




An ontology-based approach to integrate TV and IoT middlewares

Danne Makleyston G. Pereira¹  · Francisco José da S. e Silva² ·
Carlos de Salles S. Neto² · Davi Viana dos Santos² · Luciano Reis Coutinho² ·
Álan L. V. Guedes³

Received: 5 June 2019 / Revised: 3 August 2020 / Accepted: 18 August 2020 /

Published online: 10 September 2020

© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

Internet of Things (IoT) is the interconnection of physical devices, known of smart objects, that share data and collaborate among them to support users. There are IoT usage scenarios in several domains. In-home domain, the use of smart objects may support monitoring and automation of the environment. In a home, the TV is one of the most used devices. It may also collaborate with the smart objects to enhance the interactive experience of viewers on watching. To support such a meeting, we propose an (i) conceptual model, IoTTV-Ont, that allows modeling entire homes, with smart objects, people' profile, and viewers' interaction events with TV applications; (ii) a software architecture that integrates and interoperates digital TV and IoT middleware. This architecture allows the TV applications: to be aware of the physical environment context, to change aspects of the physical environment, to identify the viewers, and to receive multimodal interactions performed by viewers. We demonstrate our work through use cases. For this, we implement a prototype of our software architecture

✉ Danne Makleyston G. Pereira
danne.pereira@lsdi.ufma.br

Francisco José da S. e Silva
fssilva@lsdi.ufma.br

Carlos de Salles S. Neto
csallesneto@gmail.com

Davi Viana dos Santos
davi.viana@lsdi.ufma.br

Luciano Reis Coutinho
luciano.rc@lsdi.ufma.br

Álan L. V. Guedes
alan@telemedia.puc-rio.br

¹ Federal Institute of Tocantins, Colinas of Tocantins, TO, Brazil

² Universidade Federal do Maranhão, São Luís, MA, Brazil

³ Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil

and modeled home using IoTTV-Ont. Then we developed usage scenarios that enhance TV viewers' experience by supporting features such as content adaptation, multi-sensorial immersive experience, and multimodal user interaction.

Keywords Internet of things · Digital TV · Mulsemedia · Multimedia immersion · Multimodal interaction

1 Introduction

Internet of Things (IoT) is a combination of ubiquitous computing and the Internet. It is composed of the interconnection of physical devices, named smart objects, equipped with electronics, software, sensors, actuators, and network connectivity. Those objects get and exchange data among them, cooperating with people and the environment in which they are localized [3, 16, 25, 28]. Particularly in the household domain, those smart objects allow IoT scenarios such as monitoring (e.g., temperature sensors and kinetics) and environment control (e.g., dimmers, thermostats, and speakers).

One mainly used device in the household domain is the TV. It usually works as a media center allowing tasks (e.g., media sharing and web browsing), and gaming [7]. However, the TV can also work as a smart object collaborating with other devices to improve its interactive experience. In this way, the TV can interfere in the house environment by triggering smart objects. Managing those objects enables immersive experiences to viewers, also known as multi-sensorial media or MulseMedia [34]; such experiences aiming to provide sensorial effects to its viewers using actuators resources (e.g., lamps, fans, and smell dispenser) present in the physical environment; to adapt audiovisual content to the house environment context through smart objects sensors. For example, a TV can adapt its content according to the sensors data gathered by viewers' smartphones, allowing the movements detection of viewers through accelerometer sensor.

In this context, this work explores the development of applications that require the cooperation of TV applications and smart objects in a house. For this purpose, it requires a software infrastructure supporting a non-trivial degree of collaboration. Some challenges must be solved to converge these two areas, such as dynamic discovery of smart objects and the services they provide since they can enter and exit a physical environment offering several different services; the support to the communication protocols heterogeneity of smart objects since they use protocols both IP-based and short-range technology, such as Bluetooth and ZigBee; the physical environment representation since it is necessary to model the rooms of the residence and the objects inside them to allow the management of smart objects in a specific way (e.g., limiting the change of the physical aspects to only some rooms); the identification of viewers and their profiles since the TV application can adjust to the profile of viewers present on the physical environment; the synchronization of the playing media with the smart objects present in the physical environment, allowing the user to increase immersion to the displayed content; and programming abstractions for the development of TV applications since the exchange of messages about the smart objects in the TV application must be structured according to the representation of the conceptual model that defines the residential domain.

To solve the issues mentioned, it is necessary to export to the application running on TV the knowledge regarding sensors and actuators services present in the physical environment. It is possible to use the sensors available in mobile devices (e.g., smartphones and tablets) to increase the perception of the environment and its users. This way, using this knowledge

of sensors, actuators, and viewers through their respective mobile devices, it is possible to promote the increase in the level of immersion of the user regarding the content presented on TV. Adapting the TV content to the context data of the physical environment also enables an increase in the level of immersion. It also enables changing aspects of the physical environment according to the narrative of the content presented as well as enabling new ways for viewers to interact with TV applications.

This work proposes an approach to integrate the TV with smart objects based on the middleware perspective to solve the challenges mentioned. Therefore, we propose an (i) abstract software architecture that aims to integrate and interoperate digital TV and IoT middleware, and (ii) an ontology that defines the concepts and their relationships in a residential environment.

The abstract software architecture allows the use of several digital TV and IoT middleware, such as Ginga¹ and HBBTV² for digital TV middleware and M-Hub [35], MOSDEN [27], and AndroSIXTH [13] for IoT middleware. Also, our proposal uses a shared vocabulary for exchanging messages between TV applications and IoT middleware. We do this through an ontology, named IoTTV-Ont, that defines this vocabulary for house context (e.g., rooms and kitchen). This ontology aids the exchange of data between the digital TV applications and IoT middleware since this data exchanged must share the same semantic values.

To present our work, we have organized this paper as follows: Section 2 discusses related work; Section 3 presents the proposed ontology IoTTV-Ont to integrate IoT and TV contexts. Section 4 presents a software architecture that allows the interoperation of IoT and digital TV middleware. Section 5 presents a prototype that uses an implementation of the architecture proposed. Section 6 shows the demonstration of software architecture through use cases development over the prototype. Section 7 presents final remarks and future work.

2 Related work

Other works also share our motivation for enabling integration between TV applications and smart objects. To better discuss those works, we first highlight their main features. In this section, we discuss the presence of these features in the related works. This feature analysis is summarized in Table 1, in which it also presents the user experiences that these features can offer.

The features: (F1) supports TV to be able to change aspects of the physical environment (e.g., change of lighting and temperature) using smart objects services; (F2) supports TV to adapt according to the context data of the physical environment. For example, a TV application can pause its presentation if the presence sensors do not detect the presence of viewers in the environment; (F3) supports the use of sensors data from a personal mobile device (e.g., smartphones and tablets) to allow new ways for the viewer to interact with the TV application; (F4) supports several communication protocols because of the heterogeneity of smart objects and their communication technologies (e.g., Bluetooth Classic and BLE); and (F5) supports dynamic discovery of smart objects since they can enter and exit the IoT middleware range area.

¹<http://itu.int/rec/T-REC-H.761>

²<http://hbbtv.org>

Table 1 Relationship between features and related work

Features	Works						
	Saleme et. al. [31]	Guedes et. al. [17]	Punt et. al. [29]	Rosa et. al. [30]	Lima et. al. [22]	Lucena et. al. [32]	Kapri et. al. [21]
F1. Supports TV change the physical environment. User experience: User immersion.	x	x					x
F2. Supports TV adapt to physical environment. User experience: Content adapted to viewer.		x				x	x
F3. Supports TV use of mobile devices' data. User experience: Content adapted to viewer and multimodal interactions.			x	x	x		
F4. Supports several communication protocols. User experience: Easily connect smart objects to the TV.						x	x
F5. Supports dynamic discovery of smart objects. User experience: Easily connect smart objects to the TV.	x		x		x	x	x

Saleme et al. [31] and Guedes et al.'s [17] works focus on **using smart objects to provide TV multi-sensorial experiences**. Saleme et al. proposed that TV applications use sensory effects defined in the MPEG-V standard using the Sensory Effect Metadata (SEM) [19] with time-tags related to the video. This work proposed to decouple the renderers of sensory effects from video players. Thus, TV applications running in digital TV middleware can send SEM commands to the renderer developed in the proposal (SE Renderer) through a UPnP service, which in turn triggers an Arduino³ device connected to it. In their experiments, the Arduino device was able to exhibit sensory effects that controlled a lamp, vibration, and wind actuators. Guedes et al. follow a similar path but focus on applications developed in NCL language from the Brazilian TV middleware Ginga. They use an <media> element to communicate by socket with a microcontroller (Intel Galileo⁴) connected to actuators. Therefore, commands over this media trigger actions in actuators present in the physical environment while changes in its state inform some updates in the environment (people proximity). Guedes et al.'s work requires previous knowing of the microcontroller's IP. On the other hand, Saleme's work uses UPnP to dynamically discover and connect to sensory renderers, which triggers the smart objects attached to it.

