Received: February 28, 2020. Revised: May 11, 2020.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

ACAIOT: A Framework for Adaptable Context-Aware IoT applications

Manar Elkady[1]* **Abeer ElKorany[1]** **Amin Allam[1]**

[1]*Computer Science Department, Faculty of Computers and Artificial Intelligence, Cairo University, Giza, Egypt*
* Corresponding author's Email: m.elkady@fci-cu.edu.eg

Abstract: Recently, the Internet of Things (IoT) and context-aware IoT applications are involved in various domains such as healthcare, traffic, and smart homes. The main challenge, while developing context-aware IoT applications, is managing the massive amounts of data and events to get relevant context information. This paper proposes a semantic-based and domain-independent framework called ACAIOT. It hides the details of context management from the high-level services. It proposes a programming abstraction for adapting services' execution to various domains easily. We demonstrate the features and effectiveness of ACAIOT by its execution on a real smart home dataset. We evaluated ACAIOT against previous work in developing a set of services such as monitoring, prediction, and notification. Results show that ACAIOT achieves an average F1-score of 0.82, which is comparable with the current state-of-the-art methods. Moreover, ACAIOT is able to incorporate important compound activities that cannot be handled by previous work.

Keywords: Context-awareness, Context modeling, Ontology, Internet of things.

1. Introduction

Future Internet applications require access to information generated from a variety of heterogeneous devices. As a result, the Internet of Things (IoT) [1] has appeared as an innovative paradigm to enrich the communication between human beings and the smart things that include sensors, services, and other Internet-connected objects. IoT has been evolved to acquire different types of context information with the aid of sensing abilities. IoT utilizes context information processing and storing capabilities for generating knowledge and offering services [2]. Accordingly, IoT applications that rely on the context information to react and adapt its current state are known as context-aware IoT applications [3]. The context information is managed through context-aware middleware. The context-aware middleware acts as an integration point that manages the context to enable applications to be context-aware [4]. Moreover, context-aware middleware provides a homogeneous interface for IoT applications and services development that involves generic context management solutions. Increasing the level of context awareness is still an open issue in designing middleware [5]. Multiple comparative studies have revealed that there is no context-aware middleware architecture that complies with all IoT middleware requirements. Additionally, context-aware middleware solutions still face multiple challenges [5-7]. First, managing the heterogeneous and massive amount of data acquired from multiple things. Second, Organizing and handling events and how to recognize higher activities from a set of simple events. Finally, using the developing framework to implement applications in various domains. This paper proposes a domain-independent framework that enables flexible development for adaptive context-aware IoT applications by enhancing different characteristics for IoT middleware. Two main categories of middleware characteristics are enhanced. The first category is the services that a middleware should provide such as data management and event management. While, the other one emphasizes the characteristics that a context-awareness middleware architecture should

support such as *programming abstraction, interoperability, service-based,* and *adaptability.*

Those characteristics are supported by providing the IoT application developer with a framework that contains a set of *API* and *Service templates* that facilitates the implementation of cloud services. Furthermore, the framework's *Context Management* is responsible for encapsulating and hiding the details of data and event management. Also, it provides context-aware as a service for cloud service developers, as will be explained in section 4.

Let us define the following example, in a smart home domain, activity recognition inside a smart home highly depends on the context information acquired from smart home sensors and devices. Therefore, it is required to manage the all types of data and events acquired from sensors and devices to generate relative context information. For example, simple events (such as the door is opened) and recognized activities (e.g., the person goes out of home) are translated to context information. It could be accessed by services as *monitoring* home events (e.g., sleep, eat), and *notification* with emergencies for a patient or an elder (e.g., go out of home after midnight, sleeping more than 10 hours). Therefore, the proposed framework aims to provide the developer with templates that support an easy way to develop monitoring service, notification service, and prediction of current patient status. Furthermore, the same set of services (e.g., monitoring, notifying, prediction) could be applied for different domains such as road traffic, smart city, smart grid with different execution parameters. Thus, designing templates for such standard services will ease the development of context-aware IoT applications in different domains. Designing such abstractions should support scalability and interoperability between applications and services [8].

Therefore, the main objective of this paper is to develop a framework for Adaptive Context-Aware IoT applications (ACAIOT) that supports the following:

- Encapsulating and managing both of the data and events in a separate context management Layer that responsible for generating relevant context information.
- Support semantic interoperability by using ontology in order to propose a domain-independent context management process.
- Incorporate a novel rule structure used to facilitate activity recognition.
- Providing a high-level API and *Service templates* (ACAIOT Library) that facilitates the implementation of cloud services.

