Wise Objects for IoT (WIoT): Software Framework and Experimentation

Ilham Alloui, Eric Benoit, Stephane Perrin, Flavien Vernier

Wise Objects for IoT (WIoT): Software Framework and Experimentation

Ilham Alloui[1], Eric Benoit[1], Stéphane Perrin[1], Flavien Vernier[1]
[0000−0001−7684−6502]

Université Savoie Mont Blanc - LISTIC, 5 chemin de bellevue, Annecy-le-Vieux, 74940 ANNECY, France
{ilham.alloui, eric.benoit, stephane.perrin, flavien.vernier}@univ-smb.fr

Abstract. Despite their expansion, Internet of Things (IoT) technologies remain young and require software technologies to ensure information management in order to deliver sophisticated services to their users. Users of IOT technologies particularly need systems that adapt to their use and not the reverse. To meet those requirements, we enriched our object oriented framework WOF (Wise Object Framework) with a communication structure to interconnect WOs (Wise Objects) and IoT. Things from IoT are then able to learn, monitor and analyze data in order to be able to adapt their behavior. In this paper, we recall the underlying concepts of our framework and then focus on the interconnection between WOs and IoT. This is enabled by a software bus-based architecture and IoT related communication protocols. We designed a dedicated communication protocol for IoT objects. We show how IoT objects can benefit from learning, monitoring and analysis mechanisms provided by WOF to identify usual behavior of a system and to detect unusual behavior. We illustrate our approach through two case studies in home automation. The first shows how a wise smart presence sensor learns on a classroom occupation. The second shows how a wise system helps us to see correlation among several WOs.

Keywords: Wise Object, IoT, Software Architecture, Communication, Knowledge Analysis.

1 INTRODUCTION

The Internet of Things (IoT) is known as the extension of current Internet to provide connection and communication between devices or physical objects referred to as "Things" [7]. Even growing substantially in number and use, the Internet of Things (IoT) technologies remain young and require software technologies to ensure data/information management among things in order to deliver sophisticated services to their users. Examples are home automation (HA) things which are getting more and more involved within our daily life: HA things are either within a ready-to-use systems (like boxes) or singles to be integrated to an existing system or platform. In both cases, when it is provided, support

for data monitoring and analysis is very limited [11].Users need to have a remote access to things, for instance to switch off lights they forgot or turn off the strove. Communication provided by existing IoT technologies should then involve basic data or information such as current state of things. Moreover, users of a HA need the technology adapts to their use and not the reverse: in our previous example, the system (instead of users) would for instance detect that unusually the lights are switched on at midnight. Then it would either adapt to this change (i.e. register this new behavior as usual) or take initiative to raise an alert or to switch the lights off depending on the knowledge it has (i.e. if no presence is detected).

This implies that the system is able to: (a) identify usual behavior; (b) detect the changes in the way it is being used and (c) either react by taking initiative or change its behavior to comply with those new usages. Our proposal is that intelligent software systems could enhance IoT with useful capabilities such as learning, monitoring and adaptation to meet users' requirements.

Starting from works on IoT and on intelligent software systems [12] [6] we aim to add value to IoT through WOF (Wise Object Framework) [3], a software object framework that provides things (be them physical or software), built-in mechanisms for learning, monitoring, analyzing and managing data/information (see Figure 1). Those software mechanisms

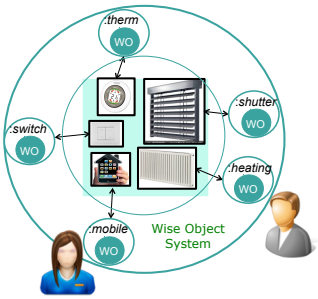


Fig. 1. Example of Home Automation Wise Object System [2]

allow IoT-based systems like in HA to: (a) identify common usage (i.e. usual behavior of their users); (b) detect changes in usage (unusual behavior); (c) adapt to the new usage (system in automatic mode) or simply give information to the users (manual mode).

Identifying a system common usage by software is not an easy work. At the best of our knowledge, common usage is usually studied from a psychological point of view for the human [1] or from the signal processing point of view with change detection methods [5] for data, but never from the software point of view. This research issue raises many questions such as: what is considered as common usage? Is common usage necessarily related to time? Is there an interval of acceptance of unusual behavior?

Which one? What methods/techniques better identify common usage? In which context? etc.
As users' behavior identification and system adaptation rely on data collected from connected things that may be distributed as is the case in IoT, we realized a software bridge linking IoT objects to our WOF software objects. In this paper we focus mainly on this link between software "wise" objects (WOs) and IoT through the WOF. WOs can be seen as software avatars related to things. This paper extends [2] and introduces a new experiment which for the first time highlights the behavior of several WOs in the same environment and shows that correlated unusual behavior among several WOs may result from a same cause. The first case study presented in [2] illustrates the behavior of a single wise thing (a smart presence device) and mainly shows how such thing is able to manage and use presence events to result knowledge on a classroom occupation during a year. The new case study in this paper is based on a system of three wise things. Our aim is twofold: (a) show that our approach works also with several wise things and (b) highlight its ability to show correlated unusual behavior among several things. Such knowledge indicates that behavior change may result from a same triggering event which is very useful for diagnosing or explaining unusual behaviour of a system. In Section 2 we recall the concept of WO and WOF, the behavior of a WO, its interaction with other WOs as well as our first representation of common usage. Section 3 introduces the connection between a WO and an IoT, from the software interaction point of view in Section 3.1 and from the communication medium and protocol point of view in Section 3.2. In section 4, we present two cases studies in home automation domain with the results we obtained using the framework. The first case study focus is on the behavior of a single wise thing while the second one focus is on a system of several wise things. Finally, we discuss our approach and conclude with ongoing work and some perspectives.

2 WO AND WOF

2.1 WOF

WOF is founded on the concept of WO. Our design decisions behind the WOF are guided by the following requirements: software support should be the less intrusive possible, reusable and generic enough to be maintainable and used in different application domains with different strategies. Developers should be able to use the framework with the minimum of constraints and intrusion in the source code of the application. We consequently separated, in the WOF, the "wisdom" and intelligence logic (we name abilities) of the objects from application services (we name capabilities) they are intended to render. As shown by Figure 2, we designed the WOF according to a layered architecture:

- the core layer, i.e. the framework building blocks, consists of a set of interrelated packages and classes that embed basic mechanisms for introspection, monitoring, analysis and communication among WO instances. WO is the main class from which a system developer may

specialize application-level classes such as the Switch and Shutter classes within the home automation system in the example;
- the software system layer: contains the package and classes related to software systems developed for end-users. The home automation cited so far is a representative of such systems. Classes representing things can inherit the structure and behavior of the WO class in the Framework layer;
- the instantiated software system: gathers the instantiated application software systems from the previous layer. Instances of application-related classes are avatars for physical or logical objects (things).

