Computing
https://doi.org/10.1007/s00607-022-01113-1

REGULAR PAPER

Towards dynamic and heterogeneous social IoT environments

Daniel Flores-Martin[1] · Javier Berrocal[1] · José García-Alonso[1] · Juan M. Murillo[1]

Received: 27 January 2022 / Accepted: 18 August 2022

Abstract
Nowadays, there are millions of smart devices connected to the Internet. The purpose of these devices is to make people's lives easier. Thanks to the collaboration among them, the possibilities that the Internet of Things brings can grow exponentially. However, many manufacturers develop closed protocols and devices to protect their market share, limiting in many ways this collaboration. This paper presents a conceptual architecture that improves the proactive collaboration between IoT devices regardless of the protocols developed by their manufacturers. This architecture aims to identify entities in smart environments, describe their features and interfaces, and identify strategies fostering their collaborations. As a result, devices from different manufacturers can communicate to create a collaborative environment in a simple, efficient, and affordable way. This architecture has been evaluated in a real and a simulated environment, to validate its feasibility and efficiency.

Keywords Internet of Things · Collaboration · Social · Behavior adaptation · Conceptual architecture

Mathematics Subject Classification 68U35

Daniel Flores-Martin, Javier Berrocal, José García-Alonso and Juan M. Murillo have contributed equally to this work.

✉ Daniel Flores-Martin
dfloresm@unex.es

Javier Berrocal
jberolm@unex.es

José García-Alonso
jgaralo@unex.es

Juan M. Murillo
juanmamu@unex.es

1 Departamento de Ingeniería de Sistemas Informáticos y Telemáticos,, Escuela Politécnica, Universidad de Extremadura, Avda. de la Universidad, S/N, 10003 Cáceres, España

Published online: 17 September 2022

1 Introduction

The goal of the Internet of Things (IoT) is to make people's lives simpler. IoT can be used in different domains such as smart homes, automotive, and healthcare [1]. The evolution of this technology allows developers to create increasingly useful devices and services that a few years ago were unthinkable.

The real potential of IoT comes from the collaboration among smart devices to perform complex tasks. The next evolution of the IoT is to ensure that smart devices can proactively collaborate [2, 3]. Unfortunately, this is still far from being realized. Indeed, manufacturers develop their own protocols, which means that their devices can collaborate among themselves but are unable to integrate with devices from other manufacturers. This not only limits the ability of devices from different manufacturers to collaborate, but also inevitably leads to the well-known *vendor lock-in* problem [4]. This phenomenon implies that if one wants to obtain the maximum benefit from the IoT, they must purchase devices from the same manufacturer to ensure maximum compatibility.

In addition, devices are designed to operate in a specific application domain, such as smart home, industry, or healthcare. The collaboration is much more difficult to achieve across different domains. This is due to little or no interoperability caused by the type of devices, the technologies used or the domain for which they are intended [5].

Currently, there are some commercial platforms, such as Alexa [1] or Home Assistant[2] that define mechanisms for supporting the collaboration between devices from different manufacturers. However, this support is still limited by the commercial interest of each platform and, more importantly, it is static due to the predefined rules. This is especially relevant in social and open environments in which the available devices can change dynamically. In addition, their behavior must adapt to people's preferences (such as temperature, lighting conditions, or anything that an IoT device can do). "Social Computing" addresses this adaptability with computer systems to create social conventions through the use of technology and software [6]. More concretely, "IoT Social Environment" integrates IoT devices as part of the core of these environments as a mean for achieving the desired behavior. The "Situation Awareness" is a conceptual term focused on considering the contextual information, such as the number of devices nearby, the people involved, and other spatio-temporal factors to adapt the IoT devices behavior [7, 8]. Therefore, new mechanisms are required to enable devices to anticipate continuous changes in IoT social environments and achieve a dynamic adaptation and collaboration.

This work proposes a solution whose purpose is to promote the proactivity and collaboration among smart devices. As a result, the smart devices will be aware of the environmental situation in which they find themselves and the contextual properties which will enable them to adapt their behavior. The main contributions of this work are:

[1] https://developer.amazon.com/en-US/alexa.

[2] https://www.home-assistant.io.

- A conceptual architecture for improving interoperability in IoT environments that addresses the coordination of both people and smart devices. This architecture details the components needed to integrate devices and people, facilitate their communication, consider the changing needs of the environment, and solve specific situations.
- A functional open-source prototype of the conceptual architecture. This prototype is detailed by defining its components, the technologies used and the information flow.
- The prototype has been evaluated to analyze the performance, the quality of experience and its scalability.

To achieve these objectives, we address complementary tasks: an analysis of the state-of-the-art of the interoperability in IoT social environments; and the extension of a standard to describe the properties of smart devices and people to promote their integration in situation-aware systems.

The rest of this paper is structured as follows. Section 2 addresses the problems that must be overcome for achieving a dynamic collaboration in IoT environments. Then, Sect. 3 details the proposed conceptual architecture, while an implementation is constructed in Sect. 4. Next, in Sect. 5, architecture is validated. The related work are detailed in Sect. 6. Finally, in Sect. 7, the conclusions are drawn and future work is sketched.

2 Dynamic collaboration in the IoT

The motivation to define levels of interoperability between devices is present within the scientific community, where different levels of interoperability are suggested by the current literature to obtain an ideal IoT ecosystem. Some of these levels are [9]: *technical*, *syntactical*, *semantical* and *organisational*; [10]: *devices*, *network*, *syntactical*, *semantic* and *platform*; [11]: *basic connectivity*, *network* and *syntactic*; [12]: *technical*, *semantic*, *syntactic*, and *cross-domain*. These levels generally overlap although with different nomenclatures.

Each work proposes a series of levels to try to cover different aspects of interoperability. However, as far as the authors know, there are still no proposals that address dynamic interoperability in social environments. A new interoperability level can be defined to focus on the evaluation of the state of the environment and where different situations can arise, namely the *situational level*. A situation is a spatio-temporal conceptual grouping of entities (people or devices) that pursues a common objective, and which is achieved by the collaboration among entities. For instance, if in a smart workplace two employees have a certain preference with the temperature, the air conditioner should proactively identify it, reach an agreement, and try to satisfy it. The situational level should reuse the existing interoperability levels to achieve a dynamic collaboration [13]. Below a pyramid (Fig. 1) that group the existing interoperability levels and the addition of the situation level is shown.

1. Technology: the diversity of manufacturers makes communication between devices from different manufacturers difficult, because each manufacturer devel-

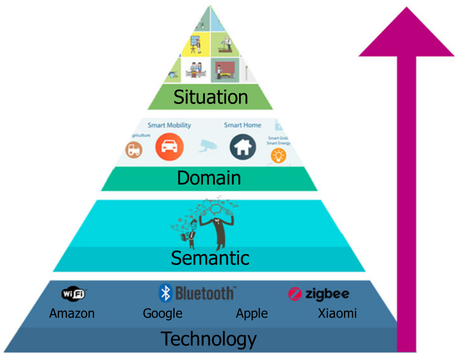


Fig. 1 Please process this figure in center and with the caption in the bottom (refer other figures)

ops its own communication mechanisms, protocols, and technologies. This layer improves the interoperability through communication protocols.

2. Semantic: once the communication is established, it is necessary to know the semantic description of the devices. This, which must specify in a clear and understandable way what information, services, or parameters they have. Smart devices can have similar characteristics or provide similar services. However, we still must get all their details to know how to interact with them.
3. Domain: devices designed for a specific domain should be reused to perform other complementary tasks and to interact with devices from different IoT domains. To improve this interaction, the benefits of the semantic layer are considered, such as the definition of device schemes or the use of query or reasoning languages to establish relationships.
4. Situation: detecting a particular situation of the environment and its characteristics is key to detail how devices should collaborate to achieve the environment goals. Even when we know the services provided by a specific device, its domain and its semantics, each situation requires these services to work in a specific way. Hence, it is necessary to be aware of the context and its attributes that define different situations. These factors can be people in the environment, IoT devices, when and where the situation occurs or what objectives are being pursued.