Punt et al. [29], Rosa et al. [30], and Lima et al. [22] make efforts to **improve the TV experience using mobile devices**. Punt et al. proposed an architecture that enables the running of multiplayer games on TV in which the users can be geographically distant and can interact through mobile devices. The game is deployed on both TV and a cloud server. The interaction of the users occurs through mobile devices dynamically participating by connecting to the cloud. Rosa et al.'s work [30] allows TV applications to adapt their content to each viewer. This work regards only one individual who uses these applications. In this way, it is possible to identify the user and stores, in the mobile device, the contextual data of the content presented (e.g., date, time, geolocation, and channel), and interaction history. Lima et al.'s work [22] allows viewers to interact with TV applications through mobile applications and use the context data of the content presented in their online social networks. In this work, the users' profiles are unknown by the TV application. Also, this work does not use a conceptual model to represent users' profiles, leaving the mobile device application to define this representation model.

Finally, Lucena et al. [32] and Kapri et al.'s [21] works focus on **using TV as an interface for the use of smart objects in homes**. Lucena et al. proposed an architecture to allow users to view on TV the data extracted from medical sensors attached to themselves (e.g., blood pressure meter and pulse oximeter) or arranged in the physical environment (e.g., temperature sensors). They implemented it using an OSGi (Open Services Gateway Initiative) gateway, but emphasized that they do not have a standard for reading the medical sensor data due to it being produced by different manufacturers. Kapri et al. proposed a TV device that joins in an IoT network. More precisely, the TV can take actions (e.g., change sound volume) through messages received from smart objects and can send messages to an IoT controller to manage smart objects. The TV should have an IoT device process center implemented in a separate hub or integrated with the TV. None of these works have semantic resources for the discovery of smart objects and their services.

Each of these features allows for an increase in the user experience of TV. F1 allows an increase in the Quality of Experience (QoE) by using sensory effects in the physical environment [17, 31]. F2 and F3 allow the personalization of content by gathering the users'

³<http://arduino.cc>

⁴<http://intel.com/content/www/us/en/ark/products/83137/intel-galileo-gen-2-board>

information from the environment (e.g., mobile devices). That is essential, especially when we consider that TV is a shared-use device and needs to be used by different people. The amount of data gathered can be vast and complex in an environment with multiple sensors and people. Therefore, it is necessary to use a component capable of processing these complex events to assist in personalizing the environment and the content. None of the related works addressed this point. Finally, F4 and F5 allow to easily configure smart objects to be connected with the TV. This is essential due to the increase of IoT devices number and types.

The related works only partially meet these functionalities. We propose an approach based on middleware integration to handle these features and then enable the cited user experience increments. This approach is divided into two parts: a TV and IoT integration ontology (discussed in Section 3) and a low couple software architecture (discussed in Section 4).

The ontology enables developers to use common vocabulary data for modeling residences, smart objects, and viewers' profiles. In other words, these modeled entities enable that the exchange of data between the digital TV and IoT middleware using the same semantic values. In this way, this semantic allows the TV application independence regarding smart objects since value exchanging is the ontology-based, not the non-semantic values created by their manufacturers. This ontology also enables the modeling of user interactions, e.g., it is possible to model movement events with the hand or events generated by voice command.

Our middleware-based approach allows us to focus on the data exchange between them and how the data will be analyzed and delivered. In this way, the ontology assumes the role of enabling interoperability. At the same time, a component capable of analyzing data is used to track and identify events in the exchanged data (CEP engine), e.g., the user interaction identification with the TV application.

On the one hand, the software architecture allows the TV application the perception of the physical environment and the people inside it. For this purpose, our software architecture exports for TV applications the context data of the environment. That is possible because our proposal uses an IoT middleware to discover, connect, and read smart object data. It also offers an application programming interface (API) to the TV applications that consume these data. At last, a TV application using our proposal can trigger smart objects present in the physical environment; consequently, TV applications can change aspects of the environment according to the content presented on TV.

3 Proposed integration ontology

Several IoT middleware gathers data from smart objects and allows software to use this data through APIs. However, this data usually has non-semantic information making it difficult to use smart objects data by several software. Thus, we note the need to define a conceptual model that allows modeling a home environment with the smart objects present inside it. In this way, the conceptual terms of the model can be used as a vocabulary for the IoT middleware to write the identification of discovered services of smart objects semantically. So the IoT middleware can work in ways interoperable with other software applications or other middleware due to the exchange of semantic data.

According to Guarino [15], "an ontology is a logic that explains the meaning intended of a formal vocabulary, that is, its ontological commitment to a particular conceptualization

of the world.” Thus, an ontology formally specifies the concepts and relations of a domain [14]. Therefore, a domain ontology can both define a shared vocabulary that can be used for the interoperation between software and IoT middleware and a semantic notation of a house and the services of the smart objects present inside it and viewers’ profiles.

On the other hand, smart objects, when discovered by IoT middleware, give their identification data, which are usually strings assigned to each service according to their specifications of the manufacturer. Thus, several smart objects that have similar services may have distinct identifications as well as the identification of those services may also be different. For example, an environment can have various smart objects that offer the same services, but distinct manufacturers can develop them. A manufacturer can attribute the label “temp” to a temperature sensor while another manufacturer can attribute the “temperature” label for this service. Therefore, several ways to identify smart objects can result in an application dependent on a single manufacturer. However, the use of semantics values to describe the services of smart objects solves that problem. That is, continuing with the previous example, smart objects could be referenced by a term that expresses the semantics of a temperature sensor (e.g., TemperatureSensor) rather than “temp” and “temperature.”

We developed an ontology named IoTTTV-Ont, which allows modeling both houses and the people/smart objects present inside them. Thus, software architecture can use the IoTTTV-Ont to exchange data among components of this architecture using the same meanings, allowing interoperability between these components. Also, the IoTTTV-Ont allows the software architecture to handle data of smart objects through the semantic, thus solving issues about their heterogeneity, due not to need to handle each smart object by their specificity but by their value semantic.

For the development of this ontology, we applied the method Development 101 [26]. This ontology development method has a sequence of steps for its completeness, as follows: determining the domain and the scope of the ontology; consider the reuse of existing ontologies; to enumerate essential terms for the ontology; define the hierarchical structure of classes; set the properties; define constraints; generate the instances.

The development of IoTTTV-Ont considered the house as the domain of ontology. We developed it to enable interoperability between digital TV and IoT middleware to increase the level of immersion of viewers to content played on TV. It also allows the viewers to be identified and to be able to interact with this content.

We considered some concepts needed to cover this domain, namely, concepts that define the rooms of a house since to allow the management of smart objects precisely (e.g., enabling to identify the smart objects present in the room); concepts that specify smart objects, so that it is possible to locate smart objects by their semantics instead of their identifier defined by their manufacturer; concepts that describe people and their profiles since one of the aims of this work enables the adjustment of the presented content to the viewers present in the physical environment; and concepts that allow defining multimodal interactions of the viewers to the application running on the TV.

