WIoT: Interconnection between Wise Objects and IoT

Ilham Alloui, Eric Benoit, Stéphane Perrin and Flavien Vernier
Université Savoie Mont-Blanc - LISTIC, 5 Chemin de Bellevue, Annecy-le-Vieux, 74 940 Annecy, France

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

Abstract: Internet of Things (IoT) technologies remain young and require software technologies to ensure data/information management among things in order to deliver more sophisticated services to their users. In particular, users of IOT-based technologies need systems 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 provided through 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 common usage of a system and unusual behavior (habit change). We illustrate this through a particular case of home automation.

1 INTRODUCTION

The Internet of Things (IoT) is the extension of current Internet to provide connection and communication between devices or physical objects referred to as Things (IEC, 2016). 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 more sophisticated services to their users. As an example, 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 (Vishwajeet and Sanjeev, 2015).Users may need a remote access to things, for instance to switch off lights they forgot or turn off the strove... Communication provided by existing IoT technologies involves then 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 and then it would switch off them.
This implies that the system is able to: (a) identify changes in the way it is being used and (b) 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 those requirements.
Starting from works on IoT, on one hand and on intelligent software systems on the other hand (Weyns et al., 2013) (Brun et al., 2013) we aim at adding value to IoT through a software object framework we named WOF (Wise Object Framework) (Alloui et al., 2015) 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 allow IoT-based systems like in HA to: (a) identify common usage (i.e. usual behavior of their users); (b) consequently detect uncommon usage (unusual behavior); (c) adapt to the new usage (system in automatic mode) or simply give information to the users (manual mode).
Identifying common usage by software is not an easy work. The common usage is usually studied from a psychological point of view for the human (Aarts et al., 2006) or from the signal processing point of view with change detection methods (Aminikhanghahi and Cook, 2017) for data, but never from the software point of view at the best of our knowledge.This research issue raises many questions such as:

Alloui, I., Benoit, E., Perrin, S. and Vernier, F.
WIoT: Interconnection between Wise Objects and IoT.
DOI: 10.5220/0006870204940505
In *Proceedings of the 13th International Conference on Software Technologies (ICSOFT 2018)*, pages 494-505
ISBN: 978-989-758-320-9

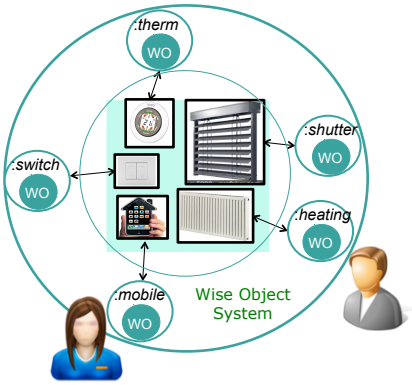


Figure 1: Example of Home Automation Wise Object System.

what is considered 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. Users' behavior identification and system adaptation rely on data/information collected from connected things that may be distributed as is the case in IoT. This led us to construct 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. In Section 2 we recall the concept of WO and WOF, the behavior of a WO, its initial interaction with others WO and 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 one in Section 3.2. 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. From a system development perspective, our design decisions behind the WOF are mainly guided by the following requirements: software support should be unintrusive, 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 depicted in 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/logical objects/things.

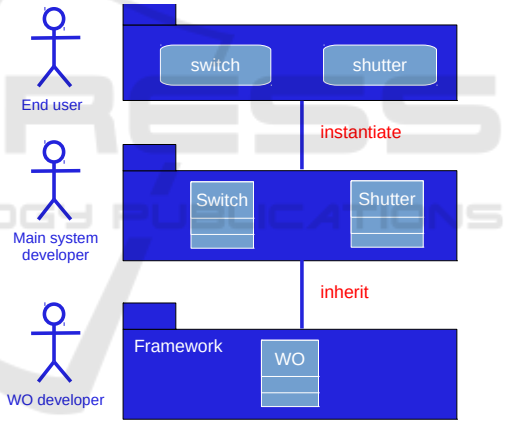


Figure 2: WOF concrete architecture.

