
1 Jie SUN, 1 Gil DE SOUSA, 1 Catherine ROUSSEY, 1 Jean-Pierre CHANET, 1 François PINET and 2 Kun-Mean HOU

1 UR TSCF, Irstea 9 avenue Blaise Pascal CS 20085, F-63178 Aubière, France
2 LIMOS, rue de la Chebarde TSA 60125 - CS 60026, F-63178 Aubière cedex, France

Tel.: +33 4 73 44 06 00, fax: +33 4 73 44 06 96
Tel.: +33 4 73 40 74 40, fax: +33 4 73 40 53 56
E-mail: jie.sun@irstea.fr, gil.de-sousa@irstea.fr, catherine.roussey@irstea.fr, jean-pierre.chanet@irstea.fr, francois.pinet@irstea.fr, kun-mean.hou@isima.fr

Received: 25 July 2016 / Accepted: 30 September 2016 / Published: 30 October 2016

Abstract: Henceforth, new generations of Wireless Sensor Networks (WSN) have to be able to adapt their behavior to collect, from the study phenomenon, quality data for long periods of time. We have thus proposed a new formalization for the design and the implementation of context-aware systems relying on a WSN for the data collection. To illustrate this proposal, we also present an environmental use case: the study of flood events in a watershed. In this paper, we detail the simulation tool that we have developed in order to implement our model. We simulate several scenarios of context-aware systems to monitor a watershed. The data used for the simulation are the observation data of the French Orgeval watershed.

Keywords: Context-aware system, Formalization, Architecture, Wireless sensor network, Internet of things, Environment, Phenomenon.

1. Introduction

The acquisition of heterogeneous data is important in the era of Internet of Things (IoT) and Big Data that is just starting. These two research topics have application in numerous fields: industry, “smart home-smart care”, e-agriculture, environment, etc. Wireless Sensor Network (WSN) technology is now viewed as part of the IoT [1]. The increased use of WSN envisioned at the beginning of the 2000’s [2], is now a reality as shown, for example, in environment [3] and agriculture [4]. In these applications, a WSN collects natural phenomenon observations (temperature, humidity, etc.) and sends them to a context-aware system, which may propose adaptation actions based on context. A context-aware system needs also to take into account the energy of the wireless sensors which provide him with data. Indeed, despite steady progress in hardware (the development of low energy communication modules for example), a wireless sensor still has scare resources. It is the case for “scalar” WSN and even more for Wireless Multimedia Sensor Networks (WMSN) [5]. Thus, to better use these limited resources, all the components that are part of the data acquisition process have to work together in a cooperative way, from the component that collects raw data to the one that provides indicators to end users. Generally, these components
are the wireless sensors, the gateway(s) and the remote Decision Support System (DSS). The acquisition and transmission frequencies required by the DSS, through the gateway, have to be consistent with the energy available at the level of the wireless sensors. For some alert applications such as fire prevention, data transmission is sometimes more important than the “survival” of a node of the network. Thus, all the components implied in the data acquisition process have to adapt their behaviors to the context in order to achieve the best performances. The combination of the common decisions and actions is the issue addressed in this paper. More precisely, we propose a formalization to define high level context which, integrated into an adaptive context-aware system, will be used to reduce the number of exchanged communication packets. Our formalization proposes different reasoning steps in order to build the high level context. This formalization was already presented in [11]. Compared to our previous paper, this paper presents in more details the different processes composing a flood context-aware system. Based on these processes, we propose two types of system. We also present new qualitative results of our simulation scenarios also based on the monitoring of a watershed. Section 2 presents the main existing concepts related to context-aware systems. Section 3 explains our formalization of context in order to build any context system based on WSN. Section 4 shows its application with the design of context-aware systems dedicated to a complex environmental use case. Section 5 presents our simulation platform that implements our formalization. Section 6 compares several context-aware systems through different metrics. Section 7 describes different context systems developed for the same purpose. The last section concludes this article.