We identified other ontologies in the literature that conceptualize the house environment and ontologies that are modeling the people present in this environment. Thus, several ontologies have been listed, such as DogOnt [5], DomoML [33], Context-Based Digital Personality (CBDP) [9], and Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA) [8]. We used several aspects of these ontologies in IoTTTV-Ont. Moreover, we

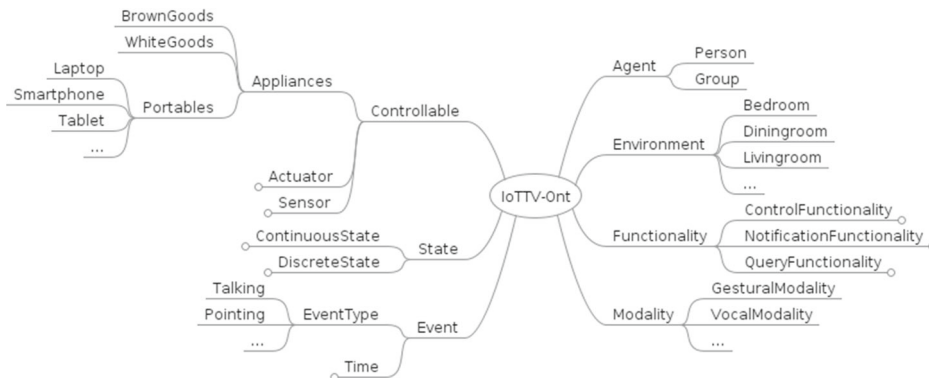


Fig. 1 Main classes of IoTTV-Ont

chose to reuse some of their concepts, their relations, and their hierarchies. The IoTTV-Ont development used the Web Ontology Language (OWL).⁵

Among these ontologies, DogOnt and DomoML have concepts that define a house from its ground to the smart objects present inside it. Both describe the functionalities of smart objects regardless of the technology used by them. The difference between these ontologies is how they represent the functions of each smart object. DogOnt uses the descriptive model that defines the functionality and later associating with smart objects. DomoML has a compositional approach. Then their functionalities are derived from the composition of other features. In the development of our ontology, we chose to use the descriptive model present in DogOnt.

To specify smart objects with their functions and their states, the terms of classes Controllable, Functionality, and State, respectively, of DogOnt were used. Figure 1 shows the IoTTV-Ont with its main classes, including those classes that define smart objects.

The State class represents the state of a smart object at a particular instant. A state can be of a continuous or discrete value, so this class has two subclasses: ContinuousState and DiscreteState. The Functionality class describes the functions that a smart object has, which can be: notification functionality (NotificationFunctionality), which is the ability of a smart object to notify its state autonomously; query functionality (QueryFunctionality), which allows query to a smart object about its state; and control functionality (ControlFunctionality), which enables the sending of commands to smart objects to work in the environment. The Controllable class has subclasses; they are Appliances, Actuator, and Sensor. In Appliances, the subclass named Portables was added, which represents mobile computing devices easily transportable (e.g., laptops, smartphones, and tablets). To describe the concepts of the physical environment (e.g., rooms), we used the term Environment class, which is equivalent to the Building Environment class, present in DogOnt and DomoML.

The CBDP and SOUPA ontologies have concepts that specify the people present in an environment. The CBDP conceptually describes the people and their preferences regarding the configurations of software using it. The SOUPA defines “people” as equivalent to the concept “Person” used in the FOAF⁶ [6] ontology. FOAF provides underlying mechanisms for specifying people’s profiles and their relationships with other people. That ontology is

⁵<http://w3.org/OWL>

⁶Dictionary of people-related terms that used in structured data (e.g., RDF, JSON-LD, Linked Data) [6].

generally used in relationship websites, allowing the exchange of data of its users. IoTTV-Ont uses the same approach used in SOUPA to reuse FOAF to describe the individuals present in the house. In this way, the term “Person”, equivalent to that defined in FOAF, was used to describe the profile of the people present in the physical environment. The Person class is a subclass of Agent, and it is different from the Group class, which in turn represents a group of people.

To identify which person is present in a particular physical environment, we supposed that each portable device belongs to a viewer. Consequently, when a mobile device is detected, it is considered that the owner of the device is present in the environment. For that purpose, we defined the relation of the type “hasPortables” of class Person to class Portables. That relationship defines an owner (an instance of the Person class) for each portable device (an instance of the Portables class). Also, the class Portables has a property, named Id, of the literal type that distinguishes its instances.

We define, in the scope of this paper, that the interactions of the viewers with the TV application should use the internal sensors of the mobile devices (e.g., accelerometer and gyroscope sensor). These interactions, in IoTTV-Ont, are described using concepts established in Wehbi et al.’s work [36]. In this case, the terms Event and Modality and their respective class hierarchies have been reused, taking advantage of their concepts and relationships. The Modality class represents the way the user interacts with the application, such as gestural (GesturalModality) or voice (VocalModality), for example. The Event class describes the event generated by interaction of the user with the TV application. The Event class has subclasses that define the time (Time) of the beginning and the end of the events, and the type of the events generated (EventType) that can be Pointing, Talking, and others. There is a relation named “hasEvent” that associates Modality and Event. For example, if the viewer makes a hand gesture, an event of type Pointing is associated with the event of type GesturalModality through relating “hasEvent.”

Therefore, using the IoTTV-Ont ontology is possible to interoperate the digital TV and IoT middleware since this ontology works as a shared vocabulary among the applications that use it. IoTTV-Ont enables an application to identify the viewer’s profile since it has concepts to model people. Also, this ontology enables IoT middleware to write the semantic way the data of smart objects present in the environment, offering support for heterogeneity of these smart objects.

4 Proposed software architecture

This section describes a software architecture that allows the interoperation between digital TV and IoT middleware. The description of this architecture is regardless of middleware for IoT and digital TV. In this architecture, TV applications communicate with smart objects through mobile devices (e.g., smartphones and tablets). These devices allow the run of applications and have several communication technologies such as Bluetooth, infrared, and Wi-Fi. Also, they contain several built-in sensors such as accelerometer and gyroscope sensors, which its use can enable multimodal interactions of the viewers with the TV application.

Figure 2 shows the proposed software architecture overview. This figure has three sections, which represent: the smart objects, the mobile device with their software components, and the TV environment in which has a TV application and communication component. Communication between the TV environment and the mobile device must use the return

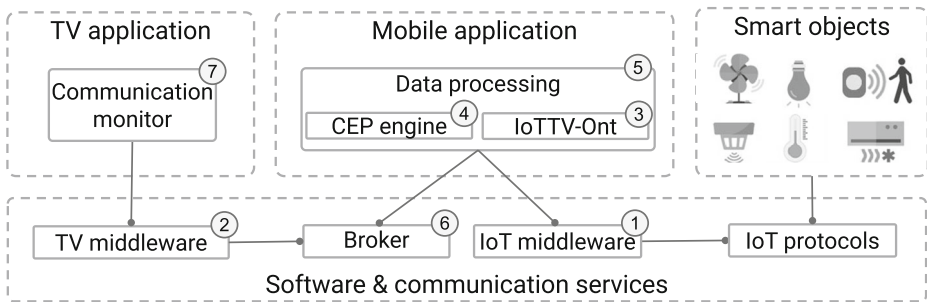


Fig. 2 Overview of the proposed software infrastructure for the exchange of data between the TV and smart objects

channel of TV (which uses the TCP/IP) and the Wi-Fi connectivity of the mobile devices. So communication can be done through the Internet or only using the local network. Communication between smart objects and the mobile device must occur through Wireless Personal Area Network (WPAN) communications, such as Bluetooth Classic or Bluetooth Low Energy (BLE).

Communication between smart objects and mobile devices occurs through **(1) IoT middleware**. This middleware must offer to support: the heterogeneity of WPAN communication protocols and technologies of smart objects; discovery dynamically smart objects and their services; exchange data between smart objects and mobile device through the short-range technologies available in the mobile device; generate notifications to the client application when there are disconnections of smart objects; gather data from the internal sensors of the mobile devices of the viewers present in the physical environment. These features of an IoT middleware are essential for the materialization of this architecture. Because these features make it possible to notify the TV application of the interactions made by the viewers, as well as enable it to export the context data of the physical environment to the TV.

(2) TV middleware must offer a TV application running environment which abstracts the functional modules of digital TV such as access to the return channel. It must provide means for TV applications to interact with the playing media regardless of their running time or screen space.

The heterogeneity of smart objects can limit technological tools to only a manufacturer because they can use different labels and properties for each object. To solve the problem, we defined that software architecture must exchange semantic data among its components. Thus, an implementation of this software architecture can use resources of the smart objects available in the physical environment using their respective semantic values defined in **(3) IoTTV-Ont** (Section 3). In this architecture, this component represents the IoTTV-Ont ontology schema and its instances.

Internal sensors in mobile devices (e.g., sound sensors, gyroscopes, accelerometers, and touch sensors) can generate multimodal interaction data of viewers to the TV application. However, these sensors produce a data stream that must be analyzed in a relatively short period in order not to impair the viewers' response time, avoiding losses in the immersion of viewers with watched content. So we chose to use a **(4) CEP Engine** (Complex Event Processing – CEP) to track and analyze these data streams.

The CEP proposes to analyze and control a complex series of interrelated events to time cost close in real-time [24]. It uses rules (CEP rules) to detect events on the data stream. CEP allows correlating event streams with patterns of interest, resulting in other complex events

derived from input events [10]. In the proposed software architecture, the event resulting from the relationship between data streams from internal sensors of mobile devices and interaction patterns predefined by the viewers should be the viewers' multimodal interaction command with the TV application.

