Univ. Grenoble Alpes, LCIS, F-26000 Valence, France

Abstract

Fault tolerance is a key research area for many of applications such as those based on sensor network technologies. In a large scale wireless sensor network (WSN), it becomes important to find new methods for fault-tolerance that can meet new application requirements like Internet of things, urbane intelligence and observation systems. The challenge is beyond the limit of a single wireless sensor network and concerns multiple widely interconnected sub networks. The domain of fault grows considerably because of this new configuration. In this context, the paper proposes a fault fuzzy-ontology (FFO) for large scale WSNs to be used within a Web service architecture for diagnosis and testing.

Peer-review under responsibility of KES International.
Keywords: Wireless sensor networks, fault tolerance, fault diagnosis, Web Ontology Language, fuzzy logic, Internet of things ;

1. Introduction

New trends in wireless technologies aim to connect wireless sensor networks (WSNs) to Internet and mobile networks such as the infrastructure developed in [1]. This leads to large virtual wireless sensor networks incorporating multiple WSNs with different locations. Application scope for this technology includes environmental parameters observation and measurement, Internet of things, urban intelligence and intelligent transportation systems. Either more, fault tolerance for wireless sensor networks had taken a growing interest due to the importance of dependability and reliability in these systems and their applications. Nevertheless, no deep attention has been paid to fault tolerance in a large scale domain where fault tolerance is not limited to a single wireless sensor network. Many challenges act as a significant brake on the development of a fault tolerance mechanism for large scale wireless sensor networks (LS-WSN). First, LS-WSN often employs multiple networks where protocols, sensors and configurations are very heterogeneous. In addition, WSN are dynamic which means that the topology of the network, sensors and protocols may change by time. However, Fault tolerance technics are generally dependent to the network specifications and fault domain. Besides, in a large scale context there is a need for implicit monitoring that supports automatic composition

* Corresponding author. Tel.: +33-(0)47-575-9446 ; fax: +33-(0)47-575-9450.
E-mail address: oum-el-kheir.aktouf@lcis.grenoble-inp.fr

Peer-review under responsibility of KES International.
doi:10.1016/j.procs.2014.08.100

of diagnosis and testing. Finally, fault tolerance in large scale wireless sensor networks should be addressed as an overall problem instead of focusing on local fault tolerance for each WSN.

The aim of this work is to propose a fault fuzzy-ontology to be used in a web services based framework for LS-WSN fault tolerance. It intends to:

- facilitate testing and diagnosis of large-scale WSNs,
- use ontologies to describe the area of faults,
- design machines that can understand the situations of faults,
- decide on the types of diagnostics to be performed,
- support the self-test and diagnostics for fault tolerance,
- permit adaptive testing and diagnosis in dynamic LS-WSNs.

These points will be covered more in depth in the rest of this paper. This paper starts by giving the basics that have led to the development of fault fuzzy-ontology for large scale WSNs. Then, it presents the fundamentals of this ontology which is followed by a detailed example of service oriented architecture (SOA) supporting this ontology. The architecture is described from the fault tolerance side and does not address the network specifications like the topology, communication and so on.

2. An overview of service oriented architecture for sensor networks

Service Oriented Computing has emerged as a design paradigm for distributed execution environments. Services for these environments are generally considered as variants of Web services and have adopted their functioning principle. They are capable to communicate and provide modularity, scalability, composability and reusability to the architecture. Furthermore, services are used to overcome higher costs of development resulting from human intervention, evolution of services and dedicated design solutions.

In the area of wireless sensor networks, Web service-oriented architecture is proven to be an efficient design. It was proposed as an architecture model in [2], [3] and, [4]. Integration of sensors into service oriented architecture is usually done via software bridges that are managed by a gateway. These bridges provide access to different sensors from anywhere in the distributed environment. Accordingly, several network configurations might be imagined to design a large scale network using heterogeneous sensor protocols and gateways. An example of implementation is presented in [5].