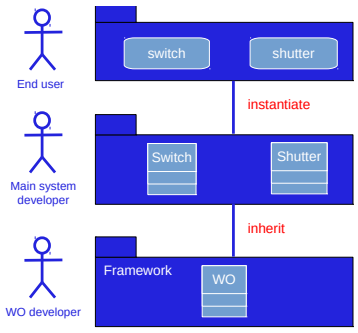


Fig. 2. WOF concrete architecture [2]

To build a WO system, the WOF provides a communication bus (Gava) for the interaction between WOs. Interactions are managed through a manager object that establishes the configured pairing between events and actions according to a publish-subscribe pattern. Figure 3 illustrates this interaction.
When a method is invoked on a WO instance: (a) the wise part of the instance raises an event at the end of the invocation; (b) the manager catches the event and sends orders to all WO instances interested in the initial event (paired WOs); (c) the paired WO instances execute the corresponding method; (d) the manager checks that the order has been correctly executed. The communication and pairing system are detailed in [4] and were initially limited to communication and pairing between WO instances.

2.2 Concept of WO

We define a Wise Object (WO) as a software object able to learn by itself on itself and on its environment (external knowledge), to deliver expected services according to the current state and using its own experience. Wisdom refers to the experience such object acquires by its own during its life. We intentionally use terms dedicated to humans as a metaphor. A Wise Object is intended to "connect" to either a physical

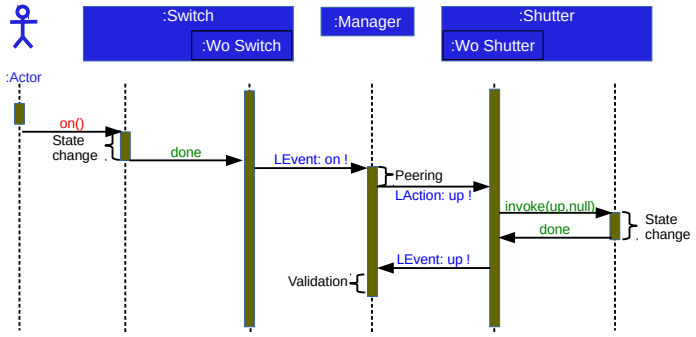


Fig. 3. UML Sequence diagram of the interaction between a WO switch and a WO shutter [2]

entity/device (e.g. a vacuum cleaner) or a logical entity (e.g. a software component). As wise object could be a cleaner able to learn on how to clean a room depending on its shape and dimensions. In the course of time, the cleaner could in addition improve its performance (less time, less energy consumption, etc.). A WO is then characterized by:

- its autonomy: it is able to behave with no human intervention;
- its intelligence: it observes itself and its environment, analyzes them and uses its knowledge to decide how to behave (introspection, monitoring, analysis, planning);
- its adaptivity: it changes its behavior when its environment changes;
- its ability to communicate: with its environment that includes other WOs and end-users in different locations.

A WO built-in behavior involves two states: The dream state and the awake state, see Figure 4.

The dream state is dedicated to acquiring knowledge about its own capabilities and to analyzing usage-related knowledge. The awake state is the state where the WO executes its methods invoked by other objects (external service requests) or by itself (internal requests), and, monitors such execution while recording usage-related knowledge.

A WO's capability-related knowledge is itself stored as a state diagram. The WO executes the methods of its sub-class (i.e. an application class like Switch) to know the effect on the attributes of this sub-class instances. Each set of attribute values produces a state in the diagram and method invocation produces a transition (see Figure 5). The main constraint in this step is that the method invocation must have no effect on the application when the WO is dreaming. This is solved thanks to a bus-based system architecture described in [3] with disconnection/reconnection mechanisms.

Regarding knowledge on an application object usage, two kinds of situations are studied: emotions and adaptation of behavior. We define an emotion of WO as a distance between its current usage and its common

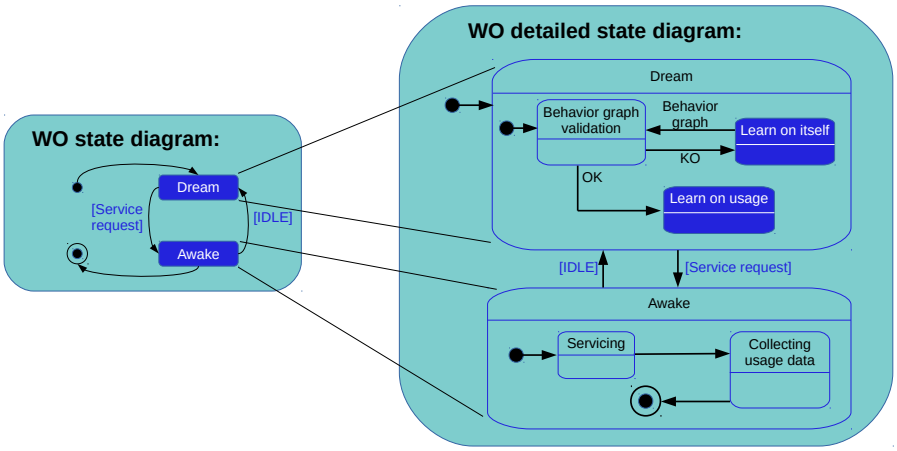


Fig. 4. UML state diagram of WO built-in behavior [4]

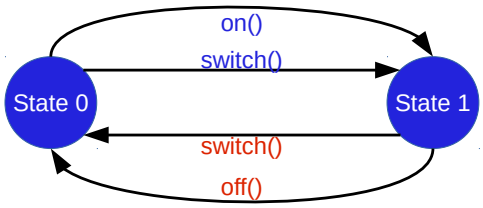


Fig. 5. UML state diagram of a switch built by its WO

usage (i.e. unusual usage). WO can be stressed if one of its methods (services) is more frequently used or conversely, a WO can be bored. WO can be surprised if one of its method is used and this was never happened before. Emotions of a WO are a projection of its current behavior with regard to its usual behavior. In Subsection 2.3, we present a Data Analyzer based on a statistical method we implemented in WOF to identify usual/unusual behavior. When a WO expresses an emotion, this information is caught by the WO system that may consequently lead to behavior adaptation. At the object level, two instances of the same class that are used differently – different frequencies, different methods... – may have different emotions, thus, different behavior and interaction in the WO system.
A WO uses its capability-related knowledge to compute a path from a current state to a known state [8]. According to the frequency of the paths used, a WO can adapt its behavior. For instance, if a path is often used between non-adjacent states, the WO can build a shortcut transition between the initial and destination states and then build the corresponding method within its subclass instance (application object). This consequently modifies the capability-related graph of this instance.

2.3 WOF and Data Analyzers

The WOF provides a connector to an evolving set of analyzers whose role is to identify a WO common behavior (usage) and to detect emotions when they occur. Each analyzer connected to a WO is waked-up during the WO's dream state to analyze the last events and to update its knowledge.
Let us recall our preliminary model of common usage introduced in [4]. It is based on a statistic approach and defines the common usage as weaker forms of stationarity (WSS) from the statistic point of view.
Let $x(i)$ be a continuous and stationary time random process. A process is a WSS process if and only if:

$$\begin{aligned} &E\left[x(i)\right] = \mu \quad \forall i, \\ &Var\left[x(i)\right] = \sigma^2 \neq \infty \quad \forall i, \\ &Cov\left[x(i), x(i-k)\right] = f(k) = \rho_k \quad \forall i \forall k. \end{aligned}$$