(5) Data Processing is the component that interconnects all the other components allocated in the mobile device. More precisely, the Data Processing manages the receipt of data from IoT middleware and stores them in runtime memory, rewriting them according to the IoTTV-Ont ontology. Whenever the data stream sent from IoT middleware is data of the mobile device's internal sensor, the Data Processing must use the CEP engine to analyze this stream looking for patterns of interaction of the viewers to the TV application. Also, Data Processing maintains in runtime memory the data of the discovered smart objects. So whenever the IoT middleware reads new values of the smart objects' services, the Data Processing must update the data kept in memory with these new values. This component should provide an application programming interface (API) for use in mobile applications. At last, it also must be able to export the context data from the physical environment to the TV application.

(6) Broker is the component responsible for allowing the exchange of data between the Data Processing and the TV application. It must use the publisher-subscriber paradigm because this paradigm allows the decoupling of the parties that use it, enabling the applications not to establish a direct communication relationship with each other [11]. So it receives data from one part and sends them to others interested in that data. In this paradigm, subscribers can express their interest in an event while the publishers are registering such events. In this way, all subscribers are notified whenever a publisher registers an event [2]. So it is the task of the broker managing the receipt and the sending of the data trafficked by it. The use of the broker allows several TV applications to subscribe to only one broker. That is, it enables physical environment data getting by IoT middleware to be sent for all TV applications interested. Also, mobile applications of other mobile devices present in the physical environment may be subscribers to the topics of only one broker.

The **(7) Communication Monitor** is the component that must be imported by TV applications to abstracts the communication between these applications and the broker. This module must have a data storage system to record the semantic data of the services and states of the detected smart objects in the physical environment, updating them whenever the IoT middleware reads new values. So TV applications are context-aware of the physical environment. Also, the Communication Monitor must provide abstractions for the development of TV applications according to the conceptual model IoTTV-Ont. In this way, the TV application developer does not focus on the particularities of smart objects, but rather on the semantic terms expressed in the IoTTV-Ont.

Use the IoTTV-Ont allows developers to implement TV applications using the terms expressed in the ontology itself. Developers can specify which sensors and actuators they want to use in their TV applications when those smart objects are discovered in the environment. In this way, TV applications can trigger a specific function of a smart object, since the definition of that smart object is clear in IoTTV-Ont. For example, for a TV application to trigger a smart object that manages the ventilation system of a physical environment, the developer must pay attention to the structure defined in the ontology. That is, he must use the following hierarchical sequence: Actuator \rightarrow Functionality \rightarrow ControlFunctionality \rightarrow Command in which it is possible to set "off" value to disable the ventilation system.

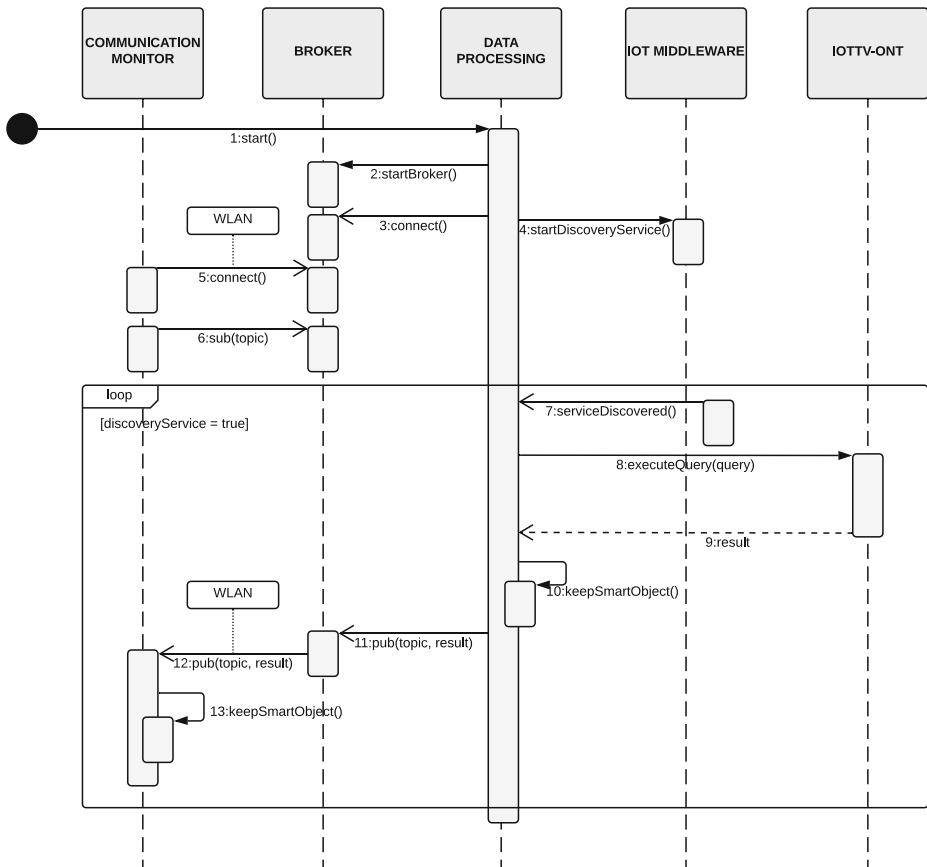


Fig. 3 Sequence Diagram: sending data from smart objects to the TV application

An implementation of this software architecture meets the aims exposed in the scope of this work, remembering: enables TV applications to change aspects of the physical environment, allows TV applications context-aware of the physical environment, and enables viewers' multimodal interaction with TV applications. The next section outlines how the components of the proposed architecture should behave to meet those aims.

4.1 Supports TV application to be physical environment context-aware

For TV applications to be context-aware of the environment, it is first needed to find out what smart objects' services are being offered. IoT middleware is responsible for service discovery since it searches for services of close smart objects through range-short technologies. Figure 3 shows a sequence diagram that illustrates the steps to be followed by the software architecture to export data of the environment to the TV applications.

After the initialization of the Data Processing (step 1), it initializes and connects to the broker (steps 2–3). Data Processing begins the process of services discovery of IoT middleware (step 4). On the TV application side, the Communication Monitor connects and subscribes to the broker, and it registers its interests over some smart objects (steps 5–6).

The discovery service of IoT middleware must remain active in the background until it is interrupted by the Data Processing. After the discovery of some smart object, its data should be gathered (e.g., the identifier of the smart object and its offered services) (step 7). This data is provided so the Data Processing can consume them. Data Processing should to query semantic information about these smart object data in IoTTV-Ont (steps 8–9). So Data Processing has semantic information about the discovered smart objects in the physical environment. After that, the Data Processing stores these data in local memory (step 10) and publishes them in the broker (step 11). The storage of these data in Data Processing is essential because it allows TV applications to request the republication of smart object discovery whenever needed. Therefore, such stored data must be updated whenever new smart objects are detected in the environment as well as when they lose communication with the IoT middleware. In the latter case, Data Processing must remove from the local storage of the data of the disconnected smart object. When the Communication Monitor receives the semantic data of the smart objects (step 12), it must store or update it if the store already has those data locally (step 13). So TV applications can make queries over on smart objects using this local storage as a search source since it remains up-to-date. Also, Communication Monitor must notify the TV application whenever new values of smart objects in which it has the interest are received, enabling those applications to react instantly to these new data.

4.2 Supports TV application to change aspects of the physical environment

For the TV application to change aspects of the physical environment, it is necessary to be aware of which smart objects are available in the environment (Section 4.1). After this, TV applications can use smart objects services to change in the environment. Figure 4 shows the steps for TV applications to change aspects of the physical environment.

Firstly, Data Processing (step 1) is started; it initializes and establishes the connection to the broker (steps 2–3). On the TV application side, the Communication Monitor connects to the broker (step 4). The TV application should use the Communication Monitor API to query its local storage of smart objects and look at the services available in the physical environment (steps 5–7). Also, using the Communication Monitor API, the TV application publishes the action commands to the smart objects (steps 8–9). Data Processing receives this data since it is a subscriber to the data coming from the TV applications (step 10). After that, the Data Processing should search the smart objects in its local storage (step 11) and send commands to the IoT middleware to trigger them (steps 12–13).