Sensors are often used to provide information on the physical environment such as temperature, humidity, pollution and noise level, user location, etc. In fact, sensors fault tolerance is supposed secondary despite the fact that it is necessary to avoid side effects, mainly when applications are critical, for example, in the case of an erroneous inundation information sent by a sensor to the city management system or the lack of presence sensor information in the building elevator. In both cases, this will conduct to a wrong decision. In fact, a research effort was done in [6] to include fault tolerance capabilities into service oriented architectures. The authors present an interesting fault ontology engine for testing and evaluating the architecture but this research does not cover the domain of wireless sensor networks. Further research effort might be directed towards fault modeling using ontology to better examine its adequacy to the area of wireless sensor networks. The following section provides additional details on such a proposal.

3. Why a fault fuzzy-ontology?

There are two main reasons for developing fault fuzzy-ontology for large scale wireless sensor networks. The first one is related to the choice of the ontology and the second one concerns the use of fuzzy logic within this ontology.

The choice of ontology in WSNs has been widely defended in the literature but the notion of faults is treated from the security point of view and not as a fault tolerance problem. The authors in [2] addressed the challenge in designing and managing complex sensor networks over heterogeneous communication infrastructures and presented ontology as efficient tools to describe the assets and services deployed in a sensor network infrastructure. Many ontologies have been proposed for sensor networks. The well-known ontologies were deeply studied in [7] and [8]. It results that semantic representation for sensors uses descriptions of sensors, networks and domain concepts to aid in searching,

querying and managing the network and data. Each ontology depends on its description capabilities and the purpose for which it has been developed, for example adaptive and heterogeneous networks support. However, no one of these ontologies covers the fault domain of wireless sensor networks. Thus, the goal of this study is not to develop sensor network ontology but to complete existing ones by integrating the fault domain concepts. The final report of the W3C Semantic Sensor Network Incubator Group [1] recommends for organisations deploying sensors to monitor their assets to detect the possible occurrences of faults and the degradation in the quality of measurement over time (drift). To this end, the authors in[6] argue that ontologies are appropriate means for describing the fault, error and failure domains of systems. They can be used to enable machines to acquire the kind of knowledge necessary to develop their own strategies for testing and evaluating unfamiliar or large distributed systems. However, this latter work has not been addressed in the context of complex sensor networks and thus it does not take into account the infrastructure network particularities or the faults types' domain.

Regarding the use of fuzzy logic, Rolf Isermann[9] presented a study showing that fuzzy logic is a good method for fault diagnosis but it necessitates much design efforts. Since the ontology already needs some design efforts, it becomes interesting to pool both fuzzy logic and ontology together. In addition, observations in presence of faults in WSNs are uncertain. The use of approximate reasoning seems more adequate.

The development of a fault fuzzy-ontology for LS-WSNs will permit an interactive control of fault tolerance management through reasoning based diagnosis mechanism's and automated and composed testing procedures. The diagnostics procedure is based on the observed symptoms, expert knowledge and monitoring users experiences. In such system, the user is aware of types of faults the system is able to diagnosis. Diagnosis in fact is able to infer more information about detected faults. In addition, FFO (Faul Fuzzy-Ontology) for LS-WSNs will support heterogeneity of faults, devices and networks.

4. Fault fuzzy-ontology representation

An ontology is a model describing a domain concepts, their relationships and their properties. On the basis of this statement, fuzzy ontology extends the model with fuzzy annotations and fuzzy functions to support blurred and vague knwoledge.

In this study, a major part of fault domain knowledge is extracted from Abhinav study[10]. Then, a fuzzy fault-ontology (FFO) is developped using the fuzzy-ontology representation introduced by F. Bobillo and U. Straccia in[11] and implemented under Protégé version 4.1 [2] and Fuzzy OWL2 Protégé plug-in [3]. Figure 1 shows major concepts of sensors faults. The focus on these faults, rather than others like network faults, aims at showing by example the process of fault tolerance starting from ontology modeling to testing and diagnosis.

In the following, the description of the fault fuzzy-ontology is given in more details.

4.1. Fault categories

A fault is the cause of an error; an error is a part of the system state and can leads to a failure. A failure breaks down the system and should be repaired. Fault types can be classified into four categories: crash, measurement, transmission and byzantine faults. They are presented below.