We justify the effectiveness of the proposed framework by comparing its architecture with recent research studies. Also, we use ACAIOT to implement smart home application services by using a real dataset. To evaluate ACAIOT capability, we compared those services' outputs with the dataset annotations, and an average of 0.82 F1-score is obtained. With comparisons with other approaches, we conclude that ACAIOT rule structure for compound activity recognition outperformed other various types of services in the smart home domain.

This paper is organized as follows: Section 2 presents the background of the research work. Section 3 illustrates some of the related works. Section 4 presents the general architecture of the proposed framework for Adaptive Context-Aware IoT (ACAIOT) applications. Section 5 describes the detailed context management process. The proposed services' templates are described in section 6. A smart home domain case study is applied for testing the ACAIOT framework is shown in section 7. The paper is concluded in section 8.

2. Background

2.1 IoT applications and middleware

One of the critical technology in realization of IoT systems is middleware, which usually described as a software system designed to be the intermediary between IoT devices and applications [5] and to facilitate the interaction between a multitude of different devices and data [9]. Middleware should have specific characteristics that cover both services requirements and architecture requirements. Service requirements are concerned with managing and discovering available resources, data management event management, and code management [5]. In this paper, we target data management and event management improvements. An IoT middleware needs to provide data management services to applications, including data processing, preprocessing, and data storage. Preprocessing concerns applying data filtering and data aggregation. Also, data management components are utilized to get relevant context information and storing context information in standard form. Event management concerns how to get meaningful events from the simply observed events. Events can be simple or complex. Most middlewares statically predefine how an event is handled, but they are not considering complex events and not considering the difference between discrete (e.g., a door opens) and continuous events (e.g., driving a car)

Middleware architecture requirements are *programming abstraction, interoperability, service-based,* and *adaptability*. *Programming abstraction* is supported by providing an API to isolate the development of the applications or services from the underlying low-level programming details. *Interoperability* in a middleware can be viewed from the network, syntactic, and semantic perspectives. A network should exchange information across different networks. Syntactic interoperation should allow for heterogeneous formatting and encoding structures of any exchanged information or service. Semantic interoperability refers to the meaning of information or a service. Semantic interoperability, which is the focus of this paper, should allow for information exchange between applications. Semantic interoperability is a challenge due to the lack of standard in context-awareness ontologies [10]. Applying the middleware functionality in a *service-based* architecture offers high flexibility when new functions need to be added to the middleware. Moreover, the middleware needs to *adapt* or adjust its component execution to fit the user satisfaction and effectiveness of the IoT application and services.

2.2 Context-awareness

The context-aware application uses context to provide relevant information and services to the user as needed, where relevancy depends on the user's task [11]. Developing context-aware IoT applications requires a well-formed context information handling process for managing the context of life cycle phases. Context life cycle phases start by *context acquisition* from various context sources. The acquired data needs to be modeled and represented according to a meaningful manner through the *context modeling* phase. Context modeling approaches are surveyed in [12]. The most popularly adopted approach is the ontological approach. Ontologies provide an application-independent representation of a specific subject domain that is usable by multiple applications [13]. The ontology model is used to formalize the context information with a more high-level organization to establish a common understanding and unification of its concepts' meanings [14]. Then, the *context reasoning* phase gets the value from the collected data and conveys it to a high-level knowledge. Through context reasoning, deriving high-level context information from the low-level information could be applied. Finally, the *context dissemination* phase delivers context information needed to the IoT application's user who is interested in it [3,15]. The context lifecycle approach needs to be standardized. This will improve the interoperability between different middleware components as well as reusability and applicability of extracted context information [5].

3. Related works

Different frameworks have been proposed to improve the development of adaptive context-awareness applications. In this section, some of those frameworks are presented.

ELIoT [16] is a development platform for Internet-connected smart devices. It concerns improving development flexibility and reducing network latency. It supports abstraction and adaptation by enabling interactions through the REST and CoAP interface with the ability of reconfiguration. On the other hand, there is no context information modeling defined. Only network interoperability is supported by the dedicated language constructs to concentrate on different communication guarantees.

Authors in [17] defined and implemented a service-oriented domain-oriented language (DSL), allowing the straightforward definition of object-oriented context modules in applications related to different domains. A context manager is responsible for defining context entities and filtering irrelevant contexts. On the other hand, there is no semantic definition of the context model to enable further extensions and interoperability between different domains.

Econtxt [18] is a model for context-aware applications that are running in the digital ecosystem. The context information is represented as a *consource* object type. It supports multiple operations, such as filtering and aggregation operations. Adaptation is implemented by applying reactive variables, which permit the propagation of data changes in a push model. However, developers cannot customize their events and rules required for data processing. Moreover, it does not support semantic interoperability at the data level, and it lacks considering complex events.