The first three levels of the interoperability have been widely addressed by the scientific community. In *technological* interoperability, solutions such as [14, 15] are developed to allow devices to be interconnected. These solutions are usually dedicated gateways to facilitate the connection through the invocation of services or microservices. There are also standards defined, for instance, by the IEEE [16] which are being introduced to facilitate the connection and exchange of data among devices. Also, *semantic* interoperability is being increasingly addressed in [17, 18] through technologies such as the Semantic Web and the use of ontologies and semantic reasoners. As for the *domain* interoperability, there have been some works to allow devices belonging to different domains to connect with each other [19, 20]. This is achieved through a complete description of the devices and the use of techniques such as the Semantic Web to establish a coherent relationship between them.

While these levels improve the IoT interoperability, the management of different situations has not yet been fully addressed. In this work we propose a conceptual architecture that improves the interoperability among devices once the situation of the environment in which they are involved is reached.

3 Conceptual architecture for managing situations

To support the above-mentioned *situation level* a conceptual architecture is proposed. It is assumed that a smart social environment will possess entities consisting of IoT devices or people, that can give rise to different environmental situations. These entities have goals that must be agreed upon and solved. These goals are the desired effects on the environment such as setting a specific temperature, lighting level, or type of music. The entities also have skills that can be combined to solve them. These skills refer to actions that can cause a change in the environment at a particular time, such as an air conditioner that can change the temperature, a lamp that can modify the brightness or a stereo that can change the type of music. This architecture aims to support the definition of all these elements, their integration, and coordination.

3.1 Conceptual architecture description

The proposed conceptual architecture (Fig. 2) contemplates that in smart scenarios, different entities can communicate and exchange data to know the situation in which they are and be able to proactively perform actions to achieve common goals. Depending on the capabilities and tasks to be performed by the devices, two types of roles are considered in the architecture: entities and controllers. Entities are devices that can sense and change the status of the environment or that have some needs or preferences. Controllers are devices that can request, receive and process information to identify what actions to execute in order to meet these needs. We note that a smart device can have both roles if it has enough computing capabilities. In the following, we separate them in order to improve the readability.

On the one hand, entities E_n obtain the values from the environment such as temperature or brightness (sensors), and perform actions to change their state (actuators). In addition to this information, entities possess crucial information that must be shared for the correct management of the interoperability, such as the identifier, the brand, the model, their characteristics, goals/objectives, previous interactions, etc. To share this information, it is grouped in a wrapper that encapsulates what can be shared with different controllers when needed.

On the other hand, the controller is a device with enough computational capacity to process all the information obtained from the entities and decide which strategy (set of actions) should be triggered depending on the desired goals of entities. Initially, the architecture assumes that the controller is physically located in the environment. However, the controller can also reside in the cloud or edge environments. To manage the interactions among entities, the controller is composed of different components:

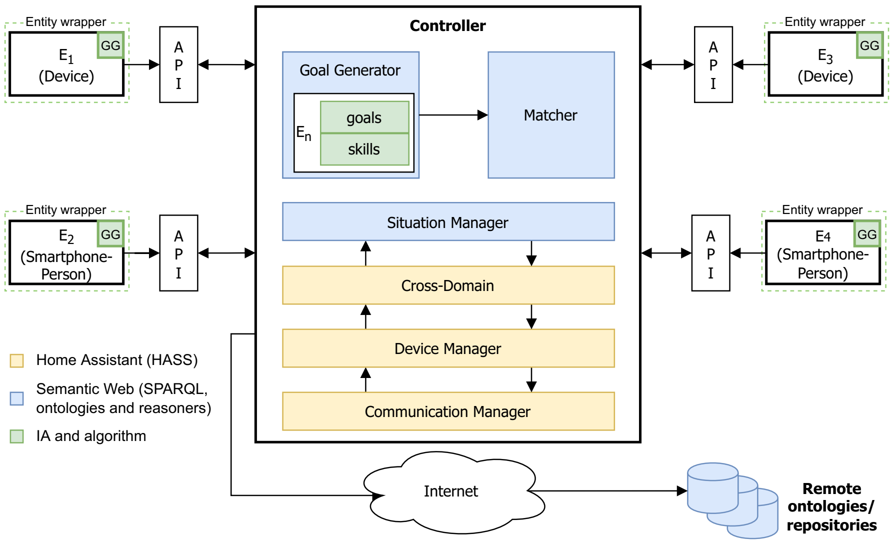


Fig. 2 Conceptual architecture

- Communication Manager: it is responsible for achieving physical interoperability among entities. In this layer, multiplatform integration tools, such as Home Assistant (HASS)[3] or OpenHUB,[4] allow connections at the network level, for example, to connect an entity *lamp* through WiFi, Bluetooth or Zigbee. This allows entities to be discovered and incorporated into the network environment to, later, interpret the information provided by each entity depending on the situation. Thus, the technological level is addressed. Entities can be discovered in the environment in several ways, such as, communication or network protocols. Once discovered, communication between them is enabled, so that they can exchange information, which is discussed in the next component.
- Device Manager: the devices are managed to get to know the existing entities in the environment and their characteristics. This information will allow determining the type of entity, which goals are pursued by each entity, and which skills are available to solve the needs of other entities. Each entity must be able to define its information in some way. Usually, each device has a typical document not written in machine language where its characteristics, functionalities, or configuration parameters are specified. This document is used as documentation or manual but not to improve interaction. It is static and can not be used to describe properties changing according with the context. The document can be provided by the manufacturer or generated following a specific specification. In this work we will use the W3C Thing Description specification as detailed below. Therefore, this component is responsible for interpreting this document to give it a semantic connotation

[3] https://www.home-assistant.io.

[4] https://www.openhab.org.

and make it easier to relate entities, regardless of the domain, as detailed in the following component.

- Cross-Domain: this component establishes relationships and provides dynamicity among different application domains. This requires the translation of data formats or languages in which device information can be specified to ensure data availability. For this purpose, content filtering and cleaning techniques can be used. So, it ensures these relationships hold regardless of the technology or language used in the domain to which an entity belongs. These relationships are achieved through the semantic connotation obtained from the information of each entity, so that objectives and skills can be related as precisely as possible. The Semantic Web makes it possible to establish these relationships in a more or less simple way, as we will see below. Based on these relationships, the different situations that may occur in the environment are identified or created.
- Situation Manager: it identifies if a situation is already known in order to apply strategies that were successful in the past, or create new strategies. For this identification, contextual properties are used, such as the people present in the environment, the installed devices, or spatio-temporal data. This component uses the *communication manager* component to discover the devices, the *device manager* component to know how they interact, and the *cross-domain* component to know which ones can cooperate. This component is the most relevant component of the work and where the greatest efforts are applied, and related directly to the situation level.
- Goal Generator (GG): entities can have goals. They may be predefined by the manufacturer, manually defined by the user, or may be inferred by the presented architecture depending on previous interactions. This component is responsible for analyzing the previous interactions to discover the entities' goals. With the information provided about the situations, these goals can be generated. The goals can be of any type, such as turning on a lamp, increasing the temperature, recommending a certain product, etc. The generation of these goals facilitates their subsequent resolution. It is important to note that this component can be deployed in the controller, providing support to other low powered entities, or can be deployed on the entities when they have enough computing capabilities, as shown in Fig. 2.
- Matcher: once the objectives have been generated, this component identifies at run time which ones can be achieved using the skills offered by the entities. For example, if a goal has been generated to increase the brightness of the room, and there is a lamp that can perform that function, these two entities are related to reach that goal. This can be done by invoking the service of an entity that allows reaching the desired environment state. Thus, the different strategies to be carried out are established so that they can be carried out if the situation is detected again in the future. This must be done in real-time since the entities enter and leave the environments continuously.