2. Main Processes involved in Context-aware System

One of the most known and accepted definitions of context is given by [6] as: “Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves”. As indicated in this definition, context is focused on one entity. Several contexts can be defined, for example, the context of the user, the context of the device running the application, etc. As explained in [7], different categories of context exist. Low level context corresponds to the raw data acquired by sensors or static data provided by users. High level context is computed from the low level one, with more informative data associated to the application and the user. Fig. 1 presents the processes associated to an adaptive context-aware system when data are collected by a WSN. It could also be applied to sensor networks or other systems that generate raw data.

In an adaptive context-aware system, different processes are required:

- Context acquisition: collecting raw data and metadata that are useful to build the context.
- Context modeling: organization of the collected data through a specific context data model. The process provides an interpretation to each raw data. For example, the value 24 becomes the measurement of the outdoor temperature in degree Celsius. This process builds the low level context. This process is also called annotation or tagging [7].
- Context reasoning: the high level context is computed or inferred from the low level one. This

![Fig. 1. Context cycle of an adaptive context-aware system based on WSN.](image)
process can imply different approaches based on machine learning [8] or rule engine [9].

- Context distribution: diffusion of the high level context to the different consumers, for example, the end user or any system component that wants to be aware of the context.
- Context adaptation: actions to adapt any system components according to context changes.

3. Formalization of Context-aware System

The work in [10] defines the concept of entity “state” as “a qualitative data which changes over times (summarizing a set of information)”. In [11], we propose our definition of “context” as a set of entities characterized by their state, plus all information that can help to derive any state changes of these entities. This definition allows us to consider two classes of entities:

1) Observable entity: entity that is directly observed by sensors.
2) Entity of interest: entity whose characterization is obtained from one or many other entities and required by the application.

The proposal is to divide the reasoning process into several reasoning steps to create the high level as illustrated in Fig. 2. Rule-based reasoning is often used to deduce high-level context [7]. The work in [11] is the first to promote the division of rules into several reasoning steps in order to make the management of rules easier and let envision an integration of more complex decision processes directly in a node in order to reduce data exchanges. Indeed, the state of an entity of interest cannot be acquired directly based on the low level context. Two levels of reasoning are presented to build the high level context:

- The low level context contains the sensor measurements stored in the context data model. The state of observable entities is inferred from the low level context as indicated by the dotted arc in Fig. 2. At this stage, the high level context contains the state of observable entities.
- The state of an entity of interest is inferred from the state of other entities. The high level context is enriched by the state of the entity of interest.

To illustrate this formalization, we take a short example based on a wireless sensor management application. In this case study, we consider a wireless sensor as an entity of interest where one of its associated observable entities is its power supply (a battery). This observable entity is associated to a sensor that measures a charge/an energy level as a raw data observation. These raw data are part of the low-level context. Based on the capacity and charge values, we deduce the percentage of energy remaining in the battery. This percentage is represented by the variable Energy. Fig. 3 presents an example of finite-state machine used to deduce the energy state (high, middle or low) of the battery which is included in the high level context.

![Fig. 3. Example of wireless sensor energy finite-state machine.](image)

The goal is to transform quantitative data into qualitative one in order to simplify the reasoning process.

The following sections present the context-aware system and describe its associated context and reasoning processes.

4. Context-aware System Dedicated to Environmental Phenomenon

The considered environmental application is a watershed monitoring system which is able to send alert about flood risks. As shown in Fig. 4, the application uses a WSN for data acquisition. This network is composed of wireless sensors, called “Water flow nodes”, each equipped with a stream gauge measuring the water flow rate. One of these wireless sensors is located on the outlet of the watershed. The network contains also “PrecipitationNodes” measuring the precipitation quantity. All the measurements are sent to a DSS. This DSS deduces the risk of the occurrence of a flood and sends it to end users. One of our assumptions is that the WSN has a star topology: each node communicates directly with the DSS; we do not introduce routing protocol constraints at this step.
In the application, we define four entities:

1) The Precipitation entity which is an observable entity.
   Its state is calculated from the data collected by the “Precipitation nodes” located at different points of the watershed. The Precipitation entity (P) has two states: high and low.

2) The WaterCourse entity which is an observable entity.
   Its state is calculated from the data collected by the “water flow nodes” located at different points of the tributary stream (water courses). The WaterCourse (W) entity has two states: high and low related to the water flow rate.