A cooperative middleware [19] is designed to enable context-aware services for users in a smart home. Its goal is to reduce the context computation cost of providing services on time. An OWL-based ontology technique was used to define the context semantically. Middleware implementation was performed using Java and adopted Open Service Gateway Initiative (OSGi) platform to ease the control and management of services. Although rules over ontology items are used for activity recognition,

Table 1. Different frameworks support to Middleware requirements

	Service Requirements		Architectural Requirements				Context-awareness Components		
	Data Management	Event Management	Abstraction	Semantic Interoperability	Service-based	Adaptability	Context Modelling	Context Reasoning	Services
ELIOT [16]	P,PP	No	Yes	No	Yes	Yes	No	No	Monitoring
iCasa Ext [17]	P,PP	No	Yes	No	Yes	Yes	OO	OO	Monitoring
Econtxt [18]	P, PP	NC	Yes	No	No	NC	OO	OO	Monitoring
Cooperative Middleware[19]	P	NC	No	No	Yes	Yes	Ontology	Ontology Rule	Monitoring
IOTVar [20]	P	No	Yes	No	Yes	NC	OO	No	Monitoring
ESDP_IoT [21]	P,PP	NC	Yes	Yes	Yes	Yes	Ontology	Rule	Monitoring, Notification
ACAIOT	P, PP	Yes	Yes	Yes	Yes	Yes	Ontology	Ontology Rule	Monitoring, Notification, Reminder, Prediction
	Processing: P Pre-processing: PP Yes: supported No: Not supported NC: Not Completely supported OO: Object-Oriented								

but there is no management mechanism for event composition. Moreover, there is no programming abstraction for accessing and adapting the middleware services execution.

IoTVar middleware [20] is designed to overcome the lack of standardization in abstractions and APIs in IoT platforms. It is offering to application developers on the client side the possibility to declare variables that are mapped to sensors and whose values are transparently updated with sensor observations by using proxies. However, there is no semantic context modeling that can fit the development of IoT applications in different domains.

To address the heterogeneity aspect of IoT data, authors in [21], designed an approach for sensory data processing on IoT gateways. This approach manipulates data based on semantic rules to filter out redundant and insignificant data. Also, it applies a complex event processing module to analyze the coming events for detecting the most critical ones and assigning them the processing priority. Although the event handler is used to perform the aggregation and classification tasks, it is not used to recognize complex event structure. Moreover, there is no service template defined for context dissemination.

Overall, the reviewed studies have some limitations. They describe entities in an environment and applying different context management, but applying a set of operations for pre-processing is not always exist. Although using object-oriented for context modeling supports a high level of abstraction, it lacks reasoning, validation, standardization, and hierarchical structure modeling support. Moreover, most of these studies do not specify how to separate the domain knowledge components in a loosely coupled manner to be able

to develop IoT applications in different domains and to support interoperability. In addition, in most of these researches, the monitoring service is supported, but there are no unified service definitions that can be customized according to the application requirements and tailored to other types of services such as notification, reminder, and prediction which will be proposed by ACAIOT framework. Comparison between different research studies and our ACAIOT framework is shown in Table1.

4. ACAIOT framework overview

In this section, an overview of a proposed framework for Adaptive Context-Aware IoT (ACAIOT) applications is presented. The proposed framework aims to enhance middleware service requirements as well as architectural requirements by utilization of semantic technology. For the middleware service requirements, ACAIOT would enhance both data and event management. While for architectural requirements, it would support the following requirements: context-awareness, adaptability, abstraction, service-based, and semantic interoperability, as will be discussed in this section. The design of ACAIOT architecture is based on the following principles:

- Encapsulating and managing both of the data and events in a separate context management Layer that responsible for generating relevant context information.
- Providing a high-level API (ACAIOT Library) to access the backend processing to support abstraction requirements.
- Support semantic interoperability by using ontology and rules in order to enhance the context management process.

Generally, a context-aware service is adapted according to the context information, and the application requirements. In order to enhance both data and event management, different types of context information are collected from different resources such as sensors, databases, web services, or others. All context information would be analyzed and processed through the *Context Management* layer shown in Fig. 1. This layer is responsible for providing context-aware as a service for cloud service users [22]. While, the general-purpose *ACAIOT_Ontology* that is extended based on the target application domain into domain-oriented ontology (in this paper, a smart home domain will be considered). The next layer