These components can be implemented through different technologies. The following sections will detail the technologies selected for this work and provide the most important technical details. Let us describe an example to better define the interactions between these components. Imagine a lamp that can change the intensity and the color of its light. Thanks to the *Communication Manager*, the lamp can communicate with other entities, and the *Device Manager* is responsible for providing a global view of the environment. This view includes information about this particular device, such as its ID, communication interfaces, functionalities, etc. Furthermore, the user needs to integrate the lamp in a smart home where there are other smart devices such as an oximeter, to measure the amount of oxygen in the blood, and an audiometer, to measure the quality of hearing. Both devices belong to the healthcare domain, where data privacy and security are crucial. To provide a truly useful service, it would be interesting if the healthcare devices could send certain information to allow the lamp to change its light color depending on the results after a measurement. The *Cross-Domain* is responsible for enabling the exchange of information with other entities regardless of the domain. In addition, the lamp should also be able to change its intensity depending on the preferences of the people. The *Situation Manager* is in charge of identifying the situation that is occurring and determining if it should change the color or the intensity. The *Goal Generator* identifies if the people present have a lighting goal that must be satisfied. Finally, the *Matcher* finds and triggers the correct strategy.

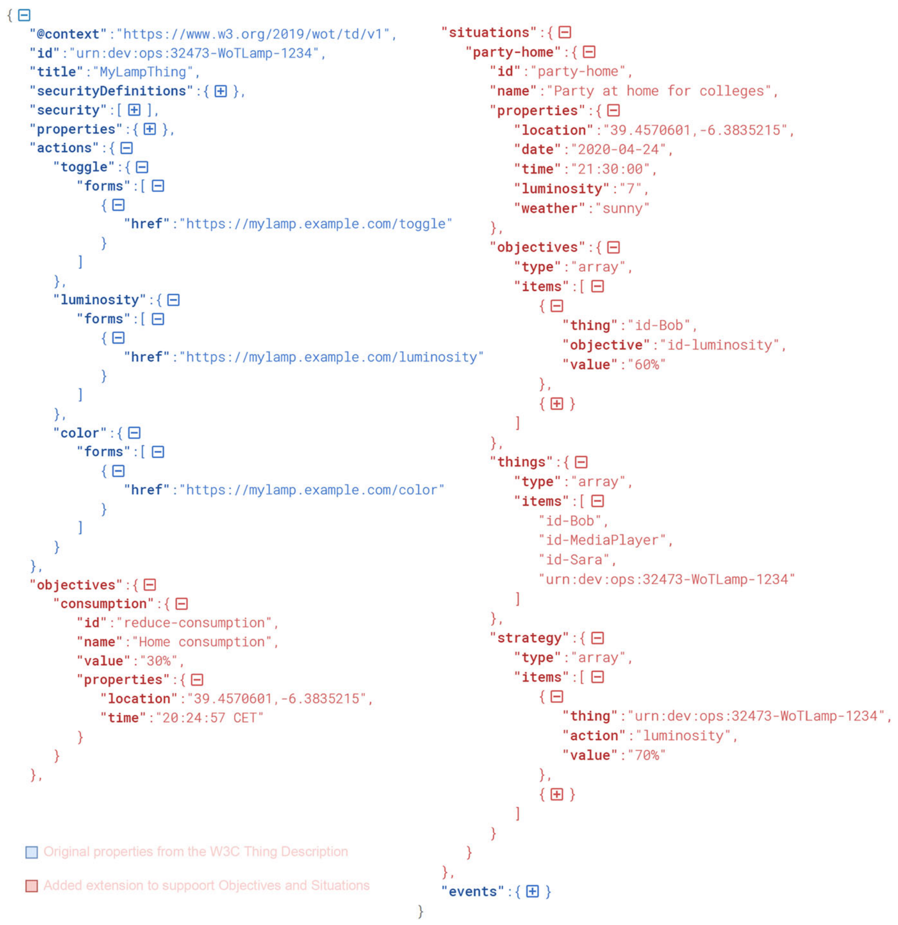
4 SMOTE: (S)ituation (M)anagement f(O)r Smar(T) (E)nvironments

SMOTE is an implementation of the proposed architecture for the management of situations in IoT environments. First, the entities according to the information provided in their wrappers are described. Second, the operations of the controller are specified, and its implementation is detailed. Finally, the data flow of the process from the initial communication of the entities until the run-time selection of the services to adapt the environment is shown.

4.1 Entity description

The information from the entities must be provided to the other entities and the controller must be analyzed and processed. This information is called entity description. The description of the entities follows the extension of the format proposed by the W3C, called Thing Description (W3C-TD) [17]. The W3C-TD is presented as a solution to counteract fragmentation in the Web of Things (WoT). It allows the developer to define smart devices using an standard based on the format JSON-LD (JavaScript Object Notation for Linked Data), detailing their id, title, security or properties. However, the W3C-TD does not provide support for collaboration between entities [21]. Therefore, we propose to extend it with two sub-classes to support the definition of objectives and situations:

- Objectives: it details the objectives (goals) of the entity with regards to the environment. Each goal has its identifier, a name, the desired value, and contextual properties that define it, such as spatio-temporal data (time, location, etc.).
- Situations: it defines the situations in which the entity was involved. The information related to the situations is composed by an identifier, a name, a title, the contextual properties specifying the spatio-temporal data of the situation and



```
{ ⊟
   "@context":"https://www.w3.org/2019/wot/td/v1",
   "id":"urn:dev:ops:32473-WoTLamp-1234",
   "title":"MyLampThing",
   "securityDefinitions":{ ⊞ },
   "security":[ ⊞ ],
   "properties":{ ⊞ },
   "actions":{ ⊟
      "toggle":{ ⊟
         "forms":[ ⊟
            { ⊟
               "href":"https://mylamp.example.com/toggle"
            }
         ]
      },
      "luminosity":{ ⊟
         "forms":[ ⊟
            { ⊟
               "href":"https://mylamp.example.com/luminosity"
            }
         ]
      },
      "color":{ ⊟
         "forms":[ ⊟
            { ⊟
               "href":"https://mylamp.example.com/color"
            }
         ]
      }
   },
   "objectives":{ ⊟
      "consumption":{ ⊟
         "id":"reduce-consumption",
         "name":"Home consumption",
         "value":"30%",
         "properties":{ ⊟
            "location":"39.4570601,-6.3835215",
            "time":"20:24:57 CET"
         }
      }
   },
   "situations":{ ⊟
      "party-home":{ ⊟
         "id":"party-home",
         "name":"Party at home for colleges",
         "properties":{ ⊟
            "location":"39.4570601,-6.3835215",
            "date":"2020-04-24",
            "time":"21:30:00",
            "luminosity":"7",
            "weather":"sunny"
         },
         "objectives":{ ⊟
            "type":"array",
            "items":[ ⊟
               { ⊟
                  "thing":"id-Bob",
                  "objective":"id-luminosity",
                  "value":"60%"
               },
               { ⊞ }
            ]
         },
         "things":{ ⊟
            "type":"array",
            "items":[ ⊟
               "id-Bob",
               "id-MediaPlayer",
               "id-Sara",
               "urn:dev:ops:32473-WoTLamp-1234"
            ]
         },
         "strategy":{ ⊟
            "type":"array",
            "items":[ ⊟
               { ⊟
                  "thing":"urn:dev:ops:32473-WoTLamp-1234",
                  "action":"luminosity",
                  "value":"70%"
               },
               { ⊞ }
            ]
         }
      }
   },
   "events":{ ⊞ }
}
```

🟦 Original properties from the W3C Thing Description

🟥 Added extension to suppoort Objectives and Situations

Fig. 3 Yeelight bulb description with the W3C-TD extension

the values of the solved goals (such as luminosity), and the strategies triggered depending on the detected goals. This information is later used to trigger the same strategies if they were successful, and to automatically infer new goals or changes in the ones already detailed.