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 (other WOs, 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 entity/device (e.g. a vacuum cleaner) or a logical entity (e.g. a software component). As an example, a wise object could be a cleaner that learns 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:

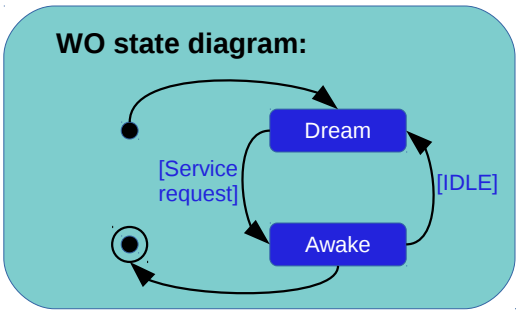
- its autonomy: it is able to behave with no human intervention;
- its adaptiveness: it changes its behavior when its environment changes;
- its intelligence: it observes itself and its environment, analyzes them and uses its knowledge to decide how to behave (introspection, monitoring, analysis, planning);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

A WO behaves transiting among two states: The dream state and the awake state, see Figure 3.

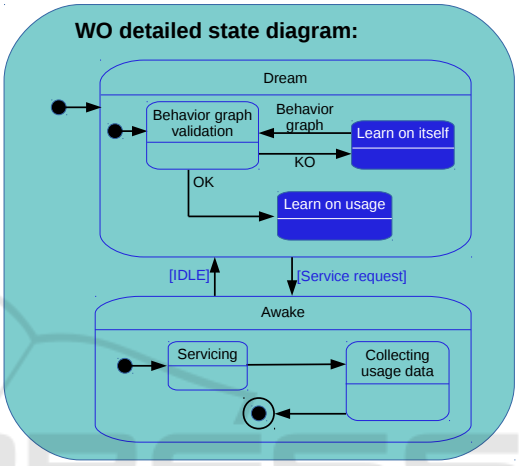
The dream state is dedicated to acquiring knowledge about its own capabilities and to analyzing usage-related knowledge. The awake state is the state where the WO executes its methods invoked by other objects (external service requests) or by itself (internal requests), and, monitors such execution while 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) to know the effect on the attributes of this sub-class. Each set of attribute values produces a state in the diagram and method invocation produces a transition. The main constraint in this step is that the method invocation must have no effect on the other objects of the application when the WO is dreaming. This is solved thanks to a bus-based system architecture described in (Alloui et al., 2015) 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 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 WO are a projection of its current usage with regard to its common (usual) usage. 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



(a) WO short state diagram



(b) WO detailed state diagram

Figure 3: UML state diagram of WO built-in behavior (Alloui and Vernier, 2017).

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 (Moreaux et al., 2012). 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 sub-class instance (application object). This consequently modifies the capability-related graph of this instance.

To build a WO system, the WOF provides communication bus (Gava) for the interaction between the WO instances. Interactions are managed through a manager object that establishes the configured pairing between events and actions according to a publish-subscribe pattern. Figure 4 illustrates this interaction. When a method is invoked on a WO instance: (a) the wise part of the instance sends 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

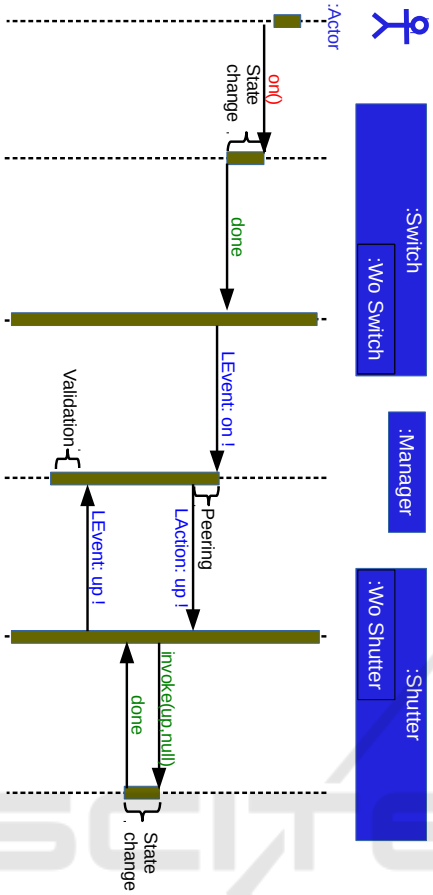


Figure 4: UML Sequence diagram of the interaction between a WO switch and a WO shutter.

manager checks that the order has been correctly executed. The communication and pairing system are detailed in (Alloui and Vernier, 2017) and were initially limited to communication and pairing between WO instances.