As the common usage can change along the time, we compute the stationarity – the common usage – on a sliding window of size w:

$$\begin{aligned} &E\left[x(i)\right] = \mu(t) \quad \forall i \in [t-w, t], \\ &Var\left[x(i)\right] = \sigma^2(t) \neq \infty \quad \forall i \in [t-w, t], \\ &Cov\left[x(i), x(i-k)\right] = f(k,t) = \rho_k(t) \forall i \in [t-w, t] \forall k, \end{aligned}$$

where the time series $x(i)$ are the occurrences $\left[e_\tau^{t-w} \dots e_\tau^i \dots e_\tau^t\right]$ of a given event – i.e. transition – τ between t-w and t.
As our system cannot be perfectly stationary, we relax the definition of WSS and consider that the system is in common use if and only if:

$$\begin{aligned} &\mu(t+1) \in [\mu(t-w), \mu(t)] \\ &\sigma^2(t+1) \in [\sigma^2(t-w), \sigma^2(t)] \\ &\rho_k(t+1) \in [\rho_k(t-w), \rho_k(t)]. \end{aligned}$$

In other words, if the new mean, variance or autocovariance at time $t+1$ are in their corresponding ranges, the new event occurrence at time $t+1$ is considered as a common usage, otherwise it is unusual.
According to this definition, we define an emotion as the distance between the current usage at $t+1$ and the common usage between $t-w$ and t.This distance $d(x(i))$ is defined by the following centered normalized scale between -1 and 1, where:

$$d(x(i)) = \begin{cases} d(E\left[x(i)\right]), \\ d(Var\left[x(i)\right]), \\ d(Cov\left[x(i), x(i-k)\right]), \end{cases}$$

where

$$d(E\left[x(i)\right]) = \frac{E[x(i)] - \overline{E[x(j)]}}{(\max(E[x(j)]) - \min(E[x(j)]))/2},$$

$$d(Var\left[x(i)\right]) = \frac{Var[x(i)] - \overline{Var[x(j)]}}{(\max(Var[x(j)]) - \min(Var[x(j)]))/2},$$

$$d(Cov\left[x(i), x(i-k)\right]) = \frac{Cov[x(i), x(i-k)] - \overline{Cov[x(j), x(j-k)]}}{(\max(Cov[x(j), x(j-k)]) - \min(Cov[x(j), x(j-k)]))/2},$$

$j \in [t-w, t]$ and $\overline{E\left[x(j)\right]}$, $\overline{Var\left[x(j)\right]}$ and $\overline{Cov\left[x(j), x(j-k)\right]}$ are respectively the means of means, variances and autocovariances on the range $[t-w, t]$.
Thus, when a new event occurs at $t+1$, we compute the distance $d(x(i))$ with the common usage between $t-w$ and t. If all values of the distance – $d(E\left[x(i)\right])$, $d(Var\left[x(i)\right])$ and $d(Cov\left[x(i), x(i-k)\right])$ – are between -1 and 1, the behavior is considered as common, otherwise it is identified as unusual relatively to the knowledge on the common usage.

3 FROM WOF TO IOT

To meet IoT related requirements cited in Section 1, we extended our framework WOF [4] with mechanisms to relate "things" to WOs. We thus define an object in WIoT as a peer composed of a physical object (thing) and a logical (software) object (WO). A WO can be viewed as an avatar of a thing. From now on, the term object will be used to refer to the thing-avatar peer.

3.1 WO model for IoT

When a thing (e.g. a physical switch) joins the application system (e.g. HA system), its corresponding avatar (a Switch class instance) is automatically instantiated and this pair forms then a new object. This means that the avatar's class of the thing exists. As it is not desirable and even not relevant to provide everything in the system with the ability of learning and analysis, we introduced a class named Generic WO without the introspection ability.
Like WO class instances, instances of Generic WO are able to construct their capability-related graph, but they cannot use introspection to analyze their behavior. A Generic WO instance learns its behavior from state

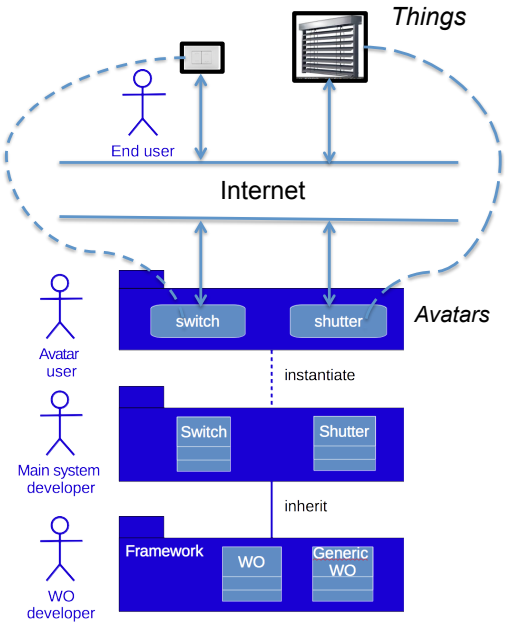


Fig. 6. WIoT architecture [2]

change messages it receives from the thing it is related to. This way, a generic WO can be related to any "thing" able to communicate its state and state changes. This is not a strong constraint as recent physical connected objects are generally able to communicate changes in their state. In the case of home automation, devices using ZigBee [15], Z-Wave [14] or other modern systems, communicate their capabilities through profiles or other kinds of descriptions. Figure 7 presents the UML Class diagram of WOs including the Generic WOs. As shown in the figure, a generic WO is a WO where the "invoke" method is redefined. While through the "invoke" method, a WO can invoke methods of its sub-classes (i.e. application classes whose instances are avatars for things), the class Generic WO has no subclass. Then when the "invoke" method is called, it just updates its usage-related diagram (knowledge on the way the thing is being used).

Figure 8 illustrates the communication flow between a physical switch and its associated physical shutter. The "PEvent/PAction" and "LEvent/LAction" are respectively sent through the physical (P) and logical (L) communication media.

When the switch is activated, it sends the message "PEvent:on!" to its avatar. When receiving this message, the wise part of the avatar learns that the state of its associated object has changed, thus it executes the method on itself, "on()" in the example, to be in a consistent state with its thing. When this is done, the switch object sends "LEvent:on!" message to inform the system that its state has changed.

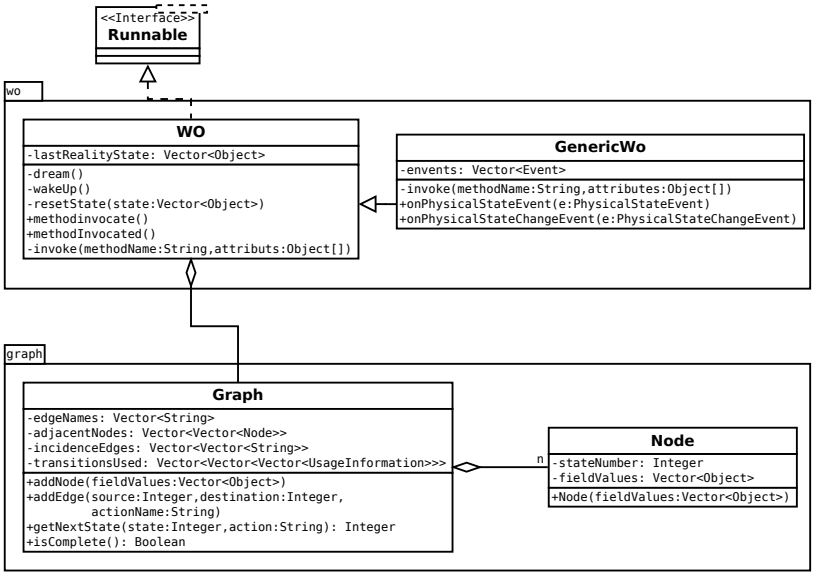


Fig. 7. UML Class diagram of generic WO [2]