Crash faults are common in wireless sensor networks. They occur when a sensor stops functioning or continue to operate without coming back to a stable state. The authors in[12] have studied this type of faults giving some reasons behind crash fault such as battery-limitation and high environmental temperature. This fault occurs at four levels: node[13], cluster-head[14] , sink[15] and gateway[16].

Measurement faults concern observable values sent by a sensor. These values can be invalid due to software and hardware failures, or to environmental conditions; for example, the deviation of an antenna. This type of fault occurs at the node level. Some methods were developed for measurement fault's recovery such as those based on testing in[17] and spatiotemporal correlation in[18].

[1] http://www.w3.org/2005/Incubator/ssn/wiki/Main_Page
[2] protege.standford.edu
[3] gaia.isti.cnr.it/ straccia/software/FuzzyOWL/index.html

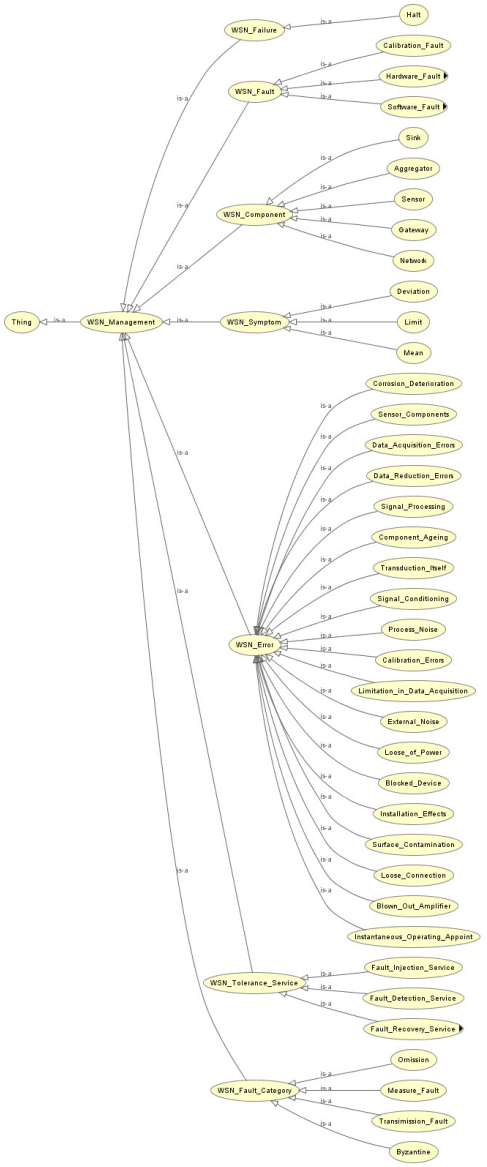

Fig. 1. Fault fuzzy-ontology structure.

Transmission faults are due to radio transmission failure or existence of obstacles for example. These faults occur at the network level. The authors in[19] propose multi-paths routing mechanism and those in[20] employ an intelligent acknowledgment technique.

Finally, *byzantine faults* are arbitrary faults happened usually when the behavior of the sensor becomes unpredictable. They occur at the node level. The work in[21] presents a method to detect and to repair software errors that can lead to byzantine faults.

These categories form four class concepts in the ontology. They are used to classifying faults and identifying their related errors and testing methods.

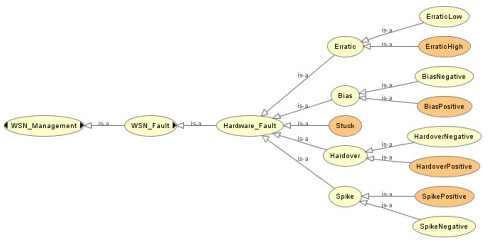

Fig. 2. Hardware faults.

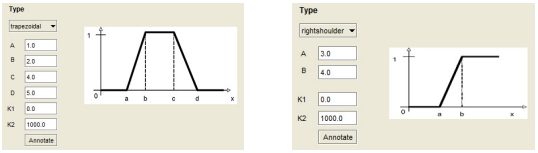

Fig. 3. (a)MLNBSupMLNS; (b) xHigh.