2.3 WOF and Data Analyzers

To manage its knowledge, to determine the common usage and to express its emotions, the WOF provides an analyzer connector to connect and to study different approach to compute the emotions. Each connected analyzer is waked-up during the dream step of the WO to analyze the last events.

We base our preliminary model of common usage of an object on a statistic approach and define the common usage as weaker forms of stationarity in the statistic point of view. Let us recall this approach introduced in (Alloui and Vernier, 2017).

Let $x(i)$ a continuous and stationary time random process. The weaker forms of stationarity (WSS) defines that the mean $E[x(i)]$ and variance $Var[x(i)]$ do not vary with respect to time and the autocovariance $Cov[x(i),x(i-k)]$ only depends on range k. A process is a WSS process if and only if:

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

This definition implies the analysis of the whole time series and a perfect stationarity. In our case, the common usage can change and is not perfectly stable. Therefore, we compute the stationarity – the common usage – on a sliding window of size w:

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

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 the system cannot be perfectly stationary, we relax the definition 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 word, when a new datum arrives at time $t+1$, its impact on the mean, variance and autocovariance gives known values in the window w. If one or more values are out of range, the behavior is not a common usage.

According to this definition of the stationarity, we define an emotion as the distance between the current usage and the common usage, in other words, the distance with the stationarity measure. We define this distance $d(x(i))$ by the following centered normalized scale where:

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

where

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

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

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

$j \in [t-w,t]$ and $\overline{E[x(j)]}$, $\overline{Var[x(j)]}$ and $\overline{Cov[x(j),x(j-k)]}$ 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 with the common usage between $t-w$ and t. If all values of the distance – $d(E[x(i)])$, $d(Var[x(i)])$ and $d(Cov[x(i),x(i-k)])$ – are in $[-1,1]$ this is considered as a common behavior, otherwise this is identified as a behavior change (unusual usage) relatively to the knowledge on the common usage.

3 WOF TOWARDS IoT

To meet IoT related requirements cited in Section 1, we extended our framework WOF (Alloui and Vernier, 2017) 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).

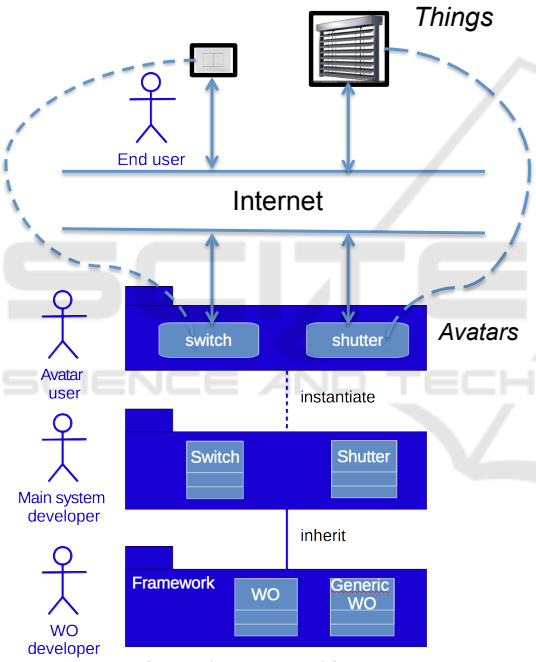


Figure 5: WIoT architecture.

A WO can be viewed as an avatar of a thing or a set of things. In this first approach, we limit the interconnection between a WO and only one thing (see Figure 5). From now on, when use the term object to refer to the thing-avatar peer.