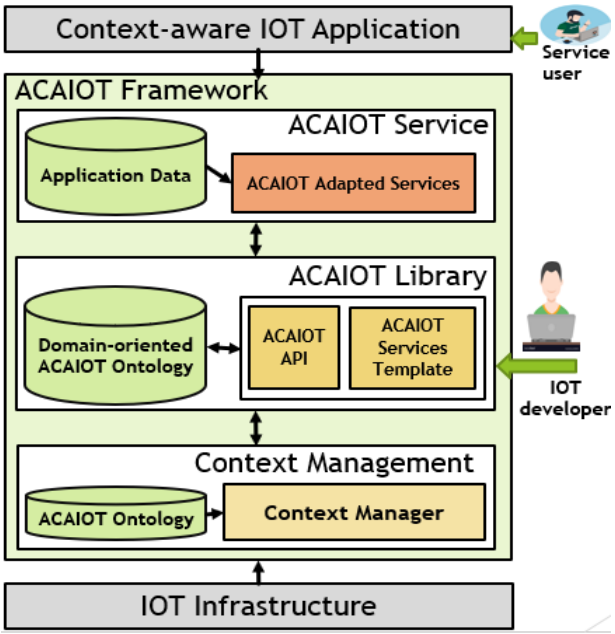

Figure. 1 ACAIOT Architecture

(*ACAIOT Library*) provides *ACAIOT API*, and *ACAIOT Service* templates. It is significant to mention that the ACAIOT framework is generally applicable for different domains as it facilitates the implementation of cloud services without re-thinking of how to handle such data and events and how to manage and deliver the context information. In the following, a brief about each of these layers that constitute ACAIOT architecture is presented.

4.1 Context management layer

Through this layer, context management is accomplished through the *Context Manager*. It is responsible for handling context modeling and context reasoning by applying a semantic solution by utilizing ontologies. The main components of the *Context Manager*, as shown in Fig. 2, are *Ontology manager*, *Rule Engine*, *Context modeling*, *Context Reasoning*. *ACAIOT_Ontology* is a general-purpose ontology for context-aware IoT applications. It includes common concepts used in IoT applications, which provides a unified interface that could be extended to various IoT domains. *ACAIOT_Ontology* will be extended by the IoT application developer during the application development process to meet the evolving application's domain concepts. Details of the context management process will be illustrated in section 5.

4.2 ACAIOT library layer

The *ACAIOT Library* is supporting context dissemination by delivering the context information to the application during its running, such that it

could adapt its execution. Therefore, *ACAIOT Library* is a programming interface that is responsible for accessing the *Context Management* at the backend. *ACAIOT Library* has an API for registering application data sources and the domain-oriented ontology and adding rules to the *Context Management. ACAIOT Library* has predefined services templates such as *Notification*, *Monitoring*, *Reminder*, and *Prediction*. The IoT developer can adapt these services templates to fit his application requirements. The adapted services will be ready to use by the application's user through the *ACAIOT Service* layer. The details of the structure of the services and how they can be delivered and adapted during the application running inside ACAIOT will be illustrated in detail in section 6.

4.3 ACAIOT service layer

Here, the *ACAIOT Adapted Services* are ready to be used by application users. The user could use them based on his application data without any prior knowledge about the backend processing. *The ACAIOT Service* layer is responsible for delivering context information. Context information is delivered to the application's user as a knowledge delivered through the adapted service executions output. ACAIOT Service layer will be described in detail in section 6.

5. ACAIOT context management process

ACAIOT context management process is applied through the *Context Manager*, which is presented in Fig. 2 The main functional components of the context management layer are organized according to context life cycle phases that are mentioned above. Assuming that the context acquisition process is done and its output is a stream of real-time data for all data sources that are registered through the *ACAIOT Library* by the developer. In the following subsection, a description of the *Context Manager* components is presented.

5.1 Context modeling

Context Modeling module is responsible for transforming the input data into context information by the aid of both of Ontology Manager and Rule Engine. The process of context modeling should follow the following steps: *Data pre-processing*, *aggregation*, and *modeling. Data pre-processing* is done by filtering irrelevant or redundant information as relevant information should only be considered as context information. For example, in the smart home

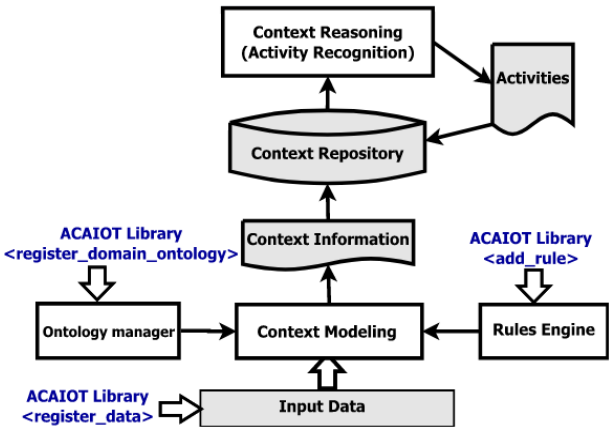

Figure. 2 ACAIOT context management process

domain, filtering the room temperature values should be done such that the values that have a significant change over the previously stored value are only considered. The amount of this change is a parameter instantiated by the developer.