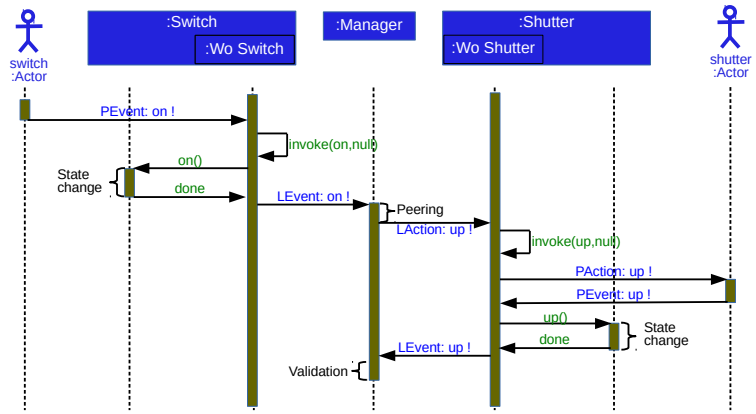


Fig. 8. UML Sequence diagram of the interaction between a physical switch and a physical shutter [2]

Let us note the system can manage pure logical objects namely objects that are not linked to physical objects. Figures 9 and 10 illustrate 2 cases. The former, Figure 9, presents the sequence diagram of a logical switch activated respectively through software and through a physical shutter. Physical devices and end-users are represented as external actors (fellow symbol) to a WOS whereas logical things (software) are represented as internal actors (blue boxes).

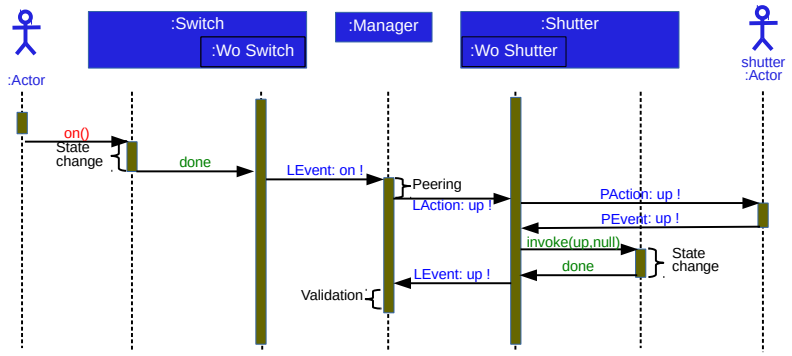


Fig. 9. UML Sequence diagram of the interaction between a logical switch and a physical shutter [2]

Figure 10, presents the sequence diagram of a physical bell push that launches on the system a video application to check who is ringing. In this case, the video application is considered as part of the WOS.
In the cases where a thing has no avatar in the system, it is associated with a generic WO. Figure 11 illustrates this configuration where a physical switch has an action on an object that is not explicitly defined in the system. Although it is named "unknown:Actor", it must respect the communication protocol defined in Section 3.2. Let us notice that there is no constraint about the fact that the "unknown:Actor" must be a logical or a physical object, it can be of both kinds.
As shown in the different sequence diagrams, the WOF offers the required support for all combinations between two objects, be them physical (e.g. devices) or logical (i.e. software). Let us however note that, if a physical object is used – a thing – a logical object – its avatar – is necessarily associated with it. Moreover, a physical object does not necessarily have a known avatar in the system. In this case, it must respect communication constraints detailed in the next section.

3.2 Communication Protocol

The WOF provides a communication system for WOs to interact and exchange information. It corresponds to communications between ob-

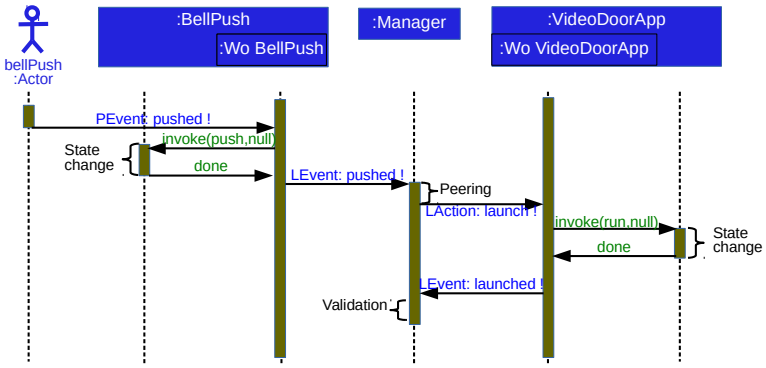


Fig. 10. UML Sequence diagram of the interaction between a physical bell push and a logical video application [2]

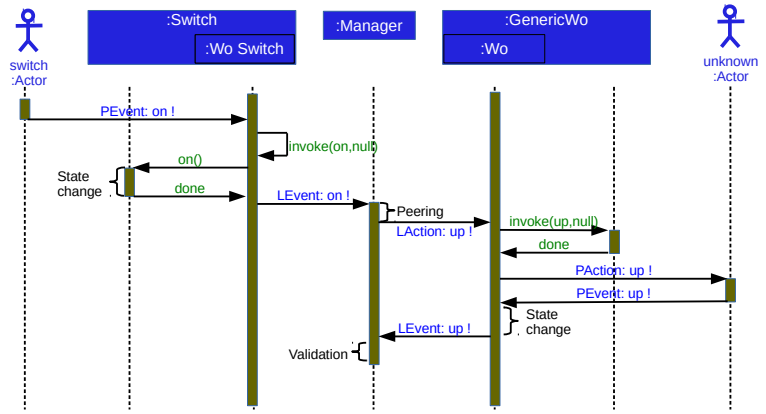


Fig. 11. UML Sequence diagram of the interaction between a physical switch and a physical object not implemented as WO (no avatar) and managed as a generic WO [2]

jects in the logical world (the software application that manages those objects). The physical objects/things are from IoT and communicate in our case through an MQTT communication system [13]. Thus we implemented a bridge between both those systems in WOF to enable communication between WOs and things.
As the communications between WOs and between a WO and its associated physical object are not of the same nature, we defined two kinds of communications we named respectively "logical" (WO-WO) and "physical" communications (WO-thing). From the conceptual point of view, this approach can be considered as a dedicated communication medium. Figure 12 shows this communication flow among physical things (button and light), their logical avatars (software WOs) and the manager software object.