3) The Outlet entity which is also an observable entity.
   Its state is calculated from the data collected by the “water flow node” located on the outlet of the watershed. The Outlet entity (O) has two states: high and low related to the water flow rate.

4) The Flood entity is the entity of interest of the application.
   The flood entity is not an observable entity but its state depends on the states of all the observable entities. The Flood entity has four states “Normal”, “Rain”, “Risk”, and “Flood”. “Normal” state means there is no risk. “Rain” state means that the watershed has received lot of precipitations. “Risk” state means that flood is coming because some tributaries are overflowing. “Flood” state means that the flood is there, the main river is overflowing. Application users want to know as soon as possible when a risk state is reached.

   All the measurements are stored in the context data model in order to build the low level context.

   Several processing steps are necessary to build the high level context of the Flood entity from the raw data:

   Each “PrecipitationNode” measures the rainfall amount that fall down during the frequency acquisition time period. Then, it computes the rainfall amount that fall down during the last 24 hours. This value is sent to the DSS. The DSS receives some precipitation measurements from the various “PrecipitationNodes”. Thus, these measurements are aggregated. The DSS sums all the last measurements received from each Precipitation node. One threshold should be set on the aggregation value in order to determine when the Precipitation entity moves from the low to the high state and vice versa.

   Some “Water flow node” measures the flow rate of tributaries of the watershed. We rename them “WaterCourseNode”. They send to the DSS their last measurement. The DSS computes the slope between the last two received measurements from each “WaterCourseNode”. All the slopes obtained from “WaterCourseNodes” are aggregated. The DSS determines their max value. One threshold should be set on the aggregation value in order to determine when the WaterCourse entity moves from the low to the high state and vice versa.

   The “OutletNode” is a “water flow node” that measures the flow rate of the outlet and sends it to the DSS. The DSS computes the slope based on the two last measurements received from this node. Based on the slope value, one threshold should be set in order to determine when the Outlet entity moves from the low to the high state and vice versa.

   Fig. 5 presents the finite-state machine that deduces the state of the Flood entity from the states of the three other observable entities. This diagram follows every step of the emergence of a flood. Usually, when a flood event occurs, the Flood entity will move from the “Normal (F1)” state to the “Rain (F2)” one, proceed to the “Risk (F3)” one and finish with the “Flood (F4)” one.
The node measurements are not taken into account in the same way depending on the node type. “PrecipitationNodes” send to the DSS an aggregation of their measurements i.e., the rainfall amount of the last 24 hours. It is not the case for “Water flow nodes” that send their last measurement. For the “Water flow nodes”, whatever the acquisition frequency is, the sent measurement is fixed by the communication one. When the communication frequency is smaller than the acquisition one, some measurements are not taken into account by the DSS. Thus, we propose a second way to process the measurements of “Water flow nodes”. For each communication interval, instead of sending its last water flow rate value, they compute the slopes between two successive measurements. Then, they calculate the maximum slope and send it to the DSS. This modification is applied to the “WaterCourseNodes” and the “OutletNode”.

The first configuration of the system is called base configuration, the second is called aggregated configuration.

5. Simulation

First, we present the components of our simulation tool based on JADE. Then, we describe the configuration of our simulation tool based on a real data set that we used to run the simulation.

5.1. Simulation Platform based on JADE

To implement our formalization, we extend the simulation tools based on the multi-agent system JADE (Java Agent Development Framework) [13] as introduced in [10]. Three main features of JADE are:

1) Agent communication: exchange of messages between agents.
2) Message content modeling by ontologies: use of ontologies to model the exchanged message contents between agents.
3) Integration with other tools: possible use of tools like Jess rule engine [14] as a decision component of an agent.

However, current implementations based on JADE do not use different levels of reasoning. The work in [15, 16] only implement message exchanges between agents. It does not take into account the content of message modeling and other tools like the integration of Jess rule engine. In our simulation, we implement the features from Fig. 6.
build the low level context from collected and aggregated measurements (e.g., rainfall amount per 24 h, rainfall amount, water flow rate, max slope of water flow rate). Then, we can use Jess engine and a set of rules to infer the states of observable entities (e.g., Precipitation, WaterCourse and Outlet) and build the high level context. The state of the entity of interest (e.g., Flood) is also inferred, by using Jess and another set of rules, from the state of the observable entities. Thus, the high level context is enriched. As mentioned in [7], the context modeling is often based on ontologies.