4.2. Faults conceptual vocabulary

In this paper, faults domain is restricted to some hardware and software faults according to the those mentioned by Abhinav in [10].

- Hardover refers to sudden change in the sensor output. The most common reason for Hardover fault includes deterioration of sensor components, total loss of power, and surface contamination.
- Bias is discrepancies between measured and true values of plant output and input. Bias fault may exist when some of the electric components of the sensor are ageing, resulting in mismatch between the electric circuits.
- Spike arises as a result of an unexpected disturbance in a hostile environment.
- Stuck is no variation in the sensor output, not even a sign of tolerable noise. Stuck fault would be reflected at the sensor output after the sensing line of sensor is fully blocked due to presence of impure materials.
- Erratic refers to the increase in deviation of the sensor reading. Erratic fault may exist due to ageing of the sensing elements or external noise
- Drift is a disturbance acting on the plant, which are normally zero and cause a shift in plant outputs independent of measured inputs.

Figure 2 presents these concepts structure in the hierarchy of the fault fuzzy-ontology.

4.3. Faults symptoms

Symptoms are indicators of faults. The sensor fault detection schema proposed in [10] is taken as an example to show how a fault fuzzy-ontology can be defined. This schema is based on statistical features extraction. Fault free data out of sensor is used to create threshold standards that are later used to detect sensor faults by comparing them with the one extracted out of online faulty sensor data. Main features are called *limit*, *mean* and *standard deviation*. For example, a Hardover indicator is positive if $x(i) > X_4$, where X_4 is the high parameter threshold of the feature *limit*. Figure 3 shows the fuzzification of this indicator. This function is called *xHigh* and it is defined in the FFO as a fuzzy datatype.

In the same way, spike indicator is negative if $vvk >= mk_{neg2}$ and $vvk < mk_{neg1}$ where mk_{neg1}, mk_{neg2} are both threshold for *mean* limit for negative spike and *mean* limit for negative spike. The fuzzy membership function corresponding to this axiome is presented in Figure 3. This function is called *MLNBSupMLNS* and it is defined in the FFO as a fuzzy datatype. The rest of faults are defined according to this principle.

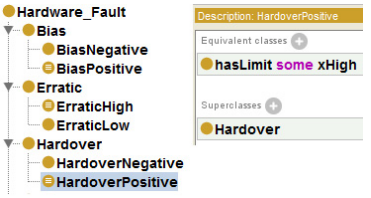

Fig. 4. HardoverPositive.

4.4. Faults fuzzification

The fuzzification of fault is described in the ontology using ”class equivalence” and fuzzy datatypes as it is shown by Figure 4 for HardoverPositive fault. Each fault has an equivalent class written in the form of a rule as follows.
HardoverPositive ≡ ∃hasLimit.xHigh
SpikePositive ≡ ∃hasMean.MLPBInfMLPS
ErraticHigh ≡ ∃hasDeviation.sHigh
Stuck ≡ ∃hasDeviation.sMean
BiasPositive ≡ ∃hasMean.MLPBSup

4.5. WSNs errors

WSN errors are defined as fuzzy concepts. They are defined in the ontology using the annotation property fuzzyLabel. For example, the *Component Ageing* is described as a weighted sum concepts. It means that *Component Ageing* error induces an ErraticHigh fault at 50% and a BiasPositive fault at 50%. For this example, the estimation is chosen arbitrary.

```
<fuzzyOwl2 fuzzyType="concept">
  <Concept type="weightedSum">
    <Concept type="weighted" value="0.5"
          base="ErraticHigh" />
    <Concept type="weighted" value="0.5"
          base="BiasPositive" />
  </Concept>
</fuzzyOwl2>
```

In the same way, *Surface contamination* is described by the following annotation.

```
<fuzzyOwl2 fuzzyType="concept">
  <Concept type="weightedSum">
    <Concept type="weighted" value="0.2"
          base="HardoverPositive" />
    <Concept type="weighted" value="0.8"
          base="Stuck" />
  </Concept>
</fuzzyOwl2>
```

5. FFO based diagnosis and testing for LS-WSN

Diagnosis allows fault location and confinement. It is complementary to testing which consists of the execution of test units to detect malfunction by comparing obtained results with expected ones. In the literature, the study in[22] shows that WSN diagnosis covers the node level despite of the behavior of other nodes in a large network.