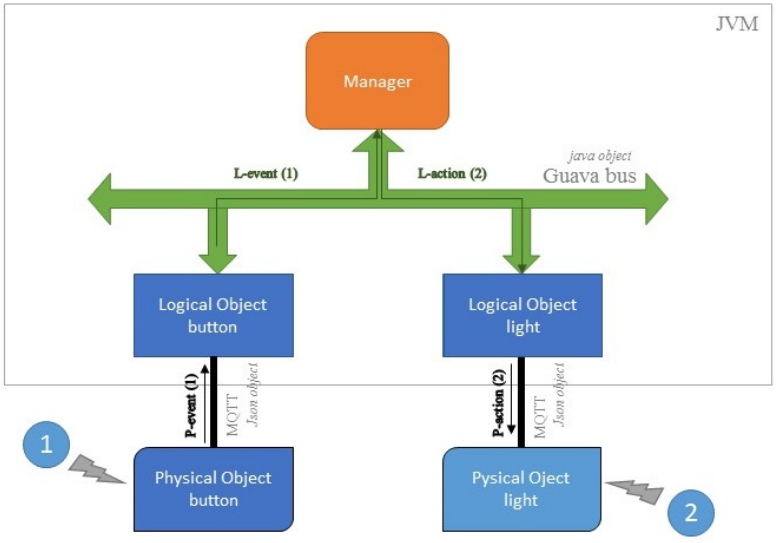


Fig. 12. Communication flow between manager, logical and physical objects [2]

From the implementation point of view, WOF uses the publish/subscribe-based Guava bus and IoT communication is based on MQTT with JSON format for messages. As both are publish/subscribe-based systems, a simple bridge is used to exchange messages from one to the other. To separate "logical" and "physical" communications, we use different types of messages that we defined as follows:

- Physical messages:
  - "PhysicNewDevice": message sent by a physical object when it connects to MQTT server.
  - "PhysicStateChange": message sent by a physical object when its state changes; it contains the event that generates the state change.

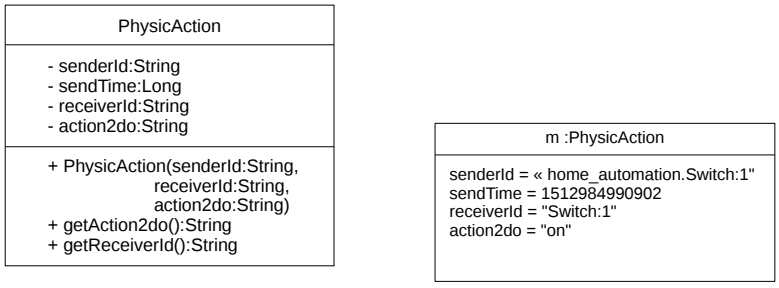
- "PhysicAction": message sent to a physical object so that it performs an action.
- "PhysicGetState": message sent to a physical object so that it sends its state; this message type is mainly dedicated to generic WOs so that they ask the things their state.
- "PhysicState": message sent by a physical object to indicate its state; this message is the answer to "PhysicGetState" message.

– Logical messages:
- "LogicNewDevice" message sent by a WO when it is created in the WOF.
- "LogicalStateChange" message sent by a WO when its state changes.
- "LogicalAction" message sent to a WO so that it performs an action.

The bridge only translates Java object messages to JSON objects and vice-versa according to the following rules:
– the MQTT topic is defined by [basetopic]/[Class] where:
- basetopic is free, "Wo" in our example,
- Class is the name of class message including the package name, for example a "PhysicAction" message of package "bus" is sent on topic "WO/bus.PhysicAction",

– any attribute of the Java object is an attribute of JSON.

Figures 13(a) and 13(b) illustrate respectively the "PhysicAction" class and an object that is translated as the following JSON message:



(a) PhysicAction class used to send "PhysicAction" message from a WO to its thing

(b) Example of PhysicAction object

Fig. 13. PhysicAction class used to receive "PhysicAction" message by a WO [2]

```
MqttConnector from MQTT: Wo/bus.PhysicAction->
{"senderId":"home_automation.Switch:1",
 "sendTime":1512984990902,
 "receiverId":"Switch:1",
 "action":"on"}
```

The WO identified by id "home_automation.Switch:1" sends, at time 1512984990902, the order "on" to its thing identified by "Switch:1".
Finally, the communication system we built between WOF and IoT allows us to connect:

- A thing defined in the WOF.
- A thing not defined in the WOF, but that can communicate using our protocol.
- A thing not defined in the WOF, that communicates with another medium (ZigBee, ZWave, WiFi...)

The constraint is that the thing must be able to give information on its state change. Figure 14 illustrates on a switch example the three communication cases.

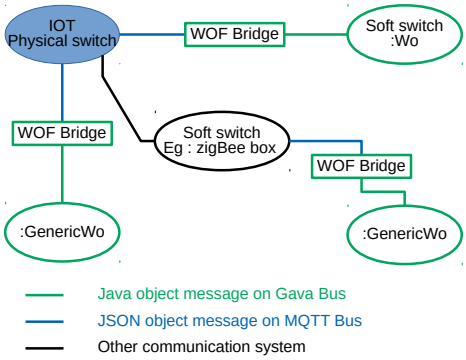


Fig. 14. An IoT object, like a switch can be connected to WOF according to 3 ways: a) the thing can communicate using our MQTT protocol and its avatar exists in the WOF; b) the thing can communicate using our MQTT protocol but its avatar does not exist in the WOF; c) the thing cannot communicate using our MQTT protocol [2]

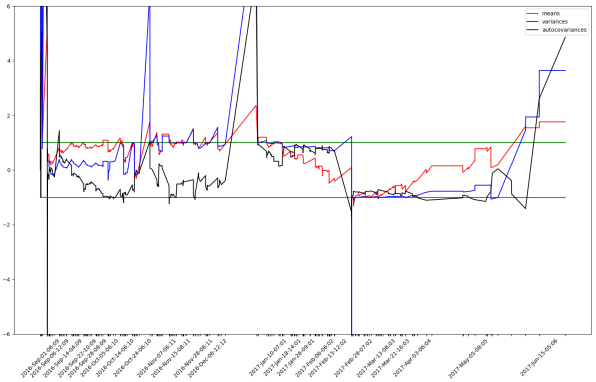
4 EXPERIMENTAL IMPLEMENTATION

We experimented our framework and its underlying approach on several examples. We present in this section two of them: the first one consists of a wise smart presence sensor in a real situation while the second one consists of a set of home automation related objects, namely a light, a shutter, a switch and a heating device. For each case we give a description of the objects, the goal of the experiment, data used/generated as well as the results and observations we did.

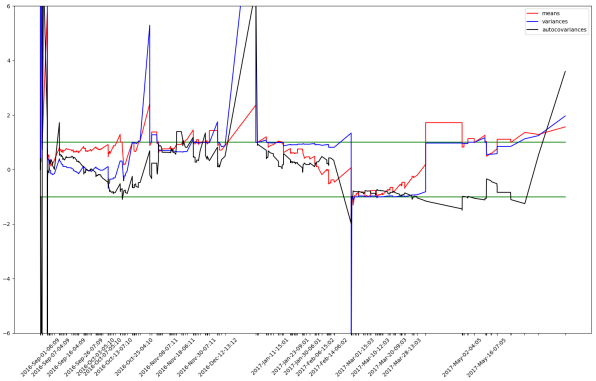
4.1 Use case 1 description and experimental results