Figure 3 shows an example of the description of a Yeelight smart light that will later be used in the case study. This description is a file specified according to the W3C-TD discussed above (blue). It shows the information of the entity and the actions it has. The description also includes the two sub-classes specified above (red). On the one hand, the lamp has a goal to reduce its consumption. On the other hand, a situation is specified where the lamp has been previously involved in a party at home. This situation specifies contextual information, the objectives to be solved, the entities involved, and the strategy conducted.

In the case of people, the description is stored on their mobile device and will be consulted and modified according produced interactions. In the case of IoT devices, the description could be provided by the manufacturer. The interpretation of this information is done through different techniques discussed below to detect the objectives and to determine how they can be achieved. Therefore, each entity is responsible for creating and storing its description.

4.2 Controller implementation

The controller is used to request and process the descriptions coming from the entities, to manage the interaction among entities, and to detect situations and trigger actions associated to them. In what follows, we present the implementation of the controller according to the components specified above and provide the key technical details. The pseudo-code for the whole process can be seen in Algorithm 1 while its notation is described in Table 1.

- Communication Manager, Device Manager, and Cross-Domain: as discussed in Sect. 2 and Sect. 3, there are technologies that support these components. Therefore it has been decided to use an existing framework. The Home Assistant is the software in charge of managing the integration of the entities and addressing the three first levels of interoperability. We use this platform because it can support the first layers with minor modifications, allowing us to focus on the other layers. HASS supports a large amount of brands ,[5] and allows the integration of devices (sensors, actuators, mobile phones, etc.) through a large number of different protocols like Bluetooth, WiFi, or ZigBee. Through the API provided by each entity, HASS can establish communications and request their descriptions (*INIT()*). These descriptions allow HASS to identify the type of the entity and get its extended W3C-TD description, with information related to the situations and objectives stored in the entity. Also, some modifications were performed to enable automatic situation discovery when a new entity is detected in the environment. This has been achieved by requesting and processing the description of an entity when it is detected in the environment (*PARSEPDESCRIPTION(D)*). This is the basis for detecting and managing devices required by the following components.
- Situation Manager: a module has been implemented (*CHECKSITUATIONS(O)*) that can identify situations by taking into account the following parameters: the entities present in the environment, the objectives to be achieved, and spatio-temporal factors (location, date, and time). In this way, a situation is identified as known if these parameters are met, and its associated strategy would be launched. Otherwise, it would be a new situation and the required process would be conducted to elaborate the strategy and trigger it. These parameters are adjustable depending on the application.
- Goal Generator: from the *skills* that the entities have, as well as the goals of each situation, this component is in charge of identifying goals of each entity with the environment. This is done based on an entity's interactions with the environment and providing them with a complimentary name to be easily linked. This is done using a system proposed by the authors in [22] that allows us to predict interactions

[5] https://www.home-assistant.io/integrations.

Table 1 Notation of the basic elements for the Algorithm 1

Element	Description
O	Defines the ontology where the entities will be stored. In this case: skeleton.owl
E	Entities in the environment: devices or people (represented by their smartphones)
S	Situation that contains information about the environment. It is associated with an Entity
X(i)	Information. It could be referred to a different object (X) such as entities or situations
X(o)	Objectives (goals). It could be referred to a different objects (X) such as entities or situations
X(a)	Actions (skills). It could be referred to a different object (X) such as entities or situations
X(p)	Properties. It could be referred to a different object (X) such as entities or situations
X(s)	The different situations that an Entity or the Ontology posses
D	Corresponds to the Entity description written following the W3C-TD extension

with IoT devices in a smart environment through a neural network. In general terms, the network has as many input neurons as features have the entities. The number of outputs offered by the network is equal to the number of different skills that exist on the devices. When new devices are added, the number of outputs also changes, and the network is redefined to infer new goals.

- Matcher: Semantic Web technologies are used for this purpose (*MATCHOBJECTIVES(O,S)*). Through ontologies and SPARQL queries we analyze what actions and objectives are present in the environment [23]. Currently, actions and objectives are linked by name. Each action and objective has a prefix, "sk_" and "g_" respectively, with which they can be identified and related. For example: "sk_illumination" and "g_illumination". In addition, in the case of similar names, the semantic reasoner is in charge of trying to establish this relationship. For example, "sk_illumination" and "g_brightness". When an action associated with an objective is found, it is performed to achieve the objective. E.g. "The room is a bit dark, I need to increase the brightness to level 7" (objective - "g_illumination"), by setting the light bulb to that brightness level (action - "sk_illumination"). With these pieces of information we can identify situations. If the situation is known, it is enough to invoke the services associated with the strategy (*TRIGGERSTRATEGY(S)*). If not, a new situation is generated with the contextual information of the environment, the entities involved, and the associated strategy to be formulated. Then, this situation is sent to all the entities in the environment to be stored in their descriptions. The use of the Semantic Web can be combined with AI techniques [24] to perform a different matching according to the requirements of the application. In fact, AI is already used in the previous component for goal generation.

Algorithm 1 Environment processing

```
Function INIT():
    $O \leftarrow$ loadOntology(skeleton.owl)
    $name \leftarrow$ "w3ctde.daniel" $\leftarrow$ detectedEntity
    $D \leftarrow$ requestDescription($name$)
    PARSEPDESCRIPTION($D$)

Function PARSEPDESCRIPTION($D$):
    $E \leftarrow$ new Entity ;                                   // New thing
    $E(i) \leftarrow$ D(i) ;                                      // Entity information
    $E(o) \leftarrow$ D(o) ;                                      // Entity objectives
    $E(a) \leftarrow$ D(a) ;                                      // Entity actions
    $E(s) \leftarrow$ D(s) ;                                      // Entity situations
    file $\leftarrow$ E.present(1) ;                              // Set entity as present
    ...
    O.add($E$)
    CHECKSITUATIONS($O$)

Function CHECKSITUATIONS($O$):
    $S \leftarrow O(s)$
    for all $S$ do
        if $S(p)$ = True then
            $properties \leftarrow$ True
        end
        if $S(e)$ in $O$ then
            $entities \leftarrow$ True
        end
        ...
        if $properties$ = True and $entities$ = True then
            TRIGGERSTRATEGY($S$)
            $situation \leftarrow$ True
        end
    end
    if $situation$ = False then
        MATCHOBJECTIVES($O$,$S$)

    end
Function TRIGGERSTRATEGY($S$,):
    $actions \leftarrow S(A)$
    for all $actions$ do
        call(action($i$), action($value$))
    end
Function MATCHOBJECTIVES($O$,$E$):
    $S \leftarrow$ new Situation
    $S(p) \leftarrow$ new Properties
    $objectives \leftarrow E(o)$
    $action \leftarrow E(a)$
    for all $objectives$ do
        $S(o)$.add($objectives(i)$) $action \leftarrow O$.search(iri = $objectives(i)$.name, class = Action)
        if $action$ then
            call($action$, $objectives(i)$.(value))
            $S(e)$.add($action(e)$)
        end
    end
    $entities \leftarrow O(e)$
    for all entities do
        if $entities(i)$ in file = 1 then
            sendSituation(entities(i),S)
        end
    end
```

Currently, requests are stored in a queue to be processed on a first-come, first-served basis. As for complexity, it should be mentioned that the system is easily scalable. Introducing more controllers would allow the management of larger environments and more entities. Also, more powerful controllers would provide shorter processing time and therefore faster objective resolution. The system could be extrapolated to larger environments such as shopping malls, schools or museums. As future work we are defining new mechanisms to allow synchronization between different controllers.

Normally in each environment there will be a controller. However, in larger environments, there may be more than one controller. Although there may be synchronization between controllers at the level of knowing which entities are connected to each one, what situations are occurring, or what objectives are being met, each controller must manage the entities that are close to it. This is because if, for example, an objective of increasing the lighting is detected, ideally, it is more efficient to select the closest controller to manage this request and select an entity that is also nearby to modify the lighting.