In addition to a previous investigation done by Hamdan et *al.*[23], the proposed approach combines test and diagnosis to ensure dependability within WSNs. The idea consists of combining the usage of a fault fuzzy-ontology that

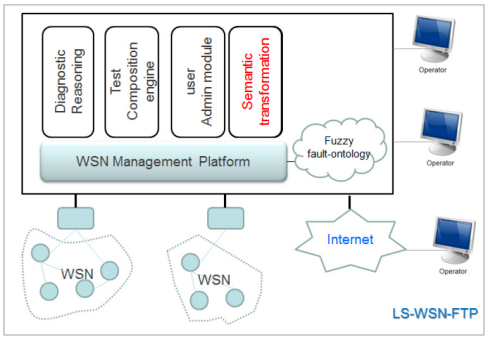

Fig. 5. LS-WSNs fault-tolerance platform.

describes each considered fault type with test execution and approximate-reasoning based diagnosis. The ontology includes task structure of diagnostic task and diagnostic process. It results in a web services based architecture that integrates different modules. They interact together to enhance dependability of WSNs by services provided through the WSN management interface. From the implementation view, large scale WSN is seen as a multiple number of gateways interconnected via Internet. Each one provides access to sensors, as well as gathering and disseminating data. LS-WSN gateway is implemented as an OSGi platform composed of three layers: -Sensors layer that cloud support one or multiple protocols like ZigBee, 6LowPan, etc.; - Middleware layer providing access to sensors. For each used protocol, a software bridge is developed and integrated into OSGi; - Fault diagnosis and testing management services (diagnosis reasoning service, test composition engine). In addition, a semantic transformation component is used to provide functions for ontology query and update. The general architecture is represented by Figure 5. The next two sections explain main components, namely diagnosis and test achievement.

5.1. Diagnosis

Fault diagnosis has been widely studied thus leading to many approaches. A short attention is given to these approches followed by a description of the proposed diagnosis process. Iman Saleh et *al.*[24], and Mahapatro et *al.*[25] have studied different schemes for fault-tolerance in wireless sensor networks. The study shows clearly the lack of generality in fault-tolerance schemes. There is no solution that covers all types of faults, and likewise there is no fault tolerance method for a particular fault type. In this context, Rolf Isermann[9], defines a fault diagnosis task as the determination of the fault type with as many details as possible. The author goes further by comparing fault diagnosis methods separated in two groups: classification and inference methods. The author pleads for a mixture method using both sources of information: Expert or domain (structured) knowledge on the one hand and measured data from fault experiments on the other hand. Particularly, fuzzy logic presents at most better or equal characteristics such as transparency, use of explicit knowledge and good performance, but it necessitates much design efforts.

According to[26], the ontology of faults provides a conceptual vocabulary to describe the scope of a diagnostic activity performed by a reasoning mechanism and enables the user to control the diagnostic system by relaxing diagnostic assumptions interactively. Moreover, the ontology spcifies the underlying assumptions of faults, thus the reasoning could explain the occurred fault in more details. For example, it is not enough to find a fault such as ”battery limitation of the node X” in a sensor network. The reasoning process is able to provide more information about the cause of this fault; for example, battery limitation of the node X is due to the oldness of the sensor.

In this paper, fuzzy diagnosis system is composed of set of rules described in fuzzyDL. fuzzyDL is a *description logic reasoner* supporting fuzzy logic and fuzzy Rough set reasoning. The authors in 1 proposed a parser translating a fuzzy-ontology into the languages supported by two fuzzy DL reasoners : fuzzy DL and DeLorean. Thus, FFO has to be translated into fuzzyDL language for reasoning. An example of the translation process is given hereafter.