5.2. JADE Ontology

Ontologies are defined by [17] as a formal explicit specification of a shared conceptualization. According to World Wide Web Consortium (W3C), ontologies are vocabularies that define the concepts and relationships used to describe and represent an area of concern. Thus, ontologies provide meaning to data (as data models do). Our ontology is based on the Semantic Sensor Network ontology (SSN) proposed by the W3C [18]. This ontology is a nucleus on which other ontologies can be connected, in order to develop a full context data model. The main concepts of SSN ontology that we reused are: “Sensor”, “FeatureOfInterest”, “Property”, and “Observation”. Our observable entities or entity of interest are defined in SSN Ontology as “FeatureOfInterest”. To complete our ontology, we reuse some entities which come from some dedicated ontologies: the Climate and Forecast ontology [19] and Irstea Hydro Ontology, based on the work in [20]. To describe the state of our entities, we reuse the ontology proposed in [10].

The organization of our main entities is given by the Stimulus-Sensor-Observation design pattern of SSN ontology. As shown in Fig. 7:

- A “FeatureOfInterest” may be associated to several “Properties”. For example, the “FeatureOfInterest” Precipitation has two “Properties”: rainfall_amount and rainfall_amount_per_24h.
- “Property” may be linked to several “FeatureOfInterest”.
- “Observation” is linked to all entities involved in a sensing process: “FeatureOfInterest”, “Sensor”, and “Property”.
- The measurement value is represented by the “SensorOutput” entity.
- “Sensor” is linked to “Property”.

In our JADE ontology, we have simplified this organization, due to the fact that we reuse only parts of SSN. Some links may be discarded.

The Fig. 8 presents our JADE ontology.

Our ontology implements some constraints:

A “FeatureOfInterest” (e.g., Precipitation) may be associated to several “Properties” (e.g., rainfall_amount and rainfall_amount_per_24h). A “Property” (e.g., waterflow_rate) may be associated to several “FeatureOfInterest” (e.g. “WaterCourse” and “Outlet”). So the cardinality of the association between “FeatureOfInterest” class and “Property” class is N: N.

“Sensor” class is specialized by different subClass (e.g., “PrecipitationNode”, “WaterCourseNode” and “OutletNode”). Each sensor class (e.g., “PrecipitationNode”) is dedicated to one “FeatureOfInterest” instance (“Precipitation”) and can measure the value of different “Properties” (e.g., rainfall_amount, rainfall_amount_per_24h).

A “Property” (e.g., waterflow_rate) may be associated to several “Sensors” (e.g. “WaterCourseNode” and “OutletNode”). So the cardinality of association between “Sensor” class and “Property” class is also N: N.
An “Observation” is associated to only one “FeatureOfInterest”, one “Property” and one “Sensor”. So the cardinality of the association between “Observation” class and these classes are all 1:1. “Observation” is indeed the structure that stores all the information about a measurement (which sensor, which property, which value).

In our simulation, there is only one DSS but many “Sensors”. When DSS sends new “communicationFrequency” to “Sensors”, “Sensors” use “FrequencyAdapt” which is an “AgentAction” to switch to this new “communicationFrequency”.

5.3. Jess Engine

The main advantage of our modeling of context is to split the reasoning process in several steps that help the management of this process. The reasoning process is dedicated to the construction of the high level context from the low level one. The first step of the high level context building is to infer the states of observable entities. From those states, a second reasoning step will be to infer the states of the entities of interest.

Concerning the reasoning process, we use the Jess rule-based engine [14] as indicated above. Jess is also implemented in Java. We define several rule sets. Some are dedicated to infer the state of observable entity based on predefined thresholds and aggregation values. Others are dedicated to infer the state of the Flood entity. For example, the rule presented in Fig. 9 deduces the state “Normal (F1)” of the Flood entity from states of observable entities.

(defrule floodState_NormalF1
  (declare (10))
  ?p <- (Precipitation {state == low})
  ?w <- (WaterCourse {state == low})
  ?o <- (Outlet {state == low})
  ?f <- (Flood)
  -> (modify ?f (state f1)))

Fig. 9. Rule deducing the Normal state of Flood entity.