Aggregation is the function of getting high-level context information by aggregating data from one or more data sources. For example, in the smart home case study, we implement an aggregator for generating no motion context information inside the home by ensuring that there is no input from any motion sensor inside the home. *Modeling* is the step of representing the context information in the form of domain-oriented ontology. So, we can get information about the location of an entity, his current activity, and the current temperature at a specific date and time.

5.2 Ontology manager

The primary role of the *Ontology Manager* is combining the domain ontology concepts and rules to the generic ontology (*ACAIOT_Ontology*). Each domain concept is a sub-class of one of the *ACAIOT_Ontology* concepts. Accordingly, the *domain-oriented ACAIOT Ontology* is ready to be used by the other ACAIOT components.

The main concepts of *ACAIOT_Ontology*, as illustrated in Fig. 3, are *Entity, Context, Activity, Location, Date, and Time* concepts. An *Entity* represents a person, application, or object that is considered relevant to the interaction between a user and an application. *Context* represents any information that can be used to characterize the situation of an entity such as *Location*, *Date*, *Time*, and *Temperature*. The date and time are useful in the inference phase to get a conclusion about the entity's behavior pattern. *The location* of an entity is vital information while recognizing the current entity activity. *The activity* represents the actions

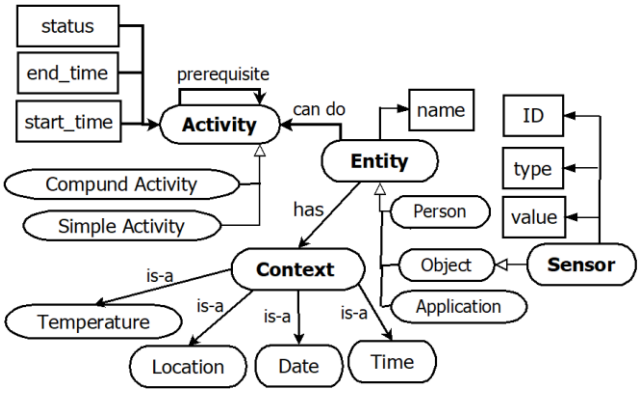

Figure. 3 ACAIOT_Ontology

performed by an entity based on its context. Activity is classified into simple activity or compound activity. A *Simple_activity* instance is generated as a result of a simple event occur by changing a group of sensors' values. For example, when the door sensor is changed from close to open, then an *Open_door*_activity is generated. A compound activity is recognized by the occurrence of a sequence of activities. For example, a compound activity such as *Leave_home* is recognized by a sequence of leave_home_begin and leave_home_end activities after applying a set of constraints illustrated in the activity reasoning rule as described in the *Rule Engine* below.

5.3 Rule engine

The *Rule Engine* is concerning the logic of how to process and manage the context information through ACAIOT context management processing. Such logic is applied over the application data that are organized as instances of the *Domain-oriented ACAIOT Ontology*. There are two sets of rules that are ready to be used by rule engine: Activity rules and service rules. Activity rules are applied to recognize activities done by the target entity. Context reasoning could be applied to recognizing current activity by applying the activity rules. The recognized activity will be added as instances to ACAIOT: *Activity class* or any of its subclasses.

Service rules are used through ACAIOT services templates to customize each service execution according to the application requirements, as shown in Fig. 4 that represents the general structure of service rules. For example, the notification service is triggered when elder people sleep more than their expected sleeping duration by applying the following rule:

Service_Rule_Begin : *service_rule*
Activity
Property
Operator
Value
Service Rule End

Figure. 4 Service Rule structure

Simple_Activity_Rule_Begin: *simpleActivity*
runningActivity,
pair (alarmSensors, alarmValue)
pair (endAlarmSensors, endAlarmValue)
timeCondition(op, value)
Simple_Activity_Rule_End

Figure. 5 Simple Activity Reasoning Rule

Service_Rule_Begin: notify_long_sleep

Sleep, sleep_duration, ">", 10

Service_Rule_End

5.3.1. Simple activity

A simple activity is instantiated when one rule is fire. The general structure of the Activity rule is shown in Fig. 5. The *simpleActivity* is the activity recognized by this rule. *runningActivity* is the prerequisite activity that must be accomplished to fire this rule. Pair *(alarmSensors, alarmValue)* represent the group of sensors that at least one of them must produce the alarmValue to fire this rule. The *Pair (endAlarmSenors, endAlarmValue)* represents another group of sensors that terminate the activity. *TimeCondition* is a time constraint for a mandatory duration with the sensors' values to be considered. The algorithm used to recognize the simple activity is stated in Fig. 6.