5 Implementation

In the IoT context, it is expected that there will be billions of smart objects, where each one has its specifics, such as services and communication protocols. **(1) IoT middleware** can communicate with several smart objects and provides means of accessing them through an API [16]. It can work as a gateway enabling smart objects to publish their data in cloud services [4, 37].

The Mobile Hub (M-Hub) middleware is a general-purpose middleware (running on a conventional personal mobile device) that enables discovering nearby smart objects and enabling unicast, broadcast, and multicast communication. It is an individual node in the

network that works as a gateway in which objects connect to send gathered data from the environment to the Internet. It also gathers internal sensor data of mobile devices to enrich the data gathered from smart objects (e.g., by assigning the location of the Global Position System - GPS).

This IoT middleware supports mobile smart objects present in the environment. The mobility support is essential in scenarios where either smart objects or M-Hub gateway is mobile [12]. In this way, some scenarios can be explored, e.g., the smart objects are static and the M-Hub is mobile; it moves in the environment, approaching smart objects and connecting to communicate in an opportunist way. Other scenarios that can be explored: the M-Hub is not moving while the smart objects move in the environment approaching or moving away from it and the scenarios in which both smart objects and M-Hub are in motion [35].

The M-Hub services discovery occurs through a generic communication protocol, independent of the communication technology called Short-Range Sensor, Presence, and Actuation (S2PA). This protocol provides an API for using smart object services, which in turn can use different short-range communication technologies. We used S2PA to connect and exchange data with smart objects, such as to receive data from sensors and send commands to actuators as well as generate notifications about smart objects that are disconnected.

There are different middleware for Digital TV, such as the ATSC used in the United States, HBBTV used in the European TV system, and the Ginga middleware, used in South America and Africa. Ginga middleware is an ITU-T recommendation for IPTV [20]. All those middleware abstract the specificity of its functional modules and provide a programming interface for application development (API). They also support the run of

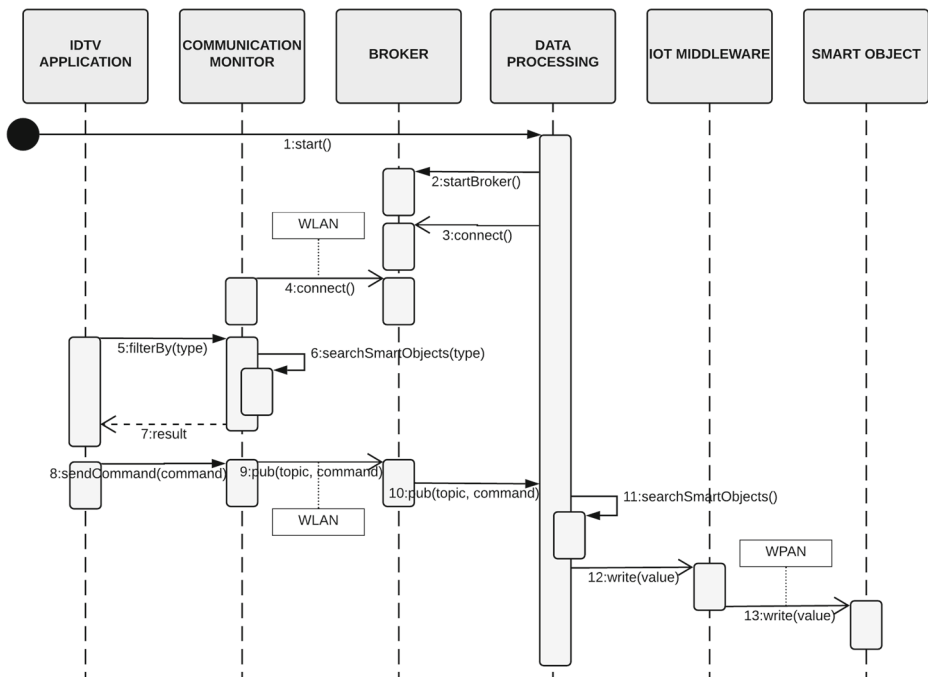


Fig. 4 Sequence Diagram: sending data from TV applications to smart objects

developed applications using declarative languages such as HTML, and NCL. Among these middlewares, we chose to use the Ginga as **(2) Digital TV middleware**.

The ontology used in this architecture was the **(3) IoTTTV-Ont** (Section 3). The access to instances of the ontology is done through a variation of the framework for ontology named Jena,⁷ developed for Android (Jena-Android⁸). The schema of IoTTTV-Ont ontology and its instances have been written in OWL. These instances model a residence containing some smart objects such as temperature and humidity sensors and some actuators such as dimmer and ventilation systems.

The **(4) CEP engine** used in this prototype is a variation of CEP Esper⁹ developed for Android named Asper¹⁰, which, in addition to meeting the specifications required in the architecture, is an open-source project. It uses Inference and Event-Condition-Action (ECA) clauses, for it analyzes the events in a data stream conditions-based specified by users. It triggers an action if it meets some specified condition. We develop these conditions in the form of CEP rules using the Event Processing Language (EPL), which is a declarative language that extends all Structure Query Language (SQL) (SQL-like extended) operators.

We implemented the **(5) Data Processing**¹¹ as a module for Android applications. The Data Processing implementation uses the resources of components in which it is related to this software architecture and provides an API for mobile applications to consume its resources.

A feature of this software architecture is the possibility of using several IoT middleware. For this purpose, we used software design patterns to allow the replacement of this middleware without much effort. One of the used patterns was the adapter design pattern since this design pattern allows adapting the interface provided by the Data Processing to the interfaces expected by several other IoT middleware. In this prototype, we used the M-Hub middleware to materialize the software architecture since it meets the requirements of this architecture regarding IoT middleware.

We use a **(6) Broker** for mobile devices that uses the MQTT protocol (Message Queuing Telemetry Transport communication).¹² We use it because it is lightweight and it was designed for devices with limited resources, low bandwidth, high latency, or unreliable [18]. We chose the broker named Moquette¹³ because it meets the requirements by the architecture regarding the broker. For the Data Processing to publish and subscribe to the Moquette broker's topics, it uses the client for MQTT brokers named MQTT Eclipse Paho Android Service 1.0.2¹⁴ (Android version). To enable the Communication Monitor to publish and subscribe to the broker, a Lua MQTT Client Library 0.2¹⁵ was used.

We chose Ginga middleware as digital TV middleware. This middleware supports Lua language for developing TV applications. Therefore, we used the Lua language to implement the **(7) Communication Monitor**¹⁶ in this prototype. This implementation followed

⁷<http://jena.apache.org>

⁸<http://github.com/sbrunk/jena-android>

⁹<http://espertech.com/esper>

¹⁰<http://github.com/mobile-event-processing/Asper>

¹¹<http://github.com/makleystonltdi/SDPEU>

¹²<http://mqtt.org>

¹³<http://github.com/technocreatives/moquette>

¹⁴<http://eclipse.org/paho>

¹⁵http://github.com/geekscape/mqtt_lua

¹⁶<http://github.com/makleystonltdi/iDTVModules>

all the specifications presented in this work, including the communication abstractions with the broker and the programming abstractions for the development of TV applications mentioned in Section 4. These programming abstractions are offered through an API that reflects the terms and relations present in the IoTTTV-Ont.

To create TV applications, the developers should import the Communication Monitor for their applications and should observe the structure of IoTTTV-Ont, since the terms and hierarchy of this ontology guides the development of these applications. For example, for the developer to specify in a TV application the commands to adjust the brightness of the physical environment, the developer must use IoTTTV-Ont to identify the control functionality of the smart object responsible for lighting management. If the smart object was a “dimmer,” then its control functionality would be “LightRegulationFunctionality.” A demonstration of the use of this approach in Lua language for the example mentioned is shown in Algorithm 1.

Algorithm 1 Sample code to send data to the smart object

```

1 local CM = require("CommunicationMonitor")
2 local smartObjects = CM.getSmartObjects
3
4 -- Function to turn on the dimmer
5 function dimmerOn()
6     local dm = smartObjects:filterByType("DimmerLamp")[1]
7     dm.Functionality.ControlFunctionality["↔
        LightRegulationFunctionality"].Command["SetCommand"]↔
        .realStateValue = "0"
8     CM.postSmartObject(dm)
9 end

```

For a TV application to manipulate the data received from smart objects, the developers must use IoTTTV-Ont to specify which notification functionality of smart objects they want to use. Algorithm 2 illustrates the use of the Communication Monitor to receive data from a presence sensor.