- (define-fuzzy-concept MLPBInfMLPS triangular(0.0, 1000.0, 2.0, 2.5, 3.0))
- (define-fuzzy-concept MLNBSupMLNS trapezoidal(0.0, 1000.0, 0.0, 1.0, 4.0, 5.0))
- (define-fuzzy-concept xHigh right-shoulder(0.0, 1000.0, 3.0, 4.0))

- (define-concept Loose_of_Power (0.2 HardoverPositive))
- (define-concept Surface_Contamination (w-sum (0.2 HardoverPositive) (0.8 Stuck)))
- (define-concept Component_Ageing (w-sum (0.5 ErraticHigh) (0.5 BiasPositive)))
- (define-primitive-concept Sensor_Components WSN_Error)
- (define-primitive-concept Signal_Conditioning WSN_Error)
- (define-concept HardoverPositive (some hasLimit xHigh))
- (define-concept SpikePositive (some hasMean MLPBInfMLPS))

Moreover, fuzzyDL allow multiple query expressions but first it is necessary to make a defuzzification of the individual variables. Variables in this case are individual instances of symptoms limit, mean and deviation presented in section 4.3. Three defuzzification methods are allowed: largest, smallest or middle of the maxima method. The rules bellow show a defuzzification example.

- (instance m Mean 1.0)
- (instance d Deviation 1.0)
- (instance l Limit 1.0)
- (instance m (= hasMean 1.3) 1.0)
- (instance d (= hasDeviation 1.2) 1.0)
- (instance l (= hasLimit 3.6) 1.0)
- #
- (defuzzify-som? HardoverPositive l hasLimit)
- (defuzzify-som? ErraticHigh d hasDeviation)
- (defuzzify-som? Stuck d hasDeviation)
- (defuzzify-som? BiasPositive m hasMean)
- (defuzzify-som? SpikePositive m hasMean)

The final part of the diagnostic process is the reasoning part using appropriate queries. The following example shows how faults and corresponding causes could be determined using indicator values which have been defined individually in the FFO.

- (min-instance? l HardoverPositive)
- (max-instance? l HardoverPositive)
- #
- (min-instance? d ErraticHigh)
- (max-instance? d ErraticHigh)
- #
- (min-instance? d Component_Ageing)
- (max-instance? d Component_Ageing)
- (min-instance? l Surface_Contamination)
- (max-instance? l Surface_Contamination)
- (min-instance? d Surface_Contamination)
- (max-instance? d Surface_Contamination)

Figure 6 gives the results obtained for the complete scenario of the above examples. Among multiple conclusions, diagnosis indicates a stuck fault which has been likely caused by a surface contamination of the sensor components.

5.2. Test

Before explaining the testing part of this work, it is better to understand the testing goal to motivate this procedure. In the case of software systems, Miller[27] argues that the principle of testing is to affirm software quality by systematically exercising the software in carefully controlled circumstances. Lu Luo[28] mentioned that a good test is one that has a high probability of finding undiscovered errors. Regarding SOA architecture, testing of wireless sensor networks deals with specific methods for ascertaining the end-to-end quality of sensors in real conditions, i.e., with real data and under real (or simulated) circumstances. Testing could be done at each level of the wireless sensor networks:

```
Is l instance of HardoverPositive ? >= 0.6
Is l instance of HardoverPositive ? <= 0.6
Is m instance of BiasPositive ? >= 0.3
Is m instance of BiasPositive ? <= 0.3
Is d instance of ErraticHigh ? >= 0.0
Is d instance of ErraticHigh ? <= 0.0
Is d instance of Stuck ? >= 0.8
Is d instance of Stuck ? <= 0.8
Is d instance of SpikePositive ? >= 0.0
Is d instance of SpikePositive ? <= 1.0
Is d instance of Component_Ageing ? >= 0.0
Is d instance of Component_Ageing ? <= 0.5
Is m instance of Component_Ageing ? >= 0.15
Is m instance of Component_Ageing ? <= 0.65
Is l instance of Surface_Contamination ? >= 0.12
Is l instance of Surface_Contamination ? <= 0.92
Is d instance of Surface_Contamination ? >= 0.64
Is d instance of Surface_Contamination ? <= 0.84
```

Fig. 6. Scenario results.