5.4. Orgeval Watershed

Our simulation are based on real data, provided by the French Orgeval watershed [21] managed by the Ifrste institute.

From the available data, we select the data of 3 weather stations of a sub-basin of Orgeval. The outlet of this sub-basin is monitored by a stream gauge, and two other stream gauges monitored the tributaries.

In JADE system, we have created one “PrecipitationNode” agent for each weather station. These nodes will provide rainfall amount measurements. These sensors also provide the total rainfall amount per 24 hours measurements.

We also create one agent per stream gauges. Two agents are “WaterCourseNodes” that provide water
flow rate of the tributaries of the watershed. One agent is an “OutletNode” that provides the flow rate of the outlet.

We also use one agent to represent the DSS.

5.5. Communication Frequency

In our simulation, we fixed the acquisition frequency of nodes at one measurement per minute.

Table 1. Table of transmission frequency based on Flood entity state.

<table>
<thead>
<tr>
<th>Flood State</th>
<th>NORMAL</th>
<th>RAIN</th>
<th>RISK</th>
<th>FLOOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication Frequency</td>
<td>1/3* acqui.freq</td>
<td>1/2* acqui.freq</td>
<td>acqui.freq</td>
<td>acqui.freq</td>
</tr>
</tbody>
</table>

When the Flood entity reaches the “Risk (F3)” or “Flood (F4)” state, all the nodes will have to communicate with the maximum frequency which is equal to the acquisition frequency (one transmission per minute).

5.6. Threshold values of Observable Entities

For the calculation of the thresholds of the Observable entities, we use the Orgeval watershed data of the whole year 2007. All the data are acquired with a frequency of one minute. We set by experimentation all the thresholds as shown in Table 2. Then, for the simulation, we take the data of the February month of 2008.

Table 2. Table of Observable entities thresholds.

<table>
<thead>
<tr>
<th>Observable Entities thresholds</th>
<th>Precipitation</th>
<th>WaterCourse</th>
<th>Outlet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>10 mm</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

6. Experimentations

The purpose of our experimentation is to test the feasibility of our formalization by implementing different flood context-aware systems. Moreover, we want to demonstrate that the adaptation of the behavior of the WSN based on the Flood entity state will improve the life time of the global system without deteriorating the Quality of Service (QoS) of the system; the system should be able to alert as soon as possible in the case of a flood risk.

6.1. Experimentation Settings

To simulate the network, we implement three systems. All these systems use the same WSN, composed of 3 “PrecipitationNodes” that collect “rainfall amount per 24h”, 2 “WaterCourseNodes” that collect water flow rate and one “OutletNode” for the water flow rates acquisition in the outlet.

At the beginning, the communication frequency is also set to one transmission per minute. Depending on the type of context adaptive system, the communication frequency will evolve.

Table 1 presents the value of the communication frequency based on the Flood state. That is to say the communication frequency depends on the state of the Flood entity. All nodes will have the same communication frequency that may change over time.

Table 2 presents the value of the observable entities thresholds used in our simulation. The thresholds are set to 10 mm for precipitation, 2 for water flow rate, and 2 for outlet.

The first system is the baseline system that computes the state of the Flood entity using all the measurements acquired by the nodes. That is to say that the baseline system uses the basic configuration: “PrecipitationNodes” send their measurements “rainfall amount per 24h”, the “Water flow nodes” send all their collected water flow rates. For this latter, the DSS computes the slopes. Moreover, the acquisition and communication frequencies of all the nodes are set to one minute. As shown in Fig. 10, in this system, the DSS receives all the measurements and performs all the reasoning processes. Then, it sends the “Flood” entity state to end users. In this system, there is no adaptation and all the measurements acquired by the nodes are used by the DSS to compute the state of the “Flood” entity.

Fig. 10 presents the message workflow between different agents and the four processes involved in the system: context acquisition, context modeling, context reasoning and context distribution. The three different types of wireless sensors are represented by one agent.

We also implement two adaptive systems. The system 2 has the basic configuration and the system 3 has a aggregated configuration for the “Water flow nodes” data processing.