3.1 WO Model for IoT

When a thing joins the application system (e.g. HA system), its corresponding avatar is automatically instantiated and this pair forms then a new object. This means that the avatar 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 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 their state changes. In the case of home automation, devices using ZigBee (Zigbee, 2018), Z-Wave (Z-Vawe, 2018) or other modern systems, communicate their capabilities through profiles or other kinds of descriptions. Figure 6 presents the UML Class diagram of WO including the Generic WO. As

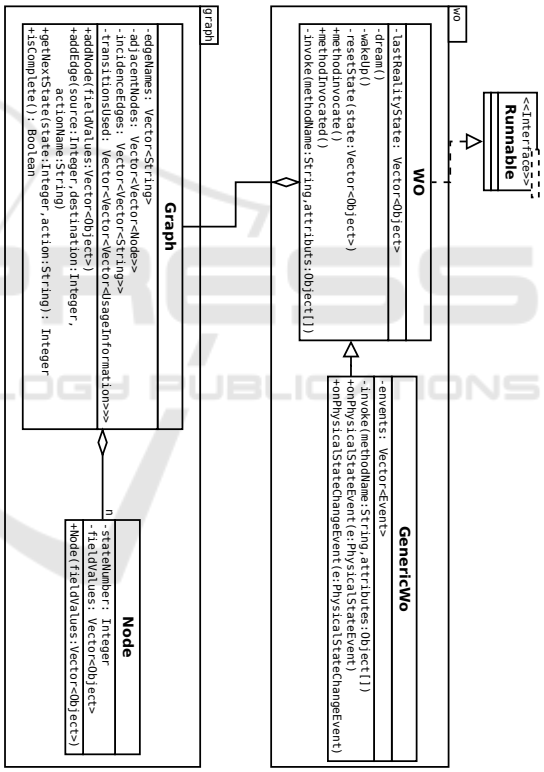


Figure 6: UML Class diagram of generic WO.

shown in the figure, a generic WO is a WO where the "invoke" method is redefined. Classically through the "invoke" method, a WO can invoke the methods of its sub-classes, i.e. application classes whose instances are avatars for things. On the opposite, 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 7 illustrates the communication flow be-

tween 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.

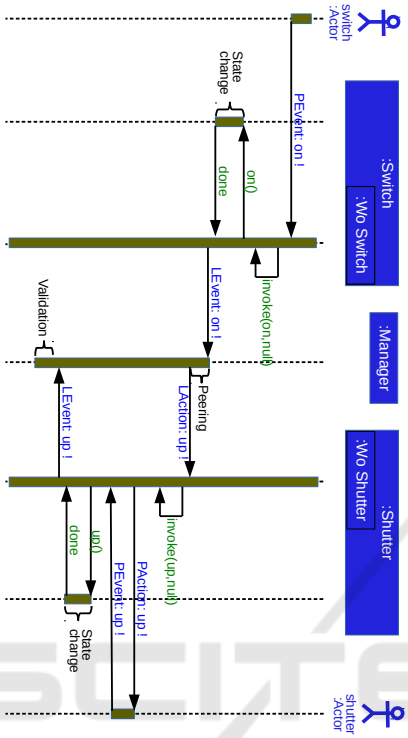


Figure 7: UML Sequence diagram of the interaction between a physical switch and a physical shutter.

When the switch is activated, it sends the message ”PEvent:on!” to its avatar. When receiving this message, the wise part of the avatar knows 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.

Let us note the system can manage pure logical objects: objects that are not linked to physical objects. Figures 8 and 9 illustrate 2 cases. The former, Figure 8, 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).

Figure 9, 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.

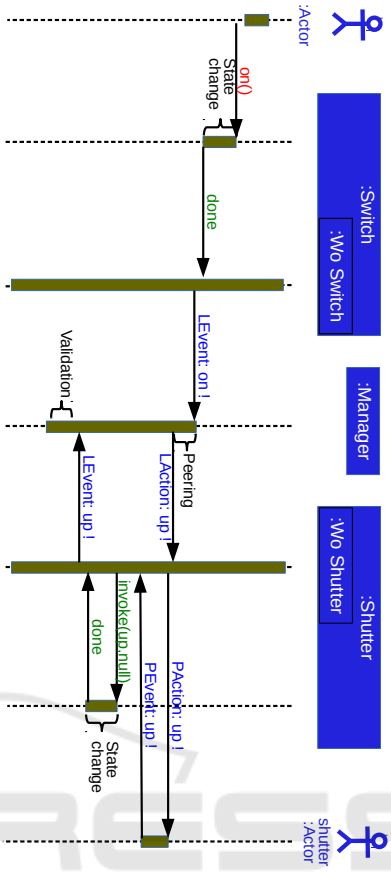


Figure 8: UML Sequence diagram of the interaction between a logical switch and a physical shutter.