SMOTE provides support for the integration of different types of entities and for the identification of situations, all in a transparent way for users. The following section presents the integration of the different devices and components to identify situations and trigger strategies to achieve the goals.

4.3 Dataflow

The flow of the system is presented below, first, from the users' point of view and, second, from a more technical perspective. On the one hand, users only have to use the mobile application where their description is stored. This application runs in the background and automatically manages the description when the controller sends the situation's updates. The system works as follows:

1. The user has the *application installed*. This application will automatically generate his/her description with the goals and depending on the situations he/she goes through.
2. When a user with the application installed *enters into a situation-aware place*, he/she does not have to perform any action. The placed controller will get the description from his/her phone automatically and will orchestrate the devices according to the goals and skills detected in the environment.
3. The user will simply see how the *actions are executed automatically* based on the stored description.
4. *Situations and goals are detected automatically*. Nevertheless, if at any time the user wants to change them to obtain more accurate customization, he/she can edit this information through the section dedicated to this purpose in the mobile application.

As observed, the process of interacting with a situation-aware place is completely transparent to the users and without the need for user intervention.

On the other hand, from a technical point of view, the data flow in the controller is as follows: (1) an entity is detected; (2) its description is requested, (3) then it is processed; (4) situations are identified or created; (5) the objectives associated with the situation are attempted to be achieved. Figure 4 details this process step by step.

1. Detect entity and description request: by using a script written in Nodejs, HASS identifies when an entity enters the network. HASS has a feature called "device tracker" [6] that allows detecting when a device enters or leaves the network range. This functionality can be configured in different ways: through Bluetooth/BLE, so that it detects close entities ($\pm$ 10 ms), WiFi, which detects entities within the same network or a router that reports which devices are connected to it; or through the NMAP protocol, which allows scanning the network for devices. In our case, we will use the Bluetooth/BLE protocol to scan for devices, but this may vary depending on the application or the needs of the environment. When an entity is

[6] https://www.home-assistant.io/integrations/device_tracker.

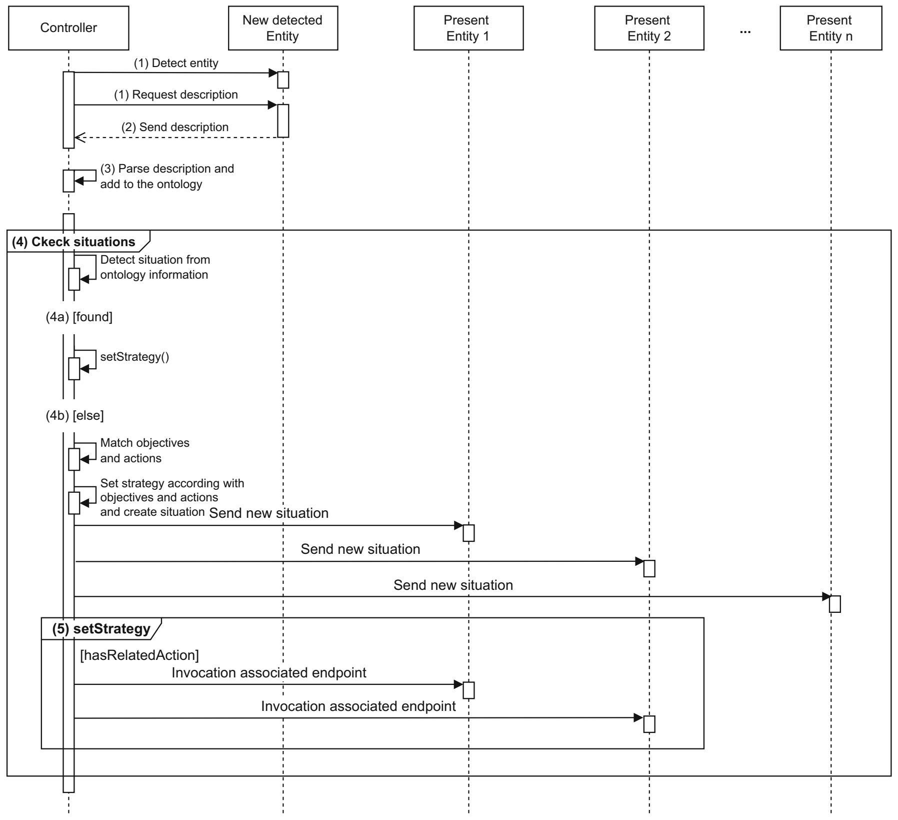


Fig. 4 Sequence of the whole process

discovered, it is subscribed to the controller topic by using the MQTT protocol and its description is requested.

2. Send description: by using the MQTT protocol the entity publishes its description. In this way, the description is sent to the controller.
3. Parse description: the information of the entity is stored in an ontology based on the IoT-O ontology [25]. This ontology contains the necessary classes that correspond to the description of the W3C-TD. All the treatment is performed by using Python, Owlready2[7] and RDFLib[8] libraries, that provide us with methods and SPARQL queries to match skill and goals.
4. Check Situations: the *Situation Manager* detects the stored situations of the entity and interprets them to detect if any of the situations is taking place. At this point, two possibilities can arise: a known situation is detected (4a), or the situation is not detected and, therefore, is new (4b).

7 https://pypi.org/project/Owlready2.

8 https://github.com/RDFLib/rdflib.

- (4a) Known situation: the strategies triggered in the past are identified and triggered again. The entities associated with the strategy are identified and their skills are invoked through the endpoints. These invocations include the associated value to set the action (e.g., luminosity-7).
- (4b) New situation: the goals and skills of all the entities involved in a new situation are detailed in their description. The goals are identified using the *Goal Generator* component, or manually set by the user. When an unknown situation is identified, all this information is processed and the goals of the devices are covered with the available skills. The methods used in the *Matcher* component to make ontology queries are: in Owlready2, the *.search(...)* method of the ontology is used to get the *iri* (Internationalized Resource Identifiers) of the individuals, and in RDFLib the SPARQL method *.query(...)* is used to obtain information from the ontology at run-time. These two libraries are used in a complementary way to set a link. In the case of conflicting goals, such as different illumination levels for two people, the goal that is reached last will be considered. Finally, the situation is sent to all the entities in the environment to be added to their description.

5. Set strategy: finally, once the strategy is identified, the endpoints of the entities are invoked. This is done by accessing the skill of a related entity that can do a change in the environment to achieve the goals.

The following section validates the proposed architecture by detailing a case study based on a smart office.

5 Validation

In this section, the feasibility and the efficiency of the proposed architecture are evaluated by using SMOTE in a case study. For feasibility, we analyze if the presented ideas can be implemented in a real environment. For efficiency, we evaluate if the obtained response times are adequate for IoT environments, where a certain processing speed is required. These two aspects are first evaluated in a real environment to get real values, and next, in simulated ones to analyze the behaviour of the system with a larger number of entities.

5.1 Smart office case study

A case study based on a smart office has been developed where there are several entities, from IoT devices that regulate the brightness and turn on or off electrical appliances, to people whose preferences must be resolved by these devices. The smart office is composed by the following entities:

- **1 Raspberry Pi 4 Model B** (ARM Cortex-A72 Quad-core and 4GB RAM). A microcomputer to take the role of the controller, managing the different entities, and coordinating them to achieve the goals of a situation.

- **1 Yeelight v2 bulb**.[9] An IoT device with the skills of changing the color and the luminosity of a room.
- **1 Shelly v1**[10] A switch with the skill of turning on or of the power supply of an appliance such as an A/C, a TV, a computer, etc.
- **3 Android Smartphones**: Belonging to three people and exposing their goals with the environment under different situations.

– Honor 9: Kirin 960 Octa-core, 4GB RAM (*Daniel*).
– Moto G6: Snapdragon 450 Octa-core, 4GB RAM (*Paul*).
– Xiaomi Redmi 7: Snapdragon 632 Octa-core, 3GB RAM (*Claudia*).