For example, recognition of *sleep_begin* activity is fired after the target entity location is on the bed for more than 15 min to be sure that the target entity is start sleeping:

Simple_Activity_Rule_Begin: sleep_Begin

Go_to_Bed, (bed_sensors,"ON"),
(out_bed_sensors, "ON"), tc(">",15)

Simpel_Activity_Rule_End

```
For each sensor event (R)
  For All simpleActivityRule(rule)
    IF (R.sensor belong_to rule.alarmSensors)
       And IF(R.sensor.value = rule.alarmValue)
       And IF(rule.runningActivity.status is active)
    Then
       IF (rule.status  = started)
       Then IF(rule.timeCondition(op,value)) is true
              Then rule.simpleActivity.status is active
              ELSE go_to next event.
       ELSE
              rule.status = started
    Else
       IF (R.sensor belong_to rule.endAlarmSensors)
         And IF(R.sensor.value = rule.endAlarmValue
         And IF(simpleActivity.status is active)
       Then  simpleActivity.status = stop
              rule.status=stoped
```

Figure. 6 Simple activity execution algorithm

```
Compound_Activity_Rule_Begin
    List(activitiy,  time_condition)
Compund_Activity_Rule_End
```

Figure. 7 The compound activity reasoning rule

```
For compound activity (A)
       where A composed of (a1,a2,a3, …an),
              and ai is simple or compound activity,
              ai must be  run before ai+1 is run
       for All ai where i=0 to n
       If each ai is run
              And timecondition is true
       Then A is recognized
```

Figure. 8 Compound activity execution algorithm

5.3.2. Compound activity

The compound activity is instantiated when a chain of activities has occurred sequentially with time conditions between each activity, as shown in Fig. 7. For example, the OutHome activity is composed of LeaveHome and EnterHome activities with time condition greater than 30 minutes. The algorithm used to evaluate the compound activity rule is presented in Fig. 8.

6. ACAIOT services template

In context-aware IoT applications, there are four types of services that usually implemented; *Monitoring*, *Notification*, *Reminder*, and *Prediction*.

```
Service_Template:Monitoring
        Activity
        Property
        outputFunction
End_Service
```

Figure. 9 Monitoring service template

```
Service_Template:Prediction
        Activity
        Property
        predictionAlgorithm
        Service_rule[]
        outputFunction
End  Service
```

Figure. 10 Prediction service template

ACAIOT services templates are proposed to support these functionalities. ACAIOT services templates are adapted by the developer, as shown below, to fit his application requirements. Then, the adapted service is executed based on the context information available in the ACAIOT repository and the adapted service rules.

The first service, *Monitoring* service, is used to get the updated status of the monitored entity. The developer determines the activity he wants to monitor and customize the monitor template defined in Fig. 9 to fit the application requirements. For example, The Developer defines a service for monitoring the sleep start_time property for sleep activity. The developer must implement the function *outputfunction* to determine how the service output will be shown. The second service, *Notification* service, is used to generate an alarm about an emergency case that happened to the target entity. Notification service has a similar template as monitoring, but it has additional service rules to set the emergency condition. For example, notification is needed when the "sleep_duration > 12" and a notification message is sent to the caregiver person for the alert about that case. The developer must implement the *outputfunction* to be used to adapt the notification service output. The third service, *Prediction* service, delivers an expectation of occurrence of a particular activity that may be happened due to the context changes. *The prediction* service template is shown in Fig. 10. It is implemented after defining a set of parameters; the activity wanted to predict its occurrences such as sleep, a specific property of activity such as sleep start_time property, and then a prediction algorithm that can be used for prediction. Here, the developer should select one from the predefined prediction

algorithms (or define his own algorithm). One more parameter is the service rule, it is used to customize the prediction to be adopted to specific situations such as a rule for predicting night sleep only. Finally, a function used to show the prediction service output. The fourth template, *Reminder* service, is delivered to the target entity itself in any form of alarm notification. It reminds him of doing a particular action according to his context and his previous behavior pattern. The reminder service pattern is similar to the prediction service template.

7. Experiments

For confirming the proposed framework's effectiveness, an experiment has been applied to a real dataset. The case study focused on activity recognition in the smart home domain. The primary data sources were smart home sensors that represent the smart home resident context.

7.1 Dataset

Our framework was tested with the Aruba dataset from the Washington State University CASAS smart-home project [23]. The smart home layout was attached to the dataset folder. It includes continuous recordings of home sensors events. Sensor data that was collected in the home is of a volunteer adult. The dataset contains 1719558 records that represent sensor inputs of 220 days (about 7 months). Aruba smart home is equipped with a network of wireless motion and door sensors and houses a single older adult resident who performs regular daily routines.

7.2 Experimental setup

The developer starts using ACAIOT by registering application ontology (in our case a smart home ontology as an extension to ACAIOT_Ontology). Then, data source registration is also done by the developer. Identifying sensors is applied through setting the location and type of each sensor. Finally, the developer customizes activity rules, services rules, and ACAIOT services templates according to the application requirements.