Figure 10 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.

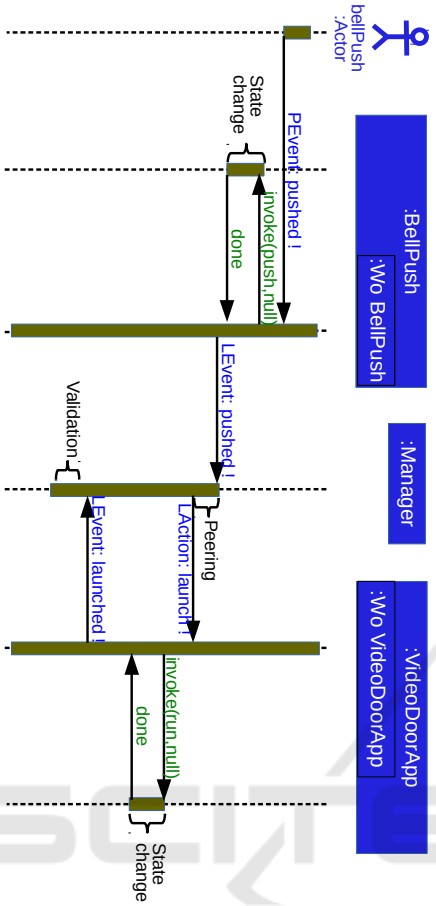


Figure 9: UML Sequence diagram of the interaction between a physical bell push and a logical video application.

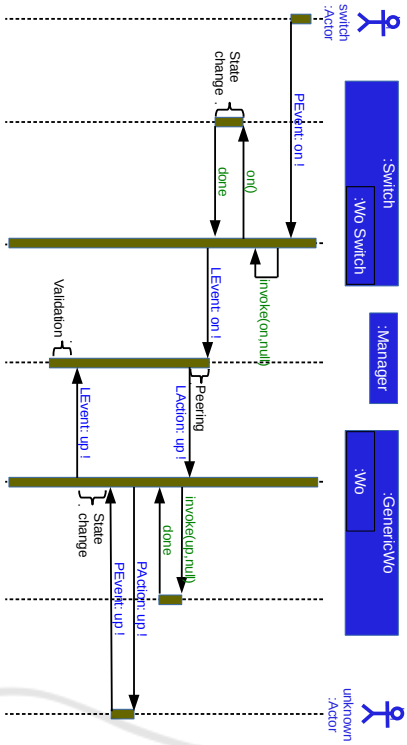


Figure 10: UML Sequence diagram of the interaction between a physical switch and a physical object not implemented as WO (no avatar) and managed as a generic WO.

3.2 Communication Protocol

The WOF provides a communication system for WOs to interact and exchange information. It corresponds to communications between objects 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 (Yassein et al., 2017). Thus we implemented a bridge between both those system 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 11 shows this communication flow among physical things (button and light), their logical avatars (software WOs) and the manager software object.

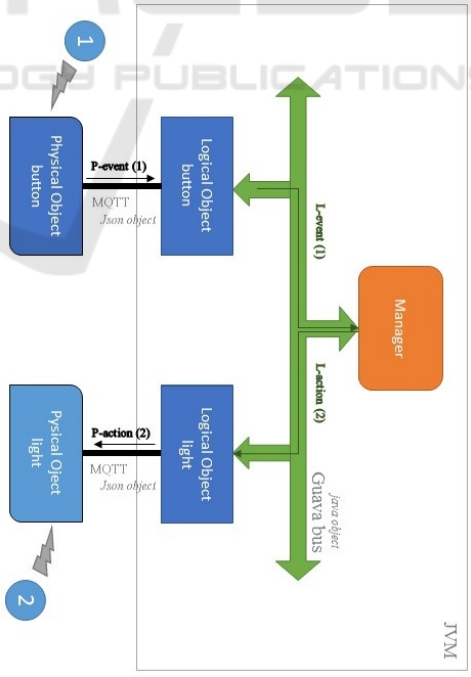


Figure 11: Communication flow between manager, logical and physical objects.