This case study is focused on evaluating the behavior of the different entities and, specially, the controller. Depending on when the people enter or leave from the office, the controller analyzes the goals of the people present to adapt the behavior of each smart device to the detected needs. The established value depends on the current order of arrival for the entities, where the last entity to arrive will have priority over the others. For this purpose, the processing time of the situations and the communication and execution time of the skills are evaluated. These tests were repeated 25 times under the same conditions, in order to obtain average values. The tests conducted were:

1. **To identify new situations (#1).**
2. **To process known situations (#2).**
3. **To measure the execution time for description parsing (#3).**
4. **To test the scalability of the system (#4).**

For tests **#1** and **#2**, Figs. 5 and 6 show the times obtained for the management of the description for a new situation and for a known situation, respectively. The results obtained show that the processing time is considerably less when dealing with a known situation. Specifically, the average execution time obtained for new situations is 2.49 s, while for familiar situations it is 0.27 s. This is because in a known situation the associated strategy is triggered and it is not necessary to invest time on matching skills with goals to define the strategies. This means that as new situations are registered, their identification accelerates the adaptation of the devices to the environment.

Regarding test **#3**, the response time to process the entities' description has also been calculated. These tests consist of requesting and sending descriptions (MQTT) and to triggering a strategy. The results obtained are shown in Fig. 7. The times obtained vary considerably due to the functioning of the MQTT protocol. However, the average response time is 146.04 milliseconds for the three mobile devices, which can be considered to be acceptable considering the size of the description, the network, the devices and the experiments performed in [26].

The results obtained validate the feasibility of the system in a real environment. The response times for situation processing and the management of descriptions through MQTT are specific to the controller installed on a Raspberry Pi. These times are promising for this type of low performance device and could be improved by using a more powerful dedicated server as controller.

[9] https://us.yeelight.com/shop/yeelight-smart-led-light-bulb-1s-color.
[10] https://shelly-api-docs.shelly.cloud/gen1.

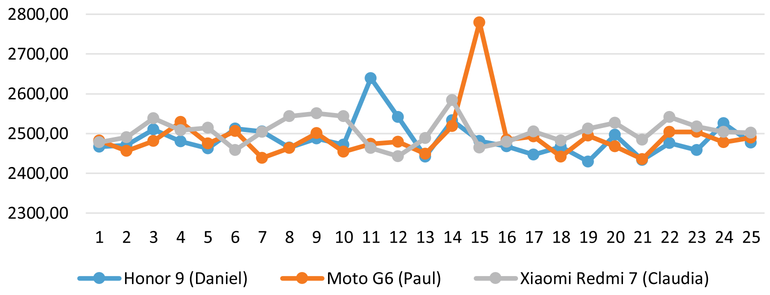


Fig. 5 Description parsing for a new situations (milliseconds)

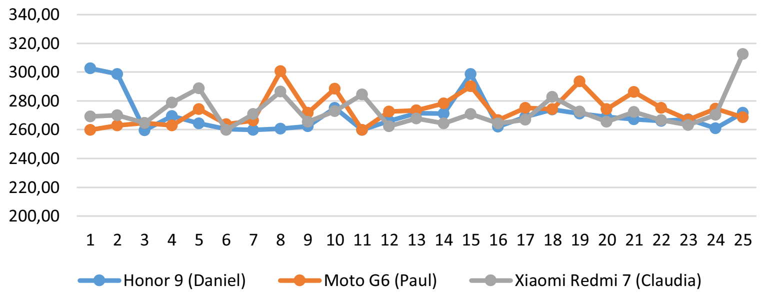


Fig. 6 Description parsing for known situations (milliseconds)

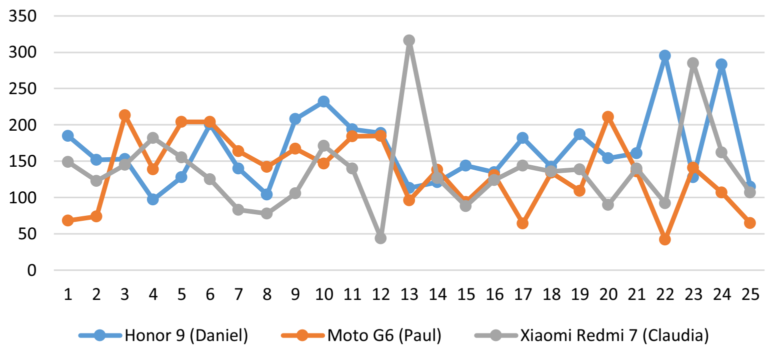


Fig. 7 Time execution for description parsing (milliseconds)

Also, test **#4** evaluates the scalability of the system, both at the network and information processing level, and it will depend on the type of controllers to be used. In our case, we have used Raspberry Pi with up to 100 entities involved in the case study, and we calculated the processing times in the controller. Figure 8 shows the obtained times:

It can be observed that the average processing time per entity (description processing) does not vary considerably with the environment (between 2.29 and 7.96 s). Hence, we conclude that even in very large environments this should not be a problem. Furthermore, the processing time for the complete environment, i.e., processing

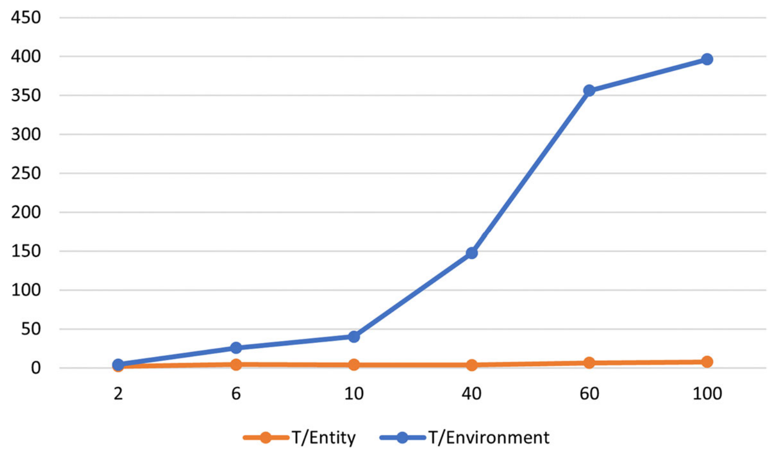


Fig. 8 Average execution time for entities and environments (seconds)

all descriptions, identifying the situations, and solving the objectives, does vary as a function of the size of the environment ranges over 4.58–396.31 s. Here it can be seen that in more crowded environments the power of the controller should be higher.

The source code of the implementation of this case study (the controller and the mobile application) is available in public repositories,[11] [12] as well as a complete video to explain the whole process of the case study.[13]

5.2 Quality of experience

To evaluate the feasibility and the behaviour of the system in larger environments, seven environments were simulated where the number of devices and people vary from 1 to 50. To maintain consistency with the previous case study, the tests performed were repeated 25 times for each of the environments. These tests are focused on the following objectives:

1. **To resolve goals with the available entities (#5).**
2. **To check the quality of the goals resolution (#6).**

For tests **#5** and **#6**, the description of the entities contain 2 skills, 4 goals and 1 situation. This configuration was selected because each IoT device has 2 skills that allow making changes in the environment, such as changing its state from on to off or changing the brightness; and people have 4 general goals, such as modifying the brightness of a room, setting a specific temperature, turning on a certain TV channel, or modifying the volume of music. Therefore, a situation about working at the office can be discovered. To do this, up to 50 different descriptions corresponding to people and devices were generated. These descriptions contain skills and goals in a random

[11] https://bitbucket.org/spilab/server-node-python-w3ctde.

[12] https://bitbucket.org/spilab/android-w3ctde.

[13] https://www.dropbox.com/s/bemodpkdc5v69rb/Video_W3CTDE.mp4.