To illustrate the use of WOF including the interconnection of WOs and IoT, we took the case of a presence smart sensor within a classroom with

the objective to identify the usual usage of the room and detect habit change (unusual behavior). This allows us to experimentally validate our approach of habit change measurement.
One objective of the case study is to know if our system is able to detect habit change in relation to a common usage, especially regarding student vacation periods. The smart presence sensor provides the "presence" state when persons are in a room and the "no presence" state when not. It is worth noting that the smart capacity of the sensor offers the possibility to filter the output state: "no presence" state is delivered if no detection occurs for one minute.
Attempting to identify a common usage (habit) requires a significant volume of data that depends on the temporal observing window or the number of observations taken into account. To cover different volumes of data, it is obviously relevant to consider a long duration of observation. However to avoid a long experiment, one year in our case, we simulate the smart presence sensor outputs by using real data coming from the real-time scheduling system of our university. Thus, real data injected in the system corresponds to the outputs of the smart presence sensor placed into a classroom. At each "state change" event from the sensor, a physical timestamped message, including the sensor id, is sent using MQTT protocol. The next section presents some results of our experimentation.
Figures 15 and 16 illustrate the experiment results according to the definition of common usage given in Section 2.3, with only one k value for covariance. Our purpose is to highlight the strengths and weaknesses of our first modeling of common usage. As the focus of this paper is on interconnection between wise objects and IOT, we do not provide an in-depth analysis of common usage modeling. This issue will be studied in the future.
In this experiment, we observe for the sensor, the delay between events as well as the time spent by it in different states. The events are the detection of "new presence", when the sensor switches from "no presence state to "presence" state and conversely, the detection of "no more presence", when the sensor switches from "presence state to "no presence" state. Figures 15(a) and 15(b) give the common usage respectively computed from the "new presence" events and from the "no more presence" events. In other word, Figure 15(a), gives the common variation of delay between two successive "new presence" events. Figures 16(a) and 16(b) give the common usage respectively computed from the duration of presence and the duration of no presence in the room. The results are computed with 15 days as window size w, any data older than 15 days are forgotten. Thus, in the range $[-1, 1]$, between green lines in the figures, the behavior is considered as common usage regarding the last 15 days. Outside the range $[-1, 1]$, behavior is considered as unusual, we qualify it as "emotion". The emotional force is represented by the distance of the behavior to the common usage.
These preliminary results are encouraging. They highlight, from different points of view – state changes and time spent in a state – the change in the classroom usage. Each part with an important distance from the common usage corresponds to holidays (in France):

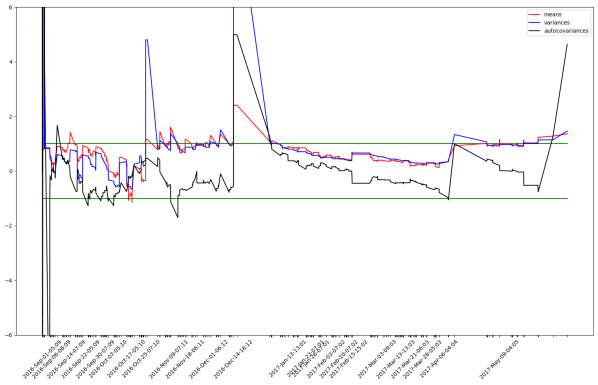


(a) Classroom usage representation computed from "new presence" events

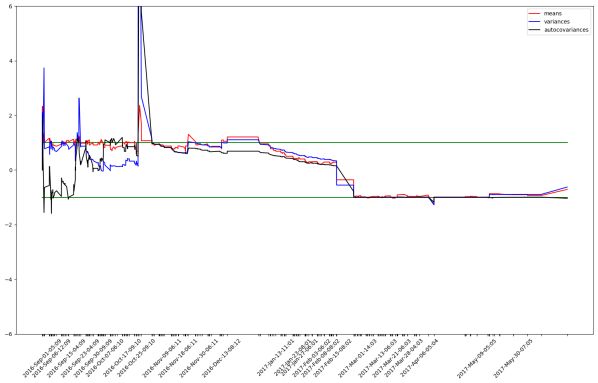


(b) Classroom usage representation computed from "no more presence" events

Fig. 15. Common usage and Emotion representation based on events [2]



(a) Classroom usage representation computed from "presence" duration



(b) Classroom usage representation computed from "no presence" duration

Fig. 16. Common usage and Emotion representation based on time spent in state [2]

- 1 week for the Halloween holidays in October,
- 2 weeks for the Christmas holidays in December,
- 1 week for the winter holidays in February,
- 1 week for the Easter holidays in April and
- the end of the school year in June.

Each part with a small distance from the common usage corresponds to weekends. Let us note that each part detected as unusual depends on the usage done during the 15 days before. Thus weekends are strongly detected when the room is frequently used in the week for example between September and December. The holidays are strongly detected before January but, weakly detected after December. As the observed room is an amphitheater, it is more used at the beginning of the school year than at the end.

We consider those results as preliminary because there is a combinatorial problem in using the underlying analysis method. Sensor modeling with 2 states and 2 transitions leads to 12 graphics, with only one k value for the covariance, to identify common usage and emotions. For a given object, the maximum number of "common usage" related graphics is $n*a*(2+n_k)$, where n is the number of states, a is the number of methods and n_k is the possible number of values of k. Thus, an information fusion step is required to reduce the combinatorial problem. Another point is that our system does not react if nothing happens during an unusual period; It detects changes only when an event occurs. The management of "no event" must also be performed by the system. Both those points will be addressed in future work.

4.2 Use case 2 description and experimental results

The second case study focus is on considering a set of objects instead of a single one as in the first case. The objective behind the experimentation is to be as close as possible to a real situation. Thus we used the WOF simulator to simulate a home automation system, within a classroom, composed of four physical objects: a switch, a light, a rolling shutter and a heating device. As in the previous case, we would like to know if our system is able to detect habit change in relation to a usual usage. As several objects are involved in the system, we also would like to study the behavior correlation among those objects. The usual behavior is the following:

- each day the heating system is automatically started at 6:00 am and it functions until 6:00 pm;
- each day at 8 am (+-5mn), the shutter is manually opened by a teacher or a student of the first course. Eleven hours later (+-30mn), the shutter is closed by the caretaker;
- the light is manually switched on at 8 am (+-5mn), at the arrival of the teacher and students to the course. It is manually switched off at the end of the first course, 1 hour later (+-5mn) and remains off until the following day.

We observe for each object, the delay between events. The events are:

- for the light: the detection of "light on" events, when the sensor switches from "no light" state to "lighting" state; the reverse holds for the detection of "light off" events;

- for the shutter: the detection of "shutter open" events, when the actuator switches from "shutter closed" state to "shutter opened" state; the reverse holds for the detection of "shutter close" events;
- for the heating: the detection of "heating on" events, when the actuator switches from "no heating" state to "heating" state; the reverse holds for the detection of "heating off" events;

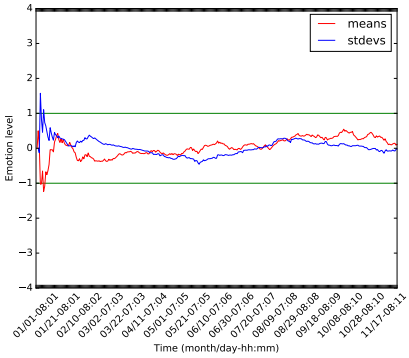
Figures 17 depicts the common usage computed respectively for the light, shutter and heating. Each sub-figure gives the common variation of delay between two successive events on an object (red) as well as the standard deviation (blue).
The results are computed within a memory window size w of 100 events corresponding to a duration of 3 months in our simulator. Data older than 3 months are forgotten (no longer considered in the computation). Thus, in the range $[-1, 1]$, between green lines in the figures, the behavior is considered as usual (common usage) during the last 3 months. Outside the range $[-1, 1]$, behavior is considered as unusual and the emotional force is represented by the distance of the behavior to the common usage. As shown by figures 17:

- the heating system behaves as usual: no emotion detected; this is consistent with the fact that the heating is automatically started and stopped everyday exactly at the same instants (no random effect);
- we can notice that for both "light on" and "light off" events, at the beginning of the time window, there is a chaotic variation resulting an emotion (a surprise): as the events outside the time window are forgotten, events occurring in the new window are considered unusual by the object that progressively integrate them as a usual behavior ;
- generally there is no significant change in the behavior of the shutter. We can however note sometimes a slight chaotic perturbation as in the sub-figure of "light off" events: this is due to the cumulative random effect.