From the implementation point of view, the 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.

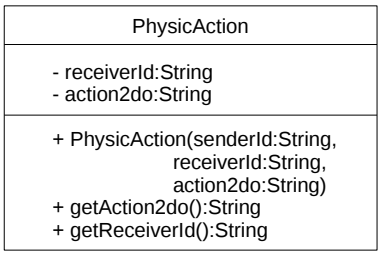
The bridge only translates Java object messages to JSON objects and vice-versa according to the following rules:

- 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.

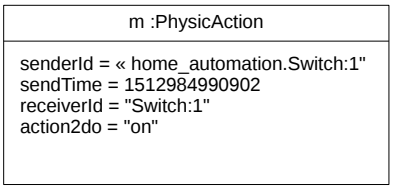
Figures 12(a) and 12(b) illustrate respectively the ”PhysicAction” class and an object that is translated as the following JSON message:

```
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”.



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



(b) Example of PhysicAction object

Figure 12: PhysicAction class used to receive ”PhysicAction” message by a WO.

Finally, the communication system we used 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 13 illustrates on a switch example the three communication cases.

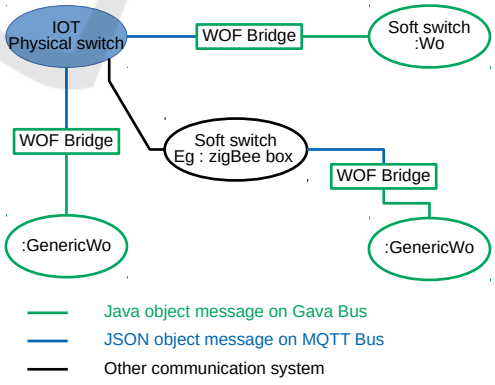


Figure 13: An IoT object, like a switch can be connected to WOF according to 3 ways: a) the thing can communicate using our MQTT protocol and its avatar exists in the WOF, b) the thing can communicate using our MQTT protocol but its avatar does not exist in the WOF, c) the thing cannot communicate using our MQTT protocol.

4 EXPERIMENTAL IMPLEMENTATION

4.1 Use Case Description

To illustrate the use of the proposed architecture solution which interconnects wise objects 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.

4.2 Experimental Results

Figures 14 and 15 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 provide an in-depth analysis of common usage modeling. This issue will be studied in the future.

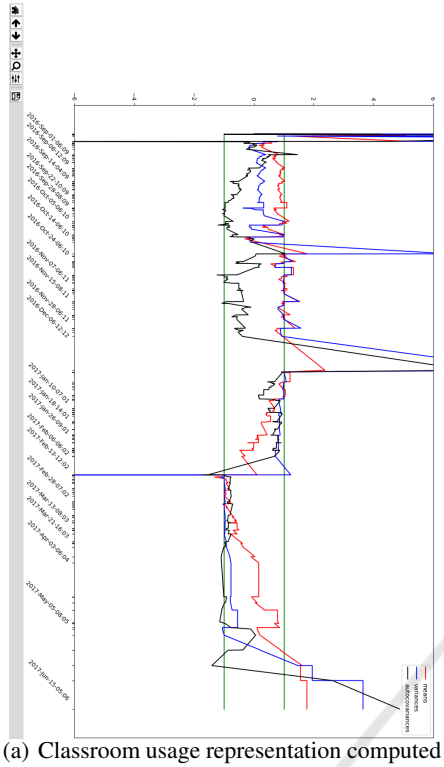
In this experiment, we observe for the sensor, the delay between events as well as the time spent by it in different states. The events are the detection of "new presence", when the sensor switches from "no presence state to "presence" state and conversely, the detection of "no more presence", when the sensor switches from "presence state to "no presence" state. Figures 14(a) and 14(b) give the common usage respectively computed from the "new presence" events and from the "no more presence" events. In other word, Figure 14(a), gives the common variation of delay between two successive "new presence" events. Figures 15(a) and 15(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):

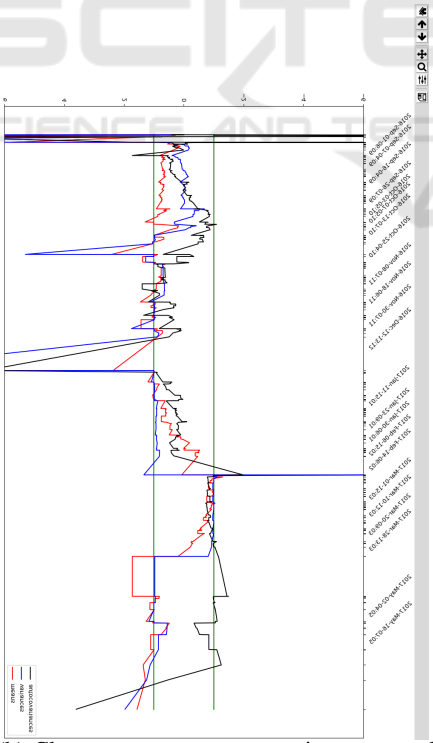
- 1 week for the Halloween holidays in October,
- 2 weeks for the Christmas holidays in December,
- 1 week for the winter holidays in February,
- 1 week for the Easter holidays in April and
- the end of the school year in June.