Algorithm 2 Sample code to receive data from smart object

```

1 local CM = require("CommunicationMonitor")
2 local presencesensor = {}
3 -- to filter only bedroom sensor, remove the comment from follow↔
   code
4 -- presencesensor.Environment = "bedroom"
5 presencesensor.Type = "presencesensor"
6 presencesensor.receiveNotification = function(↔
   notificationfunctionality) ... end
7 CM.setSmartObjectsListener({presencesensor})

```

We also developed: software for Android smartphones, which uses this prototype; TV applications for Brazilian digital TV (using the Ginga-NCL middleware) in which import the Communication Monitor; and two smart objects for sensing and gathering data in the environment. The smart object was developed using the electronic prototyping platform Arduino¹⁷ and electronic components. We named the smart objects of *objAction* and *objSensing*. The *objAction* has two services: a regulator of the lighting and the ventilation

¹⁷<http://arduino.cc>

system of the physical environment, and *objSensing* gathers context data of the physical environment, such as temperature data humidity, and presence. Both smart objects use the Bluetooth Classic communication modules.

6 Use cases

The main contributions from this work are a software architecture that allows TV and smart object integration and an ontology that allows the TV and IoT middleware exchange data regarding the residence and the smart objects/people present inside it. To demonstrate these contributions we developed use cases utilizing the prototype presented in Section 5. All the use cases show how the viewers have an increase in the level of experience, such as immersive content, personalization, and multimodal interactions.

The use cases also explore the features specified in Section 4. Some of these use cases are available in the literature, such as the generation of immersive environments regarding content played on TV [17, 31]. However, these scenarios have no user identification and do not provide a mean for handling data from complex events. We seek to reuse these use cases; however, we used our approach, which uses the IoTTV-Ont to define a common vocabulary, and the software architecture to exchange data between the digital TV and IoT middleware. Both the prototype and the recorded videos of use cases are available on the IoTTV¹⁸ project website.

We used the M-Hub middleware in all use cases, as IoT middleware, and we used Ginga-NCL, as digital TV middleware. A Linux virtual machine for the VMWare¹⁹ tool runs the Ginga-NCL middleware.

The hardware parameter settings of the Ginga-NCL virtual machine were maintained according to its downloadable image on the website of Ginga-NCL.²⁰ VMWare was running on an Intel i7 7th generation laptop with 16 GB of RAM and had eight cores of 2.7 GHz frequency processing and Linux Ubuntu 16.04 operating system. The use cases use a Mi5X smartphone from Xiaomi manufacturer to enable the communication between smart objects and the TV application (master portable device). Mi5X smartphone has the following specifications: Qualcomm Snapdragon 625 2.0 GHz Octa-Core processor and 4 GB RAM, in addition to the Android 7.2 operating system. Other smartphones (slave portable devices), also with Android operating system, were used to simulate several users and the multimodal interactions performed by them with the TV applications. Slave portable devices have the following specifications: Motorola Moto G2 with 1 GB of RAM and Qualcomm Snapdragon 400 Quad-Core processor of 1.2 GHz; and ZenFone 2 from ASUS manufacturer, which features 4GB of RAM and Z3580 Intel Atom 2.3 GHz Quad-Core processor.

6.1 Use Case 1: TV application aware of the physical environment state

This use case aims to provide viewers the experiences regarding how to interact with the TV application and immersion in the content presented. For the multimodal interactions of viewers, this use case uses viewers' smartphones as a tool for detecting movements with hand. For the immersive experience of viewers regarding the content played on TV, this use

¹⁸<http://lsdi.ufma.br/~iottv>

¹⁹<http://vmware.com/products/workstation-player.html>

²⁰<http://gingancl.org.br/pt-br/ferramentas>

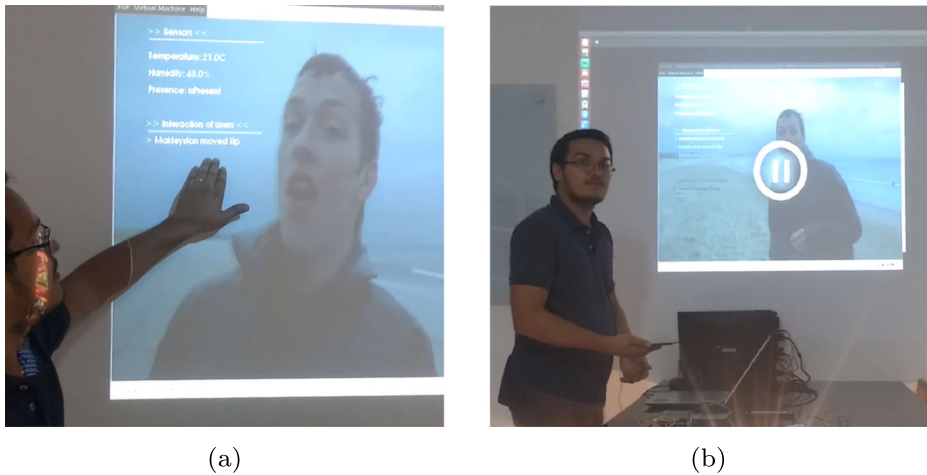


Fig. 5 Images captured during the demo: **a** the TV application display context data of physical environment on screen; **b** TV content is interrupted after the user's interactions

case shows the TV application managing smart objects present in the physical environment in sync with the TV application.

This use case uses the *objSensing*'s notification features to gather the environment data. The M-Hub dynamically discovers the *objSensing* and establishes a communication with it through a generic communication protocol named S2PA. The developed applications used IoTTV-Ont to identify viewers through the Person class, to identify multimodal interactions through the Modality and Event classes, and to handle the values of smart objects through the "NotificationFunctionality" class. The detection of viewer interactions occurs through the use of the CEP engine present in Data Processing. It tracks the data stream of the smartphone's internal sensors to identify the interaction events.

Figure 5 shows the reproduction of multimedia content in which its narrative consists of a music clip presentation. The data consumed by the TV application follow the steps shown in the sequence diagram illustrating by Fig. 3. The TV application has a code similar to the presented in Algorithm 2.

In this demo, the TV application receives the data of discovered smart objects in the environment and displays it on the screen. The TV displays the sensor name and its gathered values whenever new values of sensors are read. Figure 5a shows the TV during the presentation of the video in which it displayed the context data of the physical environment. We coded the TV application to have ten-second tolerance without detecting movements in the physical environment by the presence sensor. After this period, the TV screen displays the notification of the absence of movements in the room.

During the video, viewers can use their smartphones to interact with the TV application. When they move their phones, the device's internal sensors generate events that are published in the broker. Consequently, the TV application is able to receive these events from interactions because it imported the Communication Monitor. Our TV application was coded to interrupt and resume the play video after certain hand movements. The movements to the left and right, in that order, interrupt playback of the video and the up and down directional movements, in that order, resume playback of interrupted media. Figure 5b shows the instant that a viewer has just made a sequential left and right movements with his smartphone, thus interrupting the presentation of the presented content.

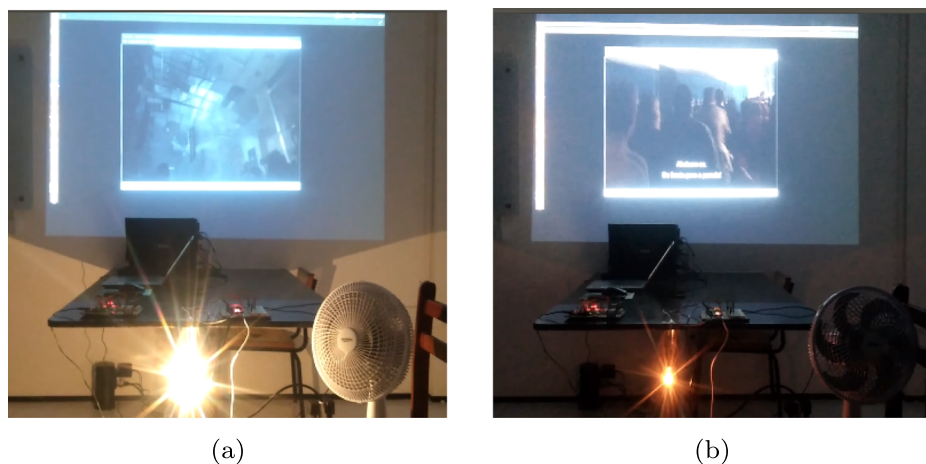


Fig. 6 Images captured during the demo: **a** the TV application adjusts the dimmer for 75% power and turn on the ventilator; **b** the TV application adjusts the dimmer for 25% power and turn off the ventilator

This use case allows us to think of other examples that explore multimodal interactions, such as quizzes for TV applications or voting systems. A TV application can also use multimodal interactions, the viewer's profile, and the context of the environment to personalize the content presented.