The two adaptive systems implemented the message workflow and process sequence presented in Fig. 11. The nodes send their measurement to the DSS. In system 2 and 3 “PrecipitationNodes” send their measurements “rainfall amount per 24h”. In system 2, the “Water flow nodes” send their last collected water flow rates. For this latter, the DSS computes the slopes. In system 3, the “Water flow nodes” computes the maximum slope from two successive collected water flow rate during communication intervals.

The DSS agent processes the context modeling in order to build the low level context. It infers the high level context from the low level one using Jess rule engine. The adaptation decision is implemented by the DSS. It deduces a new communication frequency for
the wireless sensors based on the “Flood” entity state. For this system, the communication frequency is the same for all the wireless sensors. Then, it sends a message to each node to ask them to change their communication frequency. Thus, the communication frequency of each node is modulated as presented in Table 1. The acquisition frequency of each node is constant and set to one minute.

Using our simulation architecture, we have compared these three systems using the data collected on a watershed on February 2008 provided by [22].

![Fig. 10. Sequence diagram of the Flood context-aware system (system 1).](image1)

![Fig. 11. Sequence diagram of the Flood adaptive context-aware system (system 2).](image2)

### 6.2. Results

First, we compare all these systems at the level of the energy consumption and then at the level of the flood management in a watershed QoS.

Total amount of exchanged communication packets:

First, we compare all these systems at the level of the total amount of exchanged communication packets. In most wireless applications, communication is the activity that consumes the most energy [23]. Thus, more a system exchanges packets, the less energy the WSN will have. Thus, the lifetime of the WSN will decrease accordingly.

Table 3 shows the total number of messages exchanged in February 2008. The nodes of the system 1 have transmitted more than 250,000 packets. With the systems 2 and 3, the total number of
transmitted packets is reduced to less than 100,000 packets. The system 3 sends more messages than system 2 because it detects sooner flood or risk states. Thus, it is more reactive in comparison with system 2 in terms of communication frequency adaptation.

Table 3. Total number of exchanged communication packets.

<table>
<thead>
<tr>
<th>Systems</th>
<th>NB of Exchanged packets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>System1</td>
</tr>
<tr>
<td>01/02/2008 00:00 to 01/03/2008 00:00</td>
<td>250560</td>
</tr>
</tbody>
</table>

So, if we consider our three systems, all the adaptive systems are the most efficient compared to the basic one at the level of energy management. Thus adapting the communication of the observed phenomenon is a good solution to improve the lifetime of the WSN.

Flood Phenomenon states changes:

We also compare these three systems in terms of flood phenomenon states changes. Fig. 12 presents the state change of the Flood State deduced by the three systems. Remember that the system 1, the baseline, uses all the measurement acquired by the sensors to compute the state of all entities.

From Fig. 12, we can see that the three curves associated to each system are very close. They seem similar due to the fact that the time scale is the day.

As shown on Table 4, the total number of flood phenomenon state changes in Feb, 2008 is the same for the three systems. But if the scale is enlarge to the minute, some timestamp differences will appear as shown in Table 5 where the time stamps of flood phenomenon state changes in Feb., 2008 is presented.

![Flood phenomenon states in Orgeval](image)

Table 4. Total number of flood phenomenon state changes in Feb, 2008.

<table>
<thead>
<tr>
<th>State changes</th>
<th>Normal to Rain</th>
<th>Normal to Risk</th>
<th>Rain to Risk</th>
<th>Rain to Flood</th>
<th>Risk to Flood</th>
<th>Flood to Risk</th>
<th>Flood to Rain</th>
<th>Risk to Rain</th>
<th>Rain to Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>nb of state changes</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>System1</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>System2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>System3</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>
Each line of Table 5 represents a peak of the Fig. 12. The state changes are presented in chronological order.

Compared with system 1, there are some delays for the state changes in systems 2 and 3. The system 2 has 9 states changes that are delayed. The system 3 has only 7 states changes delayed. The delayed timestamps are presented in bold format in Table 5.