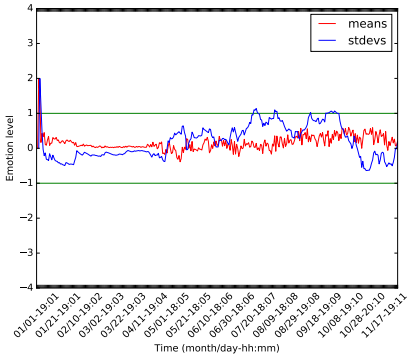
While in the previous situation nothing special happened relatively to the common behavior, in Figures 18, we can clearly see different shapes for the behavior of the light and the heating device. Something happened that disturbed the system without affecting the shutter: a power cut at 10 AM the 150th day of the experiment (May 21st). In this case we clearly see a variation value outside the normality boundaries. This translates an emotion that has an impact on the means computation. As the power cut lasted 2 hours, the delay between successive events have been disturbed (around 2 hours instead of 24). This highlights the ability of a wise system to show correlated unusual behavior among several things. Its knowledge monitoring and analysis capabilities are therefore useful for diagnosing or explaining unusual behaviour of IoT-based systems.

5 CONCLUDING REMARKS AND FUTURE WORK

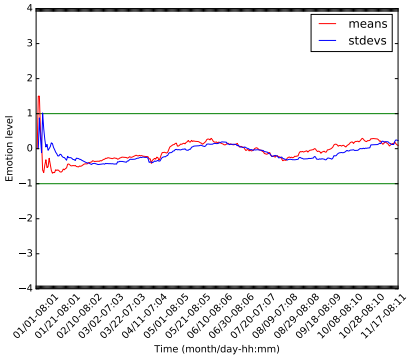
To meet the growing user requirements for IoT technology-based systems that adapt to their needs, we propose WOF (Wise Object Frame-



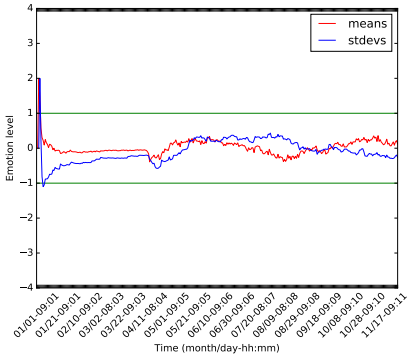
(a) Classroom usage representation computed from "light on" events



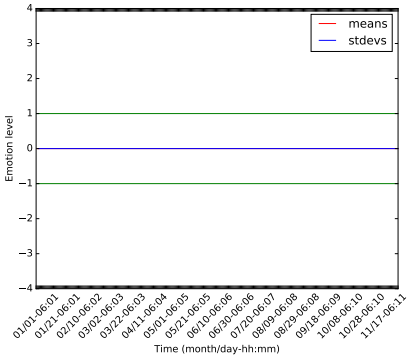
(b) Classroom usage representation computed from "light off" events



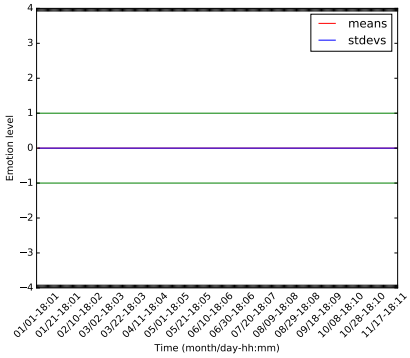
(c) Classroom usage representation computed from "shutter open" events



(d) Classroom usage representation computed from "shutter close" events

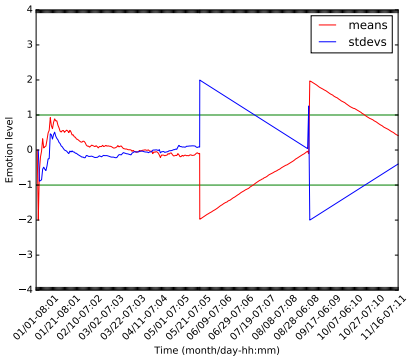


(e) Classroom usage representation computed from "heating on" events

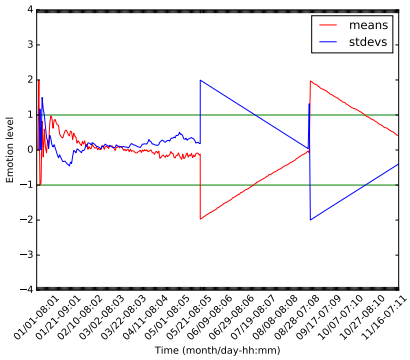


(f) Classroom usage representation computed from "heating off" events

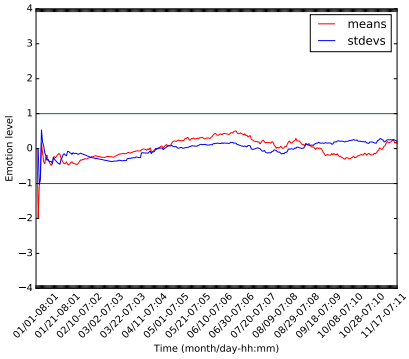
Fig. 17. Common usage and Emotion representation based on events



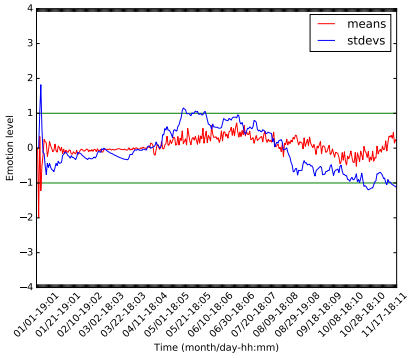
(a) Classroom usage representation computed from "light on" events



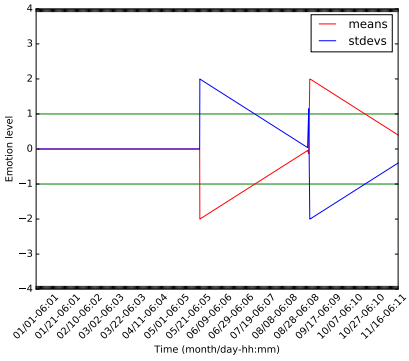
(b) Classroom usage representation computed from "light off" events



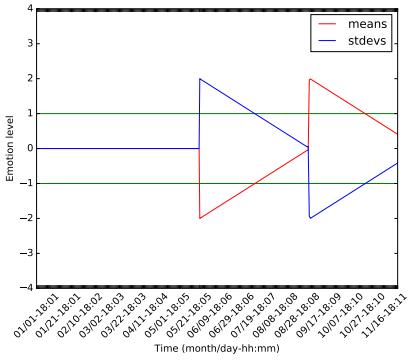
(c) Classroom usage representation computed from "shutter open" events



(d) Classroom usage representation computed from "shutter close" events



(e) Classroom usage representation computed from "heating on" events



(f) Classroom usage representation computed from "heating off" events

Fig. 18. Usage and Emotion representation based on events, integrating an unusual event: a power cut at 10am the 150th day of the experiment.

work) [4], an object oriented framework to develop software-intensive systems ("wise systems") able to learn, monitor and analyze data/information among distributed things. We recalled in this paper the underlying concepts of WO (Wise Object) and WOF and focused in particular on how to interconnect IoT to WOF to benefit from its useful built-in mechanisms (namely learning, monitoring, adaptation) and meet users' requirements. The communication protocol we propose allows things to communicate themselves. It is designed as part of WOF. For each thing, we propose to use a software avatar, a WO in our case so that it become possible to manipulate and manage this representation of the thing. In this paper, we focus on the description of the communication structure and especially the corresponding software as well as the proposed protocol which makes possible interaction between a wise object and a physical thing.