The developer adds activity rules for the sleeping compound activity with a minimum of 30 minutes of sleeping duration constraint. Therefore, sleeping durations of less than 30 minutes are not recognized. Both of sleeping begin, and sleeping end activities are recognized according to the elder person location detected using the motion sensor events.

The developer adds rules for Leave Home compound activity. Leave home recognized if the door is opened and closed, and then followed with no motion detection duration inside the home for at least 10 min.

ACAIOT framework is executed and starts detecting activities that occur in the smart home. The ACAIOT adapted services outputs were evaluated and compared with the manual activity annotation attached with the Aruba dataset, as illustrated below.

7.3 Evaluation metric

Each service has its own evaluation parameters. For example, both Notification and monitoring services are evaluated by using precision, recall, and F1-Score [24], as shown in the following equations:

$$Precision\ (P) = \frac{TP}{(TP+FP)} \quad (1)$$

$$Recall\ (R) = \frac{TP}{(TP+FN)} \quad (2)$$

$$F1 - Score\ (F1) = \frac{2(P\times R)}{(P+R)} \quad (3)$$

Where TP is the number of activities recognized by ACAIOT framework and is included in the dataset annotated activities, FP is the number of activities recognized by ACAIOT, but they are not included in the annotated dataset, and FN is the number of activities in the annotated dataset that are not recognized by ACAIOT.

While for both prediction and reminder services, the performance of these services in terms of accuracy is evaluated by the measure of Mean Absolute Error (MAE). Pattern Sequence-based Forecasting (PSF) algorithm [25] was applied as an embedded prediction algorithm in prediction and reminder service. In addition, the Average prediction (AVE_P) algorithm is implemented. It calculates the average of predicted values from previous days. A threshold is set according to the type of experiment to enhance the results. The output compared with the annotated activities in the dataset as follow:

$$MAE = \sum \frac{|a^{annotated} - a^{predicted}|}{N} \quad (4)$$

Where N is the total count of predicted activities, $a^{annotated}$ is the activity's property values in the dataset annotation, and a ^{predicted} is the activity's property values generated from the prediction algorithm.

Table 2. ACAIOT evaluation

Activity Name	ACAIOT	LSTM	STA	AISAL
Sleep	R: 0.75 P: 0.76 F1: 0.75	P: 0.81 F1: 0.77	R: 1.0 P: 1.0 F1:1.0	
LeaveHome	R: 0.70 P: 0.97 F1: 0.81	P: 0.80 F1: 0.78	R: 0.59 P: 0.83 F1: 0.69	
EnterHome	R:0.83 P:0.94 f1:0.88	P: 0.60 F1: 0.64	R: 0.40 P: 0.64 F1: 0.48	
Average F1-score	0.82	0.73	0.72	0.41

7.4 Results and discussion

We have carried out experiments on Aruba dataset. The experiments applied to support *Sleep*, *EnterHome*, and *LeaveHome* activities. The monitoring service is used for monitoring the recognition of each activity by using the ACAIOT monitoring service. The recognized activities were evaluated by comparing it to the annotated dataset using recall, precision, and F1-score defined in equations 1, 2, and 3, respectively. To evaluate the effectiveness of the proposed framework, it has been compared with a set of recent studies using the same dataset for evaluations. The results are summarized in Table 2 and Fig. 11. Through the first experiment, activity recognition is applied and compared to the approach that uses feature learning using LSTM [26]. LSTM is applied on Aruba dataset for training, and then it tested its model based on another version of Aruba without annotation. It cannot detect the simple activity go-to-Bed because it is not trained on it, while the proposed activity recognition mechanism is better in detecting simple as well as compound activities. Also, ACAIOT is better in EnterHome and LeaveHome activities. Another research used the 10-fold cross-validation for an activity classification method with the tolerance level of the Scanpath Trend Analysis (STA) algorithm [27]. Furthermore, we get a better result than STA in EnterHome and LeaveHome activities. Another approach presents a rule-based architecture for a complete AAL system (AISAL) [28]. It supports notification for an emergency as ACAIOT, but its average F1-score (0.413) is less than ACAIOT.

On the other hand, as shown in Fig. 11 ACAIOT sleep implementation provides lower value than the others. This is because of the time condition that we restrict the recognition rule, which is changeable according to the developer's point of view.