Table 2 Solved goals and false positives after 25 executions per environment

Environment	Devices simulated	People simulated	Solvable goals	Solved goals (%)	False positives (%)
1	1	1	1.12	98.67	17.33
2	1	5	5.24	92.46	16.80
3	5	5	10.8	93.91	33.06
4	5	1	1.96	98.00	33.33
5	10	50	108.4	93.83	33.49
6	20	20	42.48	94.01	34.67
7	50	50	106.92	94.25	34.34

way, simulating a real environment of arrival and leaving of people. The algorithm were modified to detect which of the predictions are accurate and which are not. This refers to finding an appropriate skill for a particular goal (*solvable goal*). For example, to increase the brightness of a room, the ability to increase the wattage of a light bulb in that room has to be identified. There may be cases where the skills and goals do not finally match, but due to the characteristics of the device or the similarity in skills and goals, the Matcher component identifies them as a match(i.e., it is a *false positive*). For example, for the goal of increasing the brightness, the ability to increase the energy savings of the bulb may be detected. False positives indicate that the goals have been identified as achievable but once the strategy was established they could not be achieved due to problems with the invocation of the associated services or because they were not fully supported. In the tests we considered different aspects to evaluate a environment: number of devices and people, solvable goals (goals solvable with the available skills), solved goals (goals satisfactorily solved), and false positives (goals identified as solvable but not finally not solved).

The results of these tests are detailed in the Table 2 for each of the seven simulated environments. The descriptions vary to cover different possibilities that may occur in real environments. Knowing this, the system processes each environment to match actions and objectives and adapt the behavior of the devices. Two of the simulated environments are explained below.

In the environment 1, 1.12 solvable goals are detected on average. As mentioned above, there may be other goals that can not be resolved because no associated skills are available. This applies to all simulated environments. The system was able to resolve 98.67% of these goals. In addition, 17.33% of the solved goals have been identified as false positives because they have been attempted to be solved with skills or devices that have not been able to perform it. For the environment 5, 108.4 solvable goals are detected on average. In this case, the system has been able to resolve 93.83% of them, and 33.49% of the solvable goals have been detected as false positives.

Given the results it can be deduced that the system promises great performance in crowded environments. The goal resolution has been satisfactory in 95.01% of the cases. In addition, the number of false positives is 29.00% for the simulated environments. The greater the number of devices and people in the environment, the more false

Table 3 Please process the table in center and with the caption in the top (refer other tables)

	SMOTE (Ours)	MAS ontology [27]
Min	2.29	3.76
Max	7.96	4.47
Average	2.29	4.09

positives will be obtained. This is because more objectives are detected to be achieved and, therefore, the probability of failing to achieve some of them also increases.

The evaluation performed above addresses the feasibility, the efficiency and the performance of the architecture. We evaluated the latency and response times in tests **#1**, **#2**, **#3** and **#4**, and the performance and the quality experience of the "Situation Manager" and "Goal Generator" components in tests **#5** and **#6**, since these two components are the most important in the architecture and have been designed from scratch.

5.3 Comparative with similar system

Our proposal addresses different levels of IoT interoperability to identify situations in IoT environments. Currently, there are other systems that pursue similar objectives. Geng and Gao [27] design an architecture called "MAS (Multi-Agent System) Ontology" for smart homes and use the Jena [14] reasoner for decision making and task planning. This work is quite similar to ours. Both solutions detect situations using the Semantic Web based on certain actions detected in the IoT environment, in our case by detecting situations, in the case of MAS by detecting events. The two works use different standards to perform device descriptions. MAS uses an RDF-based ontology whereas our system uses the W3C-Thing description. In addition, both works use contextual information to perform actions associated to the events, in our case to adapt the environment, and in the case of MAS to avoid alarm situations. The main difference found is that our work considers different application domains, while MAS is primarily intended for a Smart Home based environment.

Table 3 shows the comparative processing time. These times refer to the time that elapses from when the event is detected, which in our case would be when the entity is detected, until the information is analyzed and an action is triggered, which in our case is the discovery of situations:

The characteristics of the machines where the tests were run differed considerably, the hardware used in [27] being more powerful. However, it can be seen how the difference in processing time is 20% on average, which indicates a promising result for our system.

[14] https://jena.apache.org/documentation/inference.

6 Related work

Adapting the IoT to changing environments remains a real concern. Iqbal et al. [28] present an interoperable platform for IoT using WoO (Web-of-Objects) and the cloud. The proposed architecture provides interoperability between environment devices and communication protocols. However, a manual intervention by users is required to achieve the correct functioning of the system. In [29], the authors present a Context-Aware System called CONASYS which is able to sense the environment and to provide contextualized services. This system considers the people's needs to adapt the devices to the environment and it can be used in any IoT domain. However, the system requires a middleware to discover devices and the rules to match them are static.

The authors of this paper address the interoperability between devices on the Internet of Medical Things (IoMT) domain. In [30], the authors present a model that captures personal information from heterogeneous sources for promoting proactive interactions between smart devices and people. This work demonstrates how interactions between people and devices can be used to promote the proactivity of software applications. Jaleel et al. detect in [31] a lack of interoperability especially due to the formats of medical data. They accordingly propose an interoperability framework called MeDIC which, through the collaboration of different types of healthcare devices, can translate these formats to be easily readable by any healthcare system. MeDIC is evaluated in different use cases such as a smart city and a smart hospital, and the authors also perform load tests to check the performance of the system. Likewise, in [32] a solution is developed to promote the proactivity of smart devices in Smart-Nursing Homes to make people's lives easier. These solutions represent considerable progress in improving interoperability between devices. However, the domain of action is limited to those where the solution is applied and the definition of people's data does not follow a standard.

Also, device interoperability implies the problem of identifying reliable "things". In this sense, Javaid et al. [33] propose a reliable and context-aware network system of things called CATSWoTS for which they define an architecture oriented to offer and consume services. The objectives are twofold. First, to make the system context-aware. And second, to achieve a level of trust to allow their interoperability. This system is based on rules that allow the identification of different criteria based on the context and characteristics of the devices. The authors consider both devices and people as service providers and consumers, which is a point in common with our work. However, no information is provided as to how the information of these things is specified, and it does not handle specific situations according to contextual properties.

Another work is developed by Andrade et al. [34] where an infrastructure (LoCCAM-IoT) is proposed to support the development of self-adaptive IoT systems. With this, the authors achieve detecting devices and users in the environment as also obtaining a list of services or devices that are of interest to satisfy the user's preferences. However, it is the responsibility of the developer to describe the rules needed to adapt to the environment. In our solution, no rules as such are necessary, as entities' preferences are in charge of sending the desired conditions to the environment.

To better understand our contribution, some of the properties evaluated during the comparative are summarized in Table 4. These properties address the objectives that

Table 4 Comparison with other solutions

Metric	CATSWoTS [33]	LoCCAM-IoT [34]	SMOTE (This work)
Heterogeneous environments	Yes	Yes	Yes
Real-time prediction	Proposed as future work	Yes, but limited to predefined rules	Yes, thanks to the controller of the environment
Consider people/devices	Yes. But the preferences are not considered	Yes, but it is centered in devices	Yes, both can collaborate
Match entities	Yes. Rule-based system to enable the interoperability	Not specified	Yes, they possess a description with detailed information
Manage situation	No	Yes, by limited to rules	Yes, they are detected, processed and stored in the entities

our architecture pursues: supporting heterogeneous environments, real-time execution of information, considering devices and people in context adaptation, linking entities to achieve a collaborative environment, and managing situations to favor proactive adaptation.

The cited works make an important contribution to the interoperability between devices in IoT environments. Nevertheless, to the best of our knowledge, there is not yet a solution that achieves adaptation to the environment in an automatic way by considering those conditions that are decisive for its configuration, such as the preferences of people, place, or nearby devices.

7 Conclusions and future work

In this paper, we have addressed the interoperability problems to favor collaboration of smart devices in the IoT from the different levels detected: technological, IoT domains, semantic and situational. In addition, the importance of mitigating this problem by other researchers is highlighted, leading us to the conclusion that interoperability between devices is a real problem.