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 as it is. 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

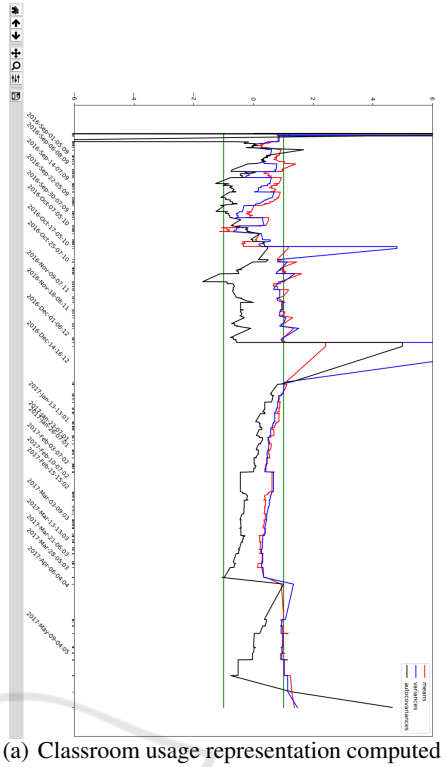


(a) Classroom usage representation computed from ”new presence” events

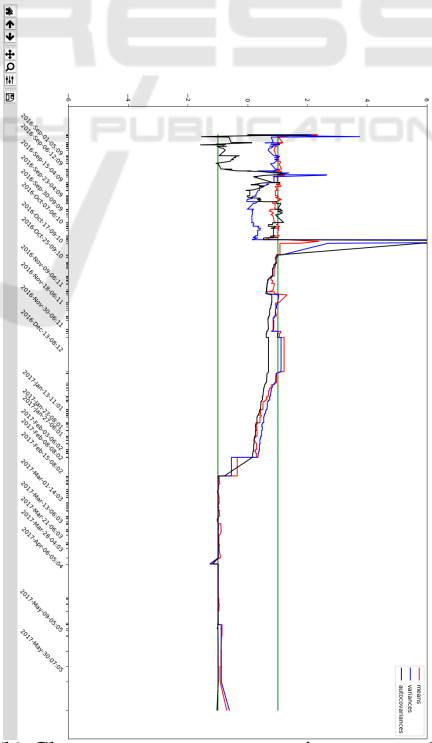


(b) Classroom usage representation computed from ”no more presence” events

Figure 14: Common usage and Emotion representation based on events.



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



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

Figure 15: Common usage and Emotion representation based on time spent in state.

event" must also be performed by the system. Both those points will be studied in future work.

5 CONCLUDING REMARKS AND FUTURE WORKS

With the growing IoT and user requirements for technology-based systems that adapt to their needs, we showed in this paper how to interconnect IoT to adaptive software systems, namely "wise systems".

We think that wise software systems could enhance IoT with useful capabilities such as learning, monitoring and adaptation to meet those requirements.

To do so, we enriched our software framework WOF (Wise Object Framework) (Alloui and Vernier, 2017) to provide IO Things, be them physical or software, with the necessary mechanisms for learning, monitoring, analyzing and managing data/information. We illustrated our approach on a home automation case study where things (a smart presence sensor in our case) are able to identify common usage and unusual behavior. This becomes possible thanks to the communication protocol we designed in WOF.

Each managed things of IoT is represented by its software avatar: a WO. This approach allows the system to learn on the common usage of any things connected to the home automation system.

The paper recalls the concepts of WO and describes the software communication structure for the interaction between WO and IoT. The main advantage of this structure is that it manages known or unknown things from IoT under the condition that the system communicates with the thing using the communication protocol we have defined. 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.

To highlight the learning capability of the system, an experiment on real data is presented. This experiment studies the common usage of a classroom using a smart presence sensor. Those results show, from different points of view, the changes in usage like weekends or vacations and this is based only on knowledge acquired from data. 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 (Röcker et al., 2011) and (Singh et al., 2017), 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. The next step for us is to be able to express emotions with a higher semantic level than the present one 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