In Fig. 13, an example of state changes of systems 1 and 2 is provided illustrating the delay. We take only into account the messages sent by the “WaterCourseNodes”. At the time “21:59”, the systems 1 and 2 deduce that the Flood entity is in the “Rain” state. Based on Table 1, the communication frequency of system 2 is one communication every two minutes. So the wireless sensors of the system 2 will only send their next measurements to the DSS at the time “22:01”. At time “22:01”, system 1 detects a state change from Rain to Risk. This change has also to be theoretically detected by system 2. But, the water flow rate values of system 2 nodes sent at this time “22:01” will generate an aggregated slope under the “WaterCourse” threshold. We will have to wait to the next communication, at time “22:03”, to switch from Rain to Risk state in system 2.

System 3 solves this kind of problem as shown in Fig. 14. Similarly, at time “21:59”, system 3 deduces that the Flood entity is in the “Rain” state. Also, based

### Table 5. Timestamps of flood phenomenon state changes in Feb, 2008.

<table>
<thead>
<tr>
<th>state changes</th>
<th>Normal to Rain</th>
<th>Normal to Risk</th>
<th>Rain to Risk</th>
<th>Rain to Flood</th>
<th>Flood to Risk</th>
<th>Flood to Rain</th>
<th>Risk to Rain</th>
<th>Rain to Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>timestamp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01/02 01:41</td>
<td>01/02 04:03</td>
<td>01/02 18:01</td>
<td>01/02 18:41</td>
<td>02/02 02:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>04/02 06:30</td>
<td>05/02 05:24</td>
<td>06/02 04:27</td>
<td>05/02 07:41</td>
<td>05/02 02:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System 2</td>
<td>01/02 01:42</td>
<td>01/02 04:03</td>
<td>01/02 18:01</td>
<td>01/02 18:41</td>
<td>02/02 02:16</td>
<td>02/02 02:16</td>
<td>05/02 14:09</td>
<td></td>
</tr>
<tr>
<td>04/02 06:30</td>
<td>05/02 05:25</td>
<td>05/02 07:43</td>
<td>05/02 02:16</td>
<td>05/02 12:21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>05/02 05:25</td>
<td>05/02 05:22:03</td>
<td>05/02 07:41</td>
<td>05/02 02:16</td>
<td>05/02 12:21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>06/02 04:27</td>
<td>06/02 05:11</td>
<td>06/02 05:11</td>
<td>06/02 05:11</td>
<td>06/02 05:11</td>
<td>07/02 05:19</td>
<td>07/02 05:19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20/02 10:35</td>
<td>21/02 05:35</td>
<td>21/02 05:35</td>
<td>21/02 05:35</td>
<td>21/02 05:35</td>
<td>21/02 05:35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26/02 15:35</td>
<td>27/02 14:25</td>
<td>27/02 14:25</td>
<td>27/02 14:25</td>
<td>27/02 14:25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System 3</td>
<td>01/02 01:42</td>
<td>01/02 04:03</td>
<td>01/02 18:01</td>
<td>01/02 18:41</td>
<td>02/02 02:16</td>
<td>02/02 02:16</td>
<td>05/02 14:09</td>
<td></td>
</tr>
<tr>
<td>04/02 06:30</td>
<td>05/02 05:25</td>
<td>05/02 07:41</td>
<td>05/02 02:16</td>
<td>05/02 12:21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>05/02 05:25</td>
<td>05/02 05:22:01</td>
<td>05/02 07:41</td>
<td>05/02 02:16</td>
<td>05/02 12:21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>06/02 04:27</td>
<td>06/02 05:11</td>
<td>06/02 05:11</td>
<td>06/02 05:11</td>
<td>06/02 05:11</td>
<td>07/02 05:19</td>
<td>07/02 05:19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20/02 10:37</td>
<td>21/02 05:35</td>
<td>21/02 05:35</td>
<td>21/02 05:35</td>
<td>21/02 05:35</td>
<td>21/02 05:35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26/02 15:35</td>
<td>27/02 14:25</td>
<td>27/02 14:25</td>
<td>27/02 14:25</td>
<td>27/02 14:25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20/02 10:37</td>
<td>21/02 05:35</td>
<td>21/02 05:35</td>
<td>21/02 05:35</td>
<td>21/02 05:35</td>
<td>21/02 05:35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26/02 15:35</td>
<td>27/02 14:25</td>
<td>27/02 14:25</td>
<td>27/02 14:25</td>
<td>27/02 14:25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
on Table 1, the communication frequency of system 3 is one communication every two minutes. So the “WaterCourseNodes” of the system 3 will only send their next measurements to the DSS at the time “22:01”. Unlike system 2, at time “22:01”, the “WaterCourseNodes” of system 3 will compute the slopes from time intervals [21:59; 22:00] and [22:00; 22:01] and send the maximum of this two values. As a consequence, the state change will be detected sooner. Thus, system 3 is more reactive than system 2. And, it is why the total exchanged packets number of System 3 is a little bigger than System 2 as showed in Table 3.