6.2 Use case 2: A mulsemedia TV application

This use case aims to provide the viewer with an immersion experience regarding content being played on TV through the synchronous management of the smart objects present in the physical environment with the content presented. In this use case (illustrated in Fig. 6), the smart object that works in the physical environment is triggered several times by the TV application to reproduce effects that seek to increase the immersion level of the viewer to the content played on TV. The desired sensory effects are the regulation of luminosity and the triggering of the environment's ventilation system.

In this use case, we use the smart object *objAction* to manage the ventilation system and the ambient lighting. The M-Hub dynamically discovers the *objAction* and establishes a communication with it through a generic communication protocol named S2PA. For the TV application to manage the available services, it uses the *ControlFunctionality* class present in IoTTV-Ont. More precisely, the control class "LightRegulationFunctionality" is used to manage the "dimmer lamp" and "OnOffFunctionality" to activate the "ventilator." The smart objects triggered by the TV application follow the steps shown in the sequence diagram illustrating by Fig. 4. The TV application uses code similar to the presented in Algorithm 1 together with markings in the video timeline. More precisely, the function calls present in lib Communication Monitor are made at the desired times for the TV application. In this case, the programmer specified the times that the fan should be activated during the video execution.

The media on playback has several scenes with several illumination intensity levels and there are scenes where the actor receives gusts of wind. Figure 6a shows the instant when the lighting of the physical environment is 75% of the maximum intensity of the dimmer lamp and the ventilation system was turned on. Figure 6b shows the instant in which the lighting

intensity of the physical environment is 25% and the ventilation was turned off. In this use case, the prototype discovered smart objects and their services offered in the physical environment, and established communication with them using the Bluetooth technology.

7 Final remarks

This work aimed at integrating TV and IoT middleware to allow TV applications: consume data and services offered by the smart objects present in the physical environment and change the aspects of this environment. For that purpose, our proposal seeks to export for TV applications the context-aware of the environment, including the smart objects and viewers' profile. We demonstrate such integration by developing a prototype and use cases.

We summarize the main contributions of this work as follows:

- Definition of a conceptual model that describes the physical environment through an ontology-based approach: one of the challenges addressed in this work concerns the need to export to the TV application the knowledge of the physical environment, smart objects, and people inside it. For that purpose, we developed an IoTTV-Ont ontology (Section 3) that allows representing a house, the smart objects, and the people. Additionally, this ontology allows describing the interactions of the viewers with TV applications.
- Software architecture (Section 4) that integrates digital TV and IoT middleware. This architecture is according to the conceptual model (IoTTV-Ont) defined in this work. Our software architecture defines a set of components responsible for exporting the physical environment's context data to a TV application, allows the TV application to trigger smart objects in this environment, and provides new forms of a user to interact with the application running on the TV. As it is a model that addresses only the specifications of components, interfaces, and interactions, its use is not limited to any digital TV and IoT middleware.
- Prototype of the proposed software architecture using the M-Hub middleware as IoT middleware, and the Ginga-NCL middleware as a digital TV middleware. This prototype is available for download on the IoTTV project website along with its documentation.

We can extend this work in some ways. Currently, many viewers use mobile devices while watching TV content. Many television programs use these mobile devices as a second screen to display additional content. This way, we intend to extend this work so that our software architecture supports the second screen applications. As a further possibility, we may develop an authoring tool to assist programmers in the development of TV applications that explore IoT resources according to the proposal presented in this work. Finally, works like [1] and [23] seek to automatically detect sensory effects in multimedia content through audio and images. In this way, we intend to extend our work to automatically support the detection of sensory effects of multimedia contents and use the services available in the physical environment to trigger them according to the detected sensory effects.

References

1. Abreu R, Mattos D, dos Santos JAF, Muchaluat-Saade DC (2019) Semi-automatic synchronization of sensory effects in mulsemmedia authoring tools. In: Proceedings of the 25th Brazilian symposium on multimedia and the Web - WebMedia 19, ACM Press. <https://doi.org/10.1145/3323503.3360302>
2. Al-Fuqaha A, Guizani M, Mohammadi M, Aledhari M, Ayyash M (2015) Internet of things: a survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials* 17(4):2347–2376. <https://doi.org/10.1109/COMST.2015.2444095>
3. Atzori L, Iera A, Morabito G (2010) The internet of things: a survey. *Computer Networks* 54(15):2787–2805. <https://doi.org/10.1016/j.comnet.2010.05.010>
4. Biljana L, Kire V (2017) A review of internet of things for smart home: Challenges and solutions. *J Clean Prod* 140:1454–1464. <https://doi.org/10.1016/j.jclepro.2016.10.006>
5. Bonino D, Corno F (2008) DogOnt - Ontology Modeling for Intelligent Domotic Environments. Springer, Berlin, pp 790–803. https://doi.org/10.1007/978-3-540-88564-1_51
6. Brickley D, Miller L. (2007) Foaf vocabulary specification 0.91. Tech. rep., FOAF
7. Cesar P, Chorianopoulos K (2009) The evolution of tv systems, content, and users toward interactivity. *Foundations and Trends® in Human–Computer Interaction* 2(4):279–373. <https://doi.org/10.1561/1100000008>
8. Chen H, Finin T, Joshi A (2005) The SOUPA Ontology for Pervasive Computing. Birkhäuser Basel, Basel, pp 233–258. https://doi.org/10.1007/3-7643-7361-X_10
9. Christophe J, Ahmed M, Yacine B (2013) An ambient assisted living framework with automatic self-diagnosis, vol 5
10. Etzion O, Niblett P (2011) Event Processing in Action, 1st edn. Manning Publications Co., Greenwich, CT USA
11. Eugster P, Felber P, Guerraoui R, Kermarrec A (2003) The many faces of publish/subscribe. *ACM Comput Surv* 35(2):114–131. <https://doi.org/10.1145/857076.857078>
12. Gomes B, Muniz L, Silva FJ, Ríos L. E. T., Endler M (2017) A comprehensive and scalable middleware for ambient assisted living based on cloud computing and internet of things. *Concurrency and Computation: Practice and Experience* 29(11):e4043. <https://doi.org/10.1002/cpe.4043>
13. Görgü L, Kroon B, Campbell A, O'Hare G (2013) Enabling a Mobile, Dynamic and Heterogeneous Discovery Service in a Sensor Web by Using AndroSIXTH. Springer International Publishing, Cham, pp 287–292. https://doi.org/10.1007/978-3-319-03647-2_26
14. Gruber T (1993) A translation approach to portable ontology specifications. *Knowl Acquis* 5(2):199–220. <https://doi.org/10.1006/knac.1993.1008>
15. Guarino N (1998) Formal Ontology in Information systems. In: Proceedings of the 1st international conference June 6–8, 1998, Trento, Italy. 1st edn. IOS Press, Amsterdam
16. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013) Internet of things (iot): A vision, architectural elements, and future directions. *Future Gener Comput Syst* 29(7):1645–1660. <https://doi.org/10.1016/j.future.2013.01.010>
17. Guedes Á, Cunha M, Fuks H, Colcher S, Barbosa S (2016) Using ncl to synchronize media objects, sensors and actuators. In: XXII Simpósio Brasileiro de Sistemas Multimídia e Web (Vol. 2): Workshops e Sessão de Pôsteres., Webmedia, Porto Alegre: Sociedade Brasileira de Computação, pp 184–189
18. Hunkeler U, Truong H, Stanford-Clark A (2008) Mqtt-s; a publish/subscribe protocol for wireless sensor networks. In: 2008. COMSWARE 2008. 3rd international conference on Communication systems software and middleware and workshops, pp 791–798
19. ISO (2016) ISO/IEC 23005-3:2016: Information technology – Media context and control – Part 3: Sensory information
20. ITU-T (2014) Nested context language (ncl) and ginga-ncl. Recommendation H.761, International Telecommunication Union, Geneva
21. Kapri A, Adalgeirsson Y, Kator K, Chung J, Holtzman H Television (tv) as an internet of things (iot) participant (2017). US Patent App. 15/087,813
22. Lima E, Rabêlo R (2015) An architectural model for communication between the idtv and mobile devices. In: 2015 international conference on Computing, networking and communications (ICNC), IEEE, pp 1102–1105
23. Lojka M, Pleva M, Kiktová E, Juhár J, Čížmár A (2015) Efficient acoustic detector of gunshots and glass breaking. *Multimedia Tools and Applications* 75(17):10441–10469. <https://doi.org/10.1007/s11042-015-2903-z>