For this reason, a conceptual architecture has been proposed with the aim of satisfying the different IoT levels, specially, the adaptation to the environment status. The implementation of SMOTE as proof of concept of the architecture allowed us to understand the importance of each level individually and how only by solving all of them in a unified way, a complete interoperability based on situations among devices will be achieved. In addition, SMOTE allows us to understand the limitations and possibilities of interoperability in a real scenario, and how the failure to overcome this interoperability conditions collaboration among devices and what is most important, the users experience. This work is a further step towards achieving a higher level of interoperability in the IoT. It delves into the different problems that condition the user experience. As future work, we plan to address conflicting goals by introducing priorities for the entities, where the adaptation of the behavior will be done according to a hierarchical system of these entities within the environment.

Acknowledgements This work was supported by the projects RTI2018-094591-B-I00 and PID2021-124054OB-C31 and the FPU17/02251 Grant (MCIU/AEI/FEDER, UE), by the 4IE+ project (0499_4IE_PLUS_4_E) funded by the Interreg V-A España-Portugal 2014–2020 program (POCTEP), by the Department of Economy, Science and Digital Agenda of the Government of Extremadura (GR21133, IB18030), and by the European Regional Development Fund.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Data availability The data used were obtained from tests conducted with real users. They are available upon request. Simulated tests were conducted with synthetic data generated with the algorithm.

References

1. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013) Internet of things (iot): a vision, architectural elements, and future directions. Futur Gener Comput Syst 29(7):1645–1660
2. Taivalsaari A, Mikkonen T (2017) A roadmap to the programmable world: software challenges in the iot era. IEEE Softw 34(1):72–80
3. Pico-Valencia P, Holgado-Terriza JA, Quiñónez-Ku X (2020) A brief survey of the main internet-based approaches: an outlook from the internet of things perspective. In: 2020 3rd International conference on information and computer technologies (ICICT), pp 536–542. IEEE
4. Roman R, Zhou J, Lopez J (2013) On the features and challenges of security and privacy in distributed internet of things. Comput Netw 57(10):2266–2279
5. Lelli F (2019) Interoperability of the time of industry 4.0 and the internet of things. Fut Internet 11(2):36
6. Parameswaran M, Whinston AB (2007) Social computing: an overview. Commun Assoc Inf Syst 19(1):37
7. Endsley MR (1995) Toward a theory of situation awareness in dynamic systems. Hum Factors 37(1):32–64
8. Berrocal J, Garcia-Alonso J, Canal C, Murillo JM (2016) Situational-context: a unified view of everything involved at a particular situation. In: International conference on web engineering. pp 476–483
9. Patel KK, Patel SM et al (2016) Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges. Int J Eng Sci comput 6(5)
10. Noura M, Atiquzzaman M, Gaedke M (2019) Interoperability in internet of things: taxonomies and open challenges. Mobile Netw Appl 24(3):796–809
11. Miori V, Russo D, Ferrucci L (2019) Interoperability of home automation systems as a critical challenge for iot. In: 2019 4th International conference on computing, communications and security, ICCCS
12. Elkhodr M, Shahrestani S, Cheung H (2016) The internet of things: new interoperability, management and security challenges. arXiv preprint arXiv:1604.04824

13. Flores-Martin D, Mäkitalo N, Berrocal J, García-Alonso J, Mikkonen T, Murillo JM (2021) Layered interoperability for collaborative iot applications. In: Future of Information and Communication Conference, pp 192–211. Springer
14. Morabito R, Petrolo R, Loscri V, Mitton N (2018) Legiot: a lightweight edge gateway for the internet of things. Futur Gener Comput Syst 81:1–15
15. Yacchirema DC, Palau CE, Esteve M (2017) Enable iot interoperability in ambient assisted living: Active and healthy aging scenarios. In: 2017 14th IEEE annual consumer communications & networking conference (CCNC), pp 53–58. IEEE
16. IEEE: IEEE Internet of Things. https://iot.ieee.org/. (Accessed on 17/01/2022)
17. Kaebisch S, Kamiya T, McCool M, Charpenay V (2019) Web of Things (WoT) Thing Description. Candidate recommendation, W3C
18. Maarala AI, Su X, Riekki J (2017) Semantic reasoning for context-aware internet of things applications. IEEE Internet Things J 4(2):461–473
19. Kim J, Yun J, Choi S-C, Seed DN, Lu G, Bauer M, Al-Hezmi A, Campowsky K, Song J (2016) Standard-based iot platforms interworking: implementation, experiences, and lessons learned. IEEE Commun Mag 54(7):48–54
20. Gyrard A, Datta SK, Bonnet C, Boudaoud K (2015) Cross-domain internet of things application development: M3 framework and evaluation. In: 2015 3rd International conference on future Internet of Things and Cloud, pp 9–16
21. Flores-Martin D, Berrocal J, García-Alonso J, Murillo JM (2020) Extending w3c thing description to provide support for interactions of things in real-time. In: International conference on web engineering, pp 30–41
22. Rojo J, Flores-Martin D, Garcia-Alonso J, Murillo JM, Berrocal J (2020) Automating the interactions among iot devices using neural networks. In: 2020 IEEE International conference on pervasive computing and communications workshops (PerCom Workshops), pp 1–6. IEEE
23. Rhayem A, Mhiri MBA, Gargouri F (2020) Semantic web technologies for the internet of things: systematic literature review. Internet of Things 11:100206
24. Seeliger A, Pfaff M, Krcmar H (2019) Semantic web technologies for explainable machine learning models: a literature review. PROFILES/SEMEX@ ISWC 2465, 1–16
25. Seydoux N, Drira K, Hernandez N, Monteil T (2016) Iot-o, a core-domain iot ontology to represent connected devices networks. In: European Knowledge Acquisition Workshop, pp 561–576. Springer
26. HTTP vs. MQTT: A tale of two IoT protocols. https://cloud.google.com/blog/products/iot-devices/http-vs-mqtt-a-tale-of-two-iot-protocols
27. Geng D, Gao Q (2019) Semantic web technology and prolog reasoning based task planning mechanism and application in smart home for internet of things. In: 2019 IEEE 9th international conference on electronics information and emergency communication (ICEIEC), pp 1–4
28. Iqbal A, Ullah F, Anwar H, Kwak KS, Imran M, Jamal W, ur Rahman A (2018) Interoperable internet-of-things platform for smart home system using web-of-objects and cloud. Sustain Cities Soc 38:636–646
29. de Matos E, Amaral LA, Tiburski RT, Schenfeld MC, de Azevedo DF, Hessel F (2017) A sensing-as-a-service context-aware system for internet of things environments. In: 2017 14th IEEE annual consumer communications & networking conference (CCNC), pp 724–727
30. Mäkitalo N, Flores-Martin D, Berrocal J, Garcia-Alonso J, Ihantola P, Ometov A, Murillo JM, Mikkonen T (2020) The internet of bodies needs a human data model. IEEE Internet Comput 24(5):28–37
31. Jaleel A, Mahmood T, Hassan MA, Bano G, Khurshid SK (2020) Towards medical data interoperability through collaboration of healthcare devices. IEEE Access 8:132302–132319
32. Flores-Martin D, Rojo J, Moguel E, Berrocal J, Murillo JM (2021) Smart nursing homes: self-management architecture based on iot and machine learning for rural areas. Wireless Commun Mobile Comput. https://doi.org/10.1155/2021/8874988
33. Javaid S, Afzal H, Arif F, Iltaf N, Abbas H, Iqbal W (2019) Catswots: context aware trustworthy social web of things system. Sensors 19(14):3076
34. Andrade RM, Junior BRA, Oliveira PAM, Maia ME, Viana W, Nogueira TP (2021) Multifaceted infrastructure for self-adaptive iot systems. Inf Softw Technol 132:106505