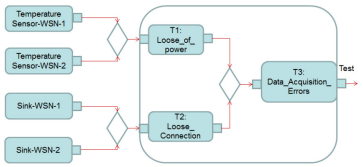

Fig. 7. Test composition example.

node, sink, gateway, and network level. A potential technique should allow composition and automation of testing procedures like those performed when two or more tested levels are involved. More generally, the test often includes both the functional/requirement specification of each level and non-functional quality attributes, such as reliability, security, and maintainability.

In this approach, the novel idea that is adopted for the testing phase consists in the creation of new tests using atomic testing procedures by composition like web service composition style. Let's consider a scalable distributed execution environment which employs high number of continuous running sensors which are used to provide information on the physical environment such as temperature, pollution and noise level, etc. Two temperature sensors are installed in different zones of a building. Each sensor, belongs to its local WSN. The temperature was regulated to have the same level in the two zones. A $Data\_Aquisition\_Errors$ test service is used to check for sensor faults that may be due to the presence an anomaly in the observed sensors reading. This test clould be done using services composition.

In this paper, the composition process is guided by a set of basic relation restrictions associated to services. The set of basic primitive relations is suitable to characterize the composition commitments. Relationships are expressed in the form: $R(S_r, S'_r)$, where R is the composition relation ($depend-on$, $replaced-by$). $Depend-on$ defines service needs for other services to achieve its task while $replaced-by$ expresses similarity of services capabilities i.e., a service can replace another one for a specific task. The test example presented by the figure 7 asserts that $Data\_Aquisition\_Errors$ service depends on two other services: $Loose\_of\_power$ and $Loose\_Connection$; formulated as follow: $Depend-on(Data\_Aquisition\_Errors, Loose\_of\_power)$, $Depend-on(Data\_Aquisition\_Errors, Loose\_Connection)$

This knowledge could be statically defined or specified by the administrator of the network. Accordingly, Service composition selects and composes services which satisfy the test query, i.e., the test for Data_Acquisition_Errors in this case. This knowledge is very elementary but additional relationship could be defined to make more complex composition processes.

6. Conclusion

Fault tolerance is strongly recommended for large scale wireless sensor network applications, particularly with the growing interest to the Internet of things and Smart Cities. This paper has addressed this requirement and motivated a

service oriented approach to build diagnosis and test services for wireless sensors. The proposal includes an ontology model to describe fault, error and failure domain and to allow machines developing their own strategies for testing and diagnosis. This work gave birth to a first prototype which is not yet totally integrated but reflects potential application scenarios. It is however sill difficult to evaluate the impact of this approach in real cases due to the lack of development achievement. However, benefits are always here to permit further research investigation.