Aarts, H., Verplanken, B., and Knippenberg, A. (2006). Predicting behavior from actions in the past: Repeated decision making or a matter of habit? *Journal of Applied Social Psychology*, 28(15):1355–1374.

Alloui, I., Esale, D., and Vernier, F. (2015). Wise Objects for Calm Technology. In *10th International Conference on Software Engineering and Applications (ICSOFT-EA 2015)*, ICSOFT-EA 2015, pages 468–471, Colmar, France. SciTePress 2015.

Alloui, I. and Vernier, F. (2017). 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*, to appear.

Aminikhanghahi, S. and Cook, D. J. (2017). A survey of methods for time series change point detection. *Knowl. Inf. Syst.*, 51(2):339–367.

Brun, Y., Desmarais, R., Geihs, K., Litoiu, M., Lopes, A., Shaw, M., and Smit, M. (2013). A design space for self-adaptive systems. In de Lemos, R., Giese, H., Müller, H. A., and Shaw, M., editors, *Software Engineering for Self-Adaptive Systems II*, volume 7475 of *Lecture Notes in Computer Science*, pages 33–50, Dagstuhl Castle, Germany. Springer.

IEC (2016). *IoT 2020: Smart and Secure IoT Platform : White Paper*. International Electrotechnical Commission.

Moreaux, P., Sartor, F., and Vernier, F. (2012). An effective approach for home services management. In *20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 47–51, Garching. IEEE.

Röcker, C., Ziefle, M., and Holzinger, A. (2011). 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*, pages 55–59.

Singh, D., Kropf, J., Hanke, S., and Holzinger, A. (2017). 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*, pages 255–266.

Vishwajeet, H. B. and Sanjeev, W. (2015). i-learning iot: An intelligent self learning system for home automation using iot.

Weyns, D., Schmerl, B., Grassi, V., Malek, S., Mirandola, R., Prehofer, C., Wuttke, J., Andersson, J., Giese, H., and Goeschka, K. (2013). On Patterns for Decentralized Control in Self-Adaptive Systems. In de Lemos, R., Giese, H., Müller, H., and Shaw, M., editors, *Software Engineering for Self-Adaptive Systems II*, volume 7475 of *Lecture Notes in Computer Science (LNCS)*, pages 76–107. Springer.

Yassein, M. B., Shatnawi, M. Q., Aljwarneh, S., and Al-Hatmi, R. (2017). Internet of things: Survey and open issues of mqtt protocol. In *2017 International Conference on Engineering MIS (ICEMIS)*, pages 1–6.

Z-Vawe (2018). Z-vawe aliance. https://z-wavealliance.org/. Accessed: 2018-04-01.

Zigbee (2018). Zigbee aliance. http://www.zigbee.org/. Accessed: 2018-04-01.