We show that using the proposed structure, any thing from IoT is manageable. The only constraint is that the considered thing uses the proposed communication protocol. Thus, the system is able to communicate with any thing, whether known or unknown. If a thing is unknown – there is no WO implementation dedicated to this thing – a generic WO implementation can be used as an avatar for this thing.

We illustrated our approach on two case studies within home automation domain and showed how wise things (smart presence sensor, light, etc.) are able to identify common usage and unusual behavior. This is enabled by analyzers connectable to WOF: in this paper we presented one of them we built using a statistic method based on "stationarity" theory.

We show the interest for the system by performing a first experiment based on real data with a single thing then a second experiment based on simulated behavior of a classroom using home automation things (a smart presence sensor, shutter, etc.). One interesting finding comes from the fact that the results show the changes due to the usage context: vacation or weekends in the first case study or a power cut in the second case. In the broader context of home automation, we are convinced that our approach can be useful, for instance to assist old people in their home (individual or nursing). Authors in [9] and [10], adopt a user driven approach and present an interesting study on nursing home users' expectations from AAL (Ambient Assistant Living) technologies. One important outcome is that there is a need for systems able to detect users' activity level and to notify the care staff and/or family members about unusual behavior.

In future work, we plan to focus our research mainly on the modeling and the management of common usage and emotions. As highlighted in the experimental results, issues of information fusion and of management of situations like "nothing happens during an unusual time" must be addressed to obtain results that are more accurate, usable and up-to-date upon request. Another important issue is studying the correlation of behavior among a set of WOs composing a system, in particular measuring the impact of a WO emotion on the other WOs. The next step for us is to be able to express emotions with a higher semantic level than the present one (i.e. the statistical method) in order to communicate lighter amounts

of information to the system. The system can then react according to an aggregated information rather than multiple pieces of information.

References

1. Aarts, H., Verplanken, B., Knippenberg, A.: Predicting behavior from actions in the past: Repeated decision making or a matter of habit? Journal of Applied Social Psychology **28**(15), 1355–1374 (2006). https://doi.org/10.1111/j.1559-1816.1998.tb01681.x, https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1559-1816.1998.tb01681.x
2. Alloui, I., Benoit, E., Perrin, S., Vernier, F.: Wiot: Interconnection between wise objects and iot. In: Proceedings of the 13th International Conference on Software Technologies - Volume 1: ICSOFT,. pp. 494–505. INSTICC, SciTePress (2018). https://doi.org/10.5220/0006870205280539
3. Alloui, I., Esale, D., Vernier, F.: Wise Objects for Calm Technology. In: 10th International Conference on Software Engineering and Applications (ICSOFT-EA 2015). pp. 468–471. ICSOFT-EA 2015, SciTePress 2015, Colmar, France (July 2015). https://doi.org/10.5220/0005560104680471, https://hal.archives-ouvertes.fr/hal-01226219
4. Alloui, I., Vernier, F.: Wof: Towards behavior analysis and representation of emotions in adaptive systems. Software Technologies, 12th International Joint Conference, ICSOFT 2017, Madrid, Spain, July 24-26, 2017, Revised Selected Papers (CCIS) **868**, 244–267 (2017)
5. Aminikhanghahi, S., Cook, D.J.: A survey of methods for time series change point detection. Knowl. Inf. Syst. **51**(2), 339–367 (May 2017). https://doi.org/10.1007/s10115-016-0987-z, https://doi.org/10.1007/s10115-016-0987-z
6. Brun, Y., Desmarais, R., Geihs, K., Litoiu, M., Lopes, A., Shaw, M., Smit, M.: A design space for self-adaptive systems. In: de Lemos, R., Giese, H., Müller, H.A., Shaw, M. (eds.) Software Engineering for Self-Adaptive Systems II. Lecture Notes in Computer Science, vol. 7475, pp. 33–50. Springer, Dagstuhl Castle, Germany (2013), http://literature.vs.eecs.uni-kassel.de/publications/2013/BDGLLSS13
7. IEC: IoT 2020: Smart and Secure IoT Platform : White Paper. International Electrotechnical Commission (2016), https://books.google.fr/books?id=aItwAQAACAAJ
8. Moreaux, P., Sartor, F., Vernier, F.: An effective approach for home services management. In: 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). pp. 47–51. IEEE, Garching (February 2012). https://doi.org/10.1109/PDP.2012.45
9. Röcker, C., Ziefle, M., Holzinger, A.: Social inclusion in ambient assisted living environments: Home automation and convenience services for elderly users. In: International Conference on Artificial Intelligence (ICAI 2011). New York CSERA Press. pp. 55–59 (2011). https://doi.org/10.1007/978-3-319-66808-6_17

10. Singh, D., Kropf, J., Hanke, S., Holzinger, A.: Ambient assisted living technologies from the perspectives of older people and professionals. In: Machine Learning and Knowledge Extraction. Springer Lecture Notes in Computer Science LNCS 10410. pp. 255–266 (2017). https://doi.org/10.1007/978-3-319-66808-6_17
11. Vishwajeet, H.B., Sanjeev, W.: i-learning iot: An intelligent self learning system for home automation using iot. International Conference on Communications and Signal Processing (ICCSP) (April 2015). https://doi.org/10.1109/ICCSP.2015.7322825
12. Weyns, D., Schmerl, B., Grassi, V., Malek, S., Mirandola, R., Prehofer, C., Wuttke, J., Andersson, J., Giese, H., Goeschka, K.: On Patterns for Decentralized Control in Self-Adaptive Systems. In: de Lemos, R., Giese, H., Müller, H., Shaw, M. (eds.) Software Engineering for Self-Adaptive Systems II, Lecture Notes in Computer Science (LNCS), vol. 7475, pp. 76–107. Springer (January 2013), http://dx.doi.org/10.1007/978-3-642-35813-5_4
13. Yassein, M.B., Shatnawi, M.Q., Aljwarneh, S., Al-Hatmi, R.: Internet of things: Survey and open issues of mqtt protocol. In: 2017 International Conference on Engineering MIS (ICEMIS). pp. 1–6 (May 2017). https://doi.org/10.1109/ICEMIS.2017.8273112
14. Z-Vawe: Z-vawe aliance. https://z-wavealliance.org/ (2018), accessed: 2018-04-01
15. Zigbee: Zigbee aliance. http://www.zigbee.org/ (2018), accessed: 2018-04-01