References

1. Hughes, D., Thoelen, K., Horr, W., Matthys, N., Cid, J.D., Michiels, S., et al. Looci: a loosely-coupled component infrastructure for networked embedded systems. In: *Proceddings of International Conference on Advances in Mobile Computing and Multimedia (MoMM)*. 2009, .
2. Ibbotson, J., Gibson, C., Wright, J., Waggett, P., Zerfos, P., Szymanski, B., et al. Sensors as a service oriented architecture: Middleware for sensor networks. In: *The Sixth International Conference on Intelligent Environments (IE)*. 2010, p. 209–214.
3. Ibbotson, J., Gibson, C., Geyik, S., Szymanski, B.K., Mott, D.. Model-driven soa for sensor networks. In: *Proceedings SPIE 8047, Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR II*. 2011, .
4. Klopfer, M.. *SANY, an open service architecture for sensor networks*. SANY Consortium; 2009.
5. Benazzouz, Y., Munilla, C., Gunalp, O., Gallissot, M., Gurgen, L.. Sharing user iot devices in the cloud. In: *IEEE World Forum on Internet of Things (WF-IoT)*. 2014, p. 373–374.
6. Gwynne, B.. Developing a fault ontology engine for the testing and evaluation of service-oriented architectures. *Sheffield and York the white rose grid e-science centre* 2006;:1–3.
7. Bell, D., Heravi, B.R., Lycett, M.. Sensory semantic user interfaces (sensui). In: *2nd International Workshop on Semantic Sensor Networks*. Washington; 2009, .
8. Compton, M., Henson, C., Lefort, L.,, , Neuhaus, H.. A survey of the semantic specification of sensors. In: *2nd International Workshop on Semantic Sensor Networks*. CEUR-WS; 2009, p. 17–32.
9. Isermann, R.. *Fault-Diagnosis Systems and An Introduction from Fault Detection to Fault Tolerance*. Springer-Verlag; 2006.
10. Abhinav, A.. *Sensors failure mode detection and self-validation*. Master's thesis; Master's thesis, Graduate school of the University of Cincinnati, India; 2008.
11. Bobillo, F., Straccia, U.. Fuzzy ontology representation using owl 2. *International Journal of approxiamte reasoning* 2011;:1073–1094.
12. de Souza, L.M.S., Vogt, H., Beigl, M.. A survey on fault tolerance in wireless sensor networks. In: *SAP Research*. Karlsruhe University, Germany; 2007, .
13. A.Taleb, , Pradhan, D.K., T.Kocak, . A technique to identify and substitute faulty nodes in wireless sensor networks. In: *Third International Conference on Sensor Technologies and Applications*. 2009, .
14. Gupta, G., Younis, M.. Fault-tolerant clustering of wireless sensor networks. *Wireless communications and Networking* 2003;:1579–1584.
15. Saleh, I., Agbaria, A.,, , Eltoweissy, M.. Network fault tolerance in networked sensor systems. In: *2nd ACM/SIGMOBILE Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks DIWANS06*. California; 2006, .
16. Gupta, G., Younis, M.. Fault-tolerant clustering of wireless sensor networks. *Wireless Communications and Networking* 2003;:1579–1584.
17. Koushanfar, F., Potkonjak, M., Sangiovanni-Vincentelli, A.. On-line fault detection of sensor measurements. In: *Proceedings of IEEE Sensors*; vol. 2. 2003, p. 974–979.
18. Chen, , J., , Kher, S., Somani, A.. Distributed fault detection of wireless sensor networks. In: *Proceedings of the workshop on Dependability issues in wireless ad hoc networks and sensor networks*. ACM: Los Angeles, CA, USA; 2006, .
19. Boukerche, A., Werner, R., Pazzi, N.. A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications. In: *Proceedings of MSWiM'04*. 2004, .
20. Lazaridis, I., Han, Q., Mehrotra, S.,, , Venkatasubramanian, N.. Fault tolerant evaluation of continuous selection queries over sensor data. *International Journal of Distributed Sensor Networks* 2009;:338–360.
21. Herbert, D., Y.Lu, , Bagchi, S., Li, Z.. Detection and repair of software errors in hierarchical sensor networks. In: *Proceedings of the IEEE International Conference on Sensor Networks Ubiquitous, and Trustworthy Computing*. 2006, p. 403–410.
22. Rodrigues, A., Camilo, T., Silva, J., Boavida, F.. Diagnostic tools for wireless sensor networks: A comparative survey. *Journal of Network and Systems Management* 2013;**21**(3):408–452.
23. Hamdan, D., Aktouf, O., Parissis, I., Hijazi, A., El Hassan, B.. Test and diagnosis of wireless sensor networks applications. In: *2013 World Congress on Computer and Information Technology (WCCIT)*. 2013, p. 1–7.
24. Saleh, I., El-Sayed, H., Eltoweissy, M.. A fault tolerance management framework forwireless sensor networks. In: *Innovations in Information Technology*. 2006, p. 1–5.
25. Mahapatro, A., Khilar, P.. Fault diagnosis in wireless sensor networks: A survey. *Communications Surveys Tutorials, IEEE* 2013; **15**(4):2000–2026.
26. Kitamura, Y., Mizoguchi, R.. An ontological analysis of fault process and category of faults. In: *Proceedings of the Tenth International Workshop on Principles of Diagnosis*. 1999, p. 118–128.
27. Miller, E.F.. Tutorial: Software testing & validation techniques. 1981, p. 4–16.
28. Luo, L.. Software testing techniques. Tech. Rep.; Institute for Software Research International, Carnegie Mellon University, Pittsburgh, PA15232, USA; 2002.