24. Luckham D (2008) The power of events: an introduction to complex event processing in distributed enterprise systems. In: Bassiliades N, Governatori G, Paschke A (eds) Rule representation, interchange and reasoning on the web. Springer, Berlin, pp 3–3
25. Miorandi D, Sicari S, Pellegrini FD, Chlamtac I (2012) Internet of things: Vision, applications and research challenges. *Ad Hoc Netw* 10(7):1497–1516. <https://doi.org/10.1016/j.adhoc.2012.02.016>
26. Noy N, McGuinness D (2001) Ontology development 101: a guide to creating your first ontology. Tech. rep., Stanford Knowledge Systems Laboratory
27. Perera C, Jayaraman P, Zaslavsky A, Christen P, Georgakopoulos D (2014) Mosden: an internet of things middleware for resource constrained mobile devices. In: 2014 47Th hawaii international conference on system sciences, pp 1053–1062. <https://doi.org/10.1109/HICSS.2014.137>
28. Perera C, Zaslavsky A, Christen P, Georgakopoulos D (2014) Context aware computing for the internet of things: a survey. *IEEE Communications Surveys Tutorials* 16(1):414–454. <https://doi.org/10.1109/SURV.2013.042313.00197>
29. Punt M, Bjelica M, Zdravkovic V, Teslic N (2015) An integrated environment and development framework for social gaming using mobile devices, digital tv and internet. *Multimedia Tools and Applications* 74(18):8137–8169. <https://doi.org/10.1007/s11042-014-2045-8>
30. Rosa R, Lucena V (2017) Contextualizing and capturing individual user interactions in shared itv environments. *Multimedia Tools and Applications* 76(6):8573–8595. <https://doi.org/10.1007/s11042-016-3489-9>
31. Saleme E, Santos C (2015) PlaySEM: A platform for rendering MulSeMedia compatible with MPEG-v. In: Proceedings of the 21st Brazilian symposium on multimedia and the Web, WebMedia '15. ACM, pp 145–148. <https://doi.org/10.1145/2820426.2820450>
32. Silva V, Maia O, Rodrigues M, Lucena V (2016) Universal system for integrating commercial medical devices with standardized digital tv system. In: 2016 IEEE International conference on consumer electronics (ICCE), pp 301–304. <https://doi.org/10.1109/ICCE.2016.7430622>
33. Sommaruga L, Formilli T, Rizzo N (2011) Domoml: an integrating devices framework for ambient intelligence solutions. In: Proceedings of the 6th international workshop on enhanced web service technologies, WEWEST '11. ACM, New York, pp 9–15. <https://doi.org/10.1145/2031325.2031327>
34. Sulema Y (2016) Mulsemedia vs. multimedia: State of the art and future trends. In: 2016 International conference on systems, signals and image processing (IWSSIP), pp 1–5. <https://doi.org/10.1109/IWSSIP.2016.7502696>
35. Talavera L, Endler M, Vasconcelos I, Vasconcelos R, Cunha M, Silva F (2015) The mobile hub concept: Enabling applications for the internet of mobile things. In: 2015 IEEE International conference on pervasive computing and communication workshops (percom workshops), pp 123–128. <https://doi.org/10.1109/PERCOMW.2015.7134005>
36. Wehbi A, Cherif A, Tadj C (2012) Modeling ontology for multimodal interaction in ubiquitous computing systems. In: Proceedings of the 2012 ACM conference on ubiquitous computing, UbiComp '12. ACM, New York, pp 842–849. <https://doi.org/10.1145/2370216.2370408>
37. Zhu Q, Wang R, Chen Q, Liu Y, Qin W (2010) Iot gateway: Bridging wireless sensor networks into internet of things. In: 2010 IEEE/IFIP 8th international conference on Embedded and ubiquitous computing (EUC). IEEE, Hong Kong, pp 347–352. <https://doi.org/10.1109/EUC.2010.58>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Danne Makleyston G. Pereira He holds a degree in Information Systems from Paraíso do Ceará College (FAP). He holds a postgraduate degree in Systems and Network Security from the Faculdade Leão Sampaio (FALS). He is a professor of Informatics with an emphasis in Database of the Federal Institute of Tocantins, IFTO - Campus Colinas of the Tocantins. He holds a master's degree in computer science from the Federal University of Maranhão - UFMA - in the line of research "Architecture of Computational Systems," having a study area the distributed systems. He is a member of the Laboratory of Distributed and Intelligent Systems - LSDi. He has professional experience in database and web, mobile, and desktop systems development.



Francisco José da S. e Silva Francisco Silva is an Associate Professor in the Department of Informatics at UFMA, having started his academic career in 1991. He is currently working as a graduate student in Computer Science and currently works as coordinator of the postgraduate program in Computer Science, where he has been working since 2011. He is also a permanent professor of the post-graduation program in Electrical Engineering at UFMA since 2003. He holds a doctorate in Computer Science from the Institute of Mathematics and Statistics of the University of São Paulo (USP) 2003, Ph.D. in the Information Technology Department of PUC-Rio between 2014 and 2015. He is the coordinator of the Laboratory of Intelligent Distributed Systems (LSDi - <http://www.lsdι.ufma.br>) of UFMA. His research area includes distributed and ubiquitous computing systems, with emphasis on the Internet of Things and Smart Cities. He is a member of the Brazilian Computer Society (SBC) and ACM (Association for Computing Machinery) and has served as a member of the program committee of important national and international conferences, as well as a reviewer of important periodicals related to distributed systems and ubiquitous computing



Carlos de Salles S. Neto He holds a degree in Computer Science from the Federal University of Maranhão (2000), a master's degree and a Ph.D. in Computer Science from the Pontifical Catholic University of Rio de Janeiro (2003 and 2010). He is currently an associate professor at the Federal University of Maranhão where he is the Coordinator of TeleMedia / MA and LAWS (Advanced Web Systems Laboratory). He is an associate researcher at the Telemedia Laboratory of PUC-Rio. He has experience in the area of Computer Science, with an emphasis on Hypermedia, working mainly in the following subjects: multimedia applications, multimedia document engineering, and educational data analysis. He was Coordinator of the Special Committee on Multimedia and Web Systems of the Brazilian Computer Society (2015-2016) and is currently a member of its Steering Committee.



Davi Viana dos Santos PhD. and Master in Informatics by the Post-Graduation Program in Informatics of the Federal University of Amazonas. He is graduated in Computer Science from the Federal University of Amazonas (UFAM). He holds a technical course in computer science from Nokia Teaching Foundation. He is currently Adjunct Professor at the Federal University of Maranhão (UFMA). Also, he is a permanent member of the Post-Graduation Program in Computer Science (PPGCC) of UFMA and Director of the Division of Dissemination of Entrepreneurship of UFMA. He is a member of SBC's Special Committee on Information Systems (CESI) and vice-regional secretary of SBC at UFMA. He is interested in the areas of software quality, software process improvement (MPS), implementation of MPS programs with an emphasis on the adoption of maturity models, Teaching in Software Engineering, Experimental Software Engineering, Information Systems, Software Engineering for the Internet of Things and Software Startup.



Luciano Reis Coutinho Graduated in Computer Science from the Federal University of Maranhão (1996), Master in Computer Science from the Federal University of Paraíba (1999) and Ph.D. in Sciences from the Electrical Engineering program of the University of São Paulo (2009). He is currently Associate Professor II at the Federal University of Maranhão. He has experience in Computer Science, with emphasis on Distributed Artificial Intelligence, working mainly on the following topics: multiagent systems, intelligent systems, machine learning and software engineering.



Álvaro L. V. Guedes holds a Ph.D. (2017) from the Pontifical Catholic University of Rio de Janeiro, where he acts as Researcher Assistant in TeleMedia Lab. He received his Bachelor (2009) and M.Sc. (2012) degrees in Computer Science from the Federal University of Paraíba, where he worked as researcher in Lavid Lab. In both labs, Álvaro worked in interactive video and TV research projects, and acquired experience in C++/video development and research writing/publication. At Lavid, he worked in awarded projects such Ginga Store and Brasil 4D. At TeleMídia, he contribute to the Ginga and NCL specifications which today are standards for DTV, IPTV and IBB. His research interests include Multimedia, Interactive Video, Immersive Media, and Deep Learning for Multimedia.