7. Related Work

No system of this type dedicated to flood monitoring has been proposed before. However, a context-aware system for water quality management exists. The InWaterSense project proposes a context-aware system to deduce the water quality of any water bodies (lake, river) [9, 24]. This system is totally built using Semantic Web technologies. SSN ontology is used as a nucleus in order to build the context model. They also use the Jess rule-based engine. Their rule format is based on the SQWRL language. It is able to build aggregation value using rules. Thus, the rules merge the characteristics of observable entities and those of the entity of interest. The rules infer the state of the water body without intermediate steps.
Compared to our approach, their rules are much more difficult to manage due to their complexity. Our formalization eases the reasoning process by splitting it into several steps: deduction of the states of observable entities; then, deduction of the state of the entity of interest.

The work of [25] proposes a WSN architecture called “Sepsen” in order to integrate several components in nodes: semantic annotator based on fragments of ontology, rule-based engine and a knowledge base that stores events. The goal is to decrease the number of event messages between sensors by classifying them as: share, forward or discard event. The share events are sent to other sensor nodes to update their knowledge base. The forward events are sent to the gateway. The discarded events are removed. However, the semantic annotation is done manually. The rule indicates that a sensed value should be above a threshold in order to become a share or forward event. Using the PowerTOSSIM environment, the “Sepsen” architecture is applied on a simulation scenario highlighting the energy saved.

None of these systems uses the same formalization based on observable entities, entities of interest and states. Thus, even if all these systems use a rule-based engine and ontologies, their rules are very complex and hard to maintain.

The Sensorsgrid4Env project [26] wants to help coastal flood planning managers to make decisions during coastal flooding events. This project proposes a mash-up application that integrates heterogeneous datasets: sensor data stream, historical database. The integration is made possible by a set of ontologies: SSN, SWEET, etc. In this project, the context is not modeled explicitly.

When dealing with complex phenomena like natural disasters, context-aware systems based on WSN become situation awareness system also based on WSN. In this type of system, the data management model is composed of different layers (sensor data, aggregation data, and situation representation knowledge) [27]. Our formalization can be integrated in the situation layer. In the work of [27], a situation is defined as the representation of a “structured part of the reality”. It contains all the description of entities involved in the situation. Context is a point of view of one entity about the situation.

8. Conclusions and Perspectives

In this paper, we have proposed a new formalization for the design and the implementation of context-aware systems. One of its advantages is that our approach can be used for multiple purposes. It can integrate both the monitoring of the studied phenomenon (feature of interest) and the management of the hardware and the software system used on WSN or IoT devices to observe it. More generally, it provides a unified way to deal with all the components/entities of an observation process. This formalization can be used in different application topics related to agriculture, environment, “smart care-smart home”, industry. To illustrate its use, we have provided an environmental use case application: the study of flood events in a watershed. In the Irstea institute, we have different data related to this topic and we will continue the implementation and the experiment of our approach in this application field. Simulation architecture is provided to evaluate the systems developed using our formalization. This architecture is based on the ontology concept with the use of the multi-agent system JADE and the rule-based engine Jess. Different scenarios for this environmental application will be proposed in our future work taking into account different states and extended wireless sensors reasoning capabilities. Our application will also be implemented with tools suitable for the limited resources of wireless sensors.

Acknowledgements

This research is partly funded by the grant of the French Auvergne-Rhône-Alpes region and, the grant of the European Regional Development Fund (ERDF). The authors would like to thank the team of the Orgeval observatory for their help.

References


[22]. Observatoire oracle (http://dx.doi.org/10.17180/ OBS. ORACLE).