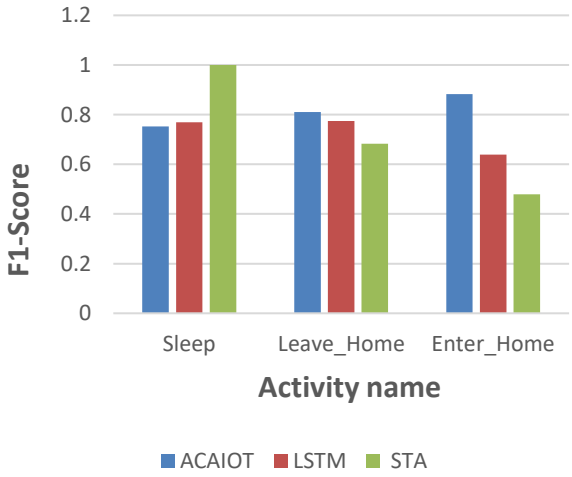

Figure. 11 F1-Score of our ACAIOT activities recognition and comparison with others

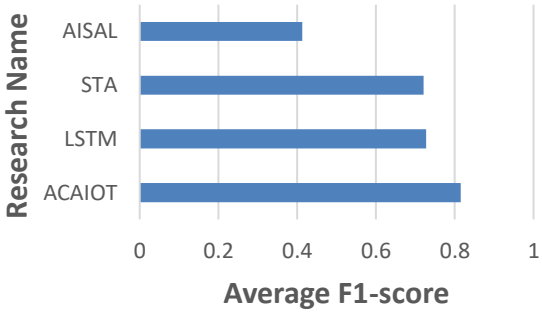

Figure. 12 average F1-Score of our ACAIOT implementation and comparison with related work

Moreover, we find our approach is more robust than the others. It is found that the other approaches can't differentiate between entering and leave home due to their high-frequency occurrence in the dataset. Overall, we achieved F1-score 0.82, which is comparable with the state of the art [29], as shown in Fig. 12.

Table 3. Reminder and prediction services evaluation

Service	Activity	Property	PSF	AVE_P
Prediction	Sleep	Start time	1.060	0.996
Reminder	Sleep	Wakeup time	1.063	0.771

Reminder and Prediction Services implementation are evaluated using MAE defined in equation 4. We applied two different algorithms; PSF and AVE_P, as illustrated in Table 3. Here, we noticed that the AVE_P algorithm has MAE less than the PSF. This is depending on the domain and the pattern of data. In this dataset, we find that the sleeping start time and end time are almost in a similar duration every day, so applying the AVE_P algorithm is better in this case. ACAIOT is flexible enough to run more than on algorithm and then the developer can select the more suitable one to fit the application requirements.

8. Conclusions

In summary, this paper proposes a semantic-based and domain-independent framework ACAIOT that enables adaptive context-awareness management for data acquired by the IoT applications. ACAIOT includes high-level programming constructs and services, which encapsulate application domain constructs and their actions during the implementation phase to raise the level of programming abstraction, enabling developers to implement context-aware IoT applications without worrying about the complexity of low-level connections and sensors data streams. In addition, it enables developers to customize the proposed services to fit different domains and various application requirements. We have evaluated the feasibility of our framework to process real-world sensory data in the context of the daily activity monitoring scenario. It is significant to mention that the ACAIOT framework is generally applicable for different domains as it facilitates the implementation of cloud services without re-thinking of how to handle such data and events and how to manage and deliver the context information. Our future work will focus on testing the proposed approach considering application with several residents with a focus on abnormal cases notification. Also, we will implement ACAIOT on a cloud infrastructure to evaluate its performance on a real implementation.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, ME and AE; methodology, ME and AE; software, ME and AA; validation, ME, AA; formal analysis, ME and AA; investigation, ME, AE and AA; resources, AE; data Curation, ME; writing—original draft and preparation, ME and AE; writing—review editing, ME, AE and AA; visualization, ME; supervision, AE and AA; project administration, AE.

References

[1] P. Rodrigues, YD. Bromberg, L. Réveillère, and D. Négru, “ZigZag: A Middleware for Service Discovery in Future Internet”, In: *Proc. of IFIP International Conference on Distributed Applications and Interoperable Systems*, Berlin, Heidelberg, pp. 208–221, 2012.

[2] E. Borgia, “The Internet of Things vision: Key features, applications and open issues”, *Computer Communication*, Vol. 54, pp. 1–31, 2014.

[3] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, “Context Aware Computing for The Internet of Things: A Survey”, *IEEE Communications Surveys Tutorials*, Vol. 16, No. 1, pp. 414–454, 2014.

[4] P. Pradeep and S. Krishnamoorthy, “The MOM of context-aware systems: A survey”, *Computer Communication*, Vol. 137, pp. 44–69, 2019.

[5] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, “Middleware for Internet of Things: A Survey”, *IEEE Internet of Things Journal*, Vol. 3, No. 1, pp. 70–95, 2016.

[6] X. Li, M. Eckert, J. F. Martinez, and G. Rubio, “Context Aware Middleware Architectures: Survey and Challenges”, *Sensors*, Vol. 15, No. 8, pp. 20570–20607, 2015.

[7] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, “IoT Middleware: A Survey on Issues and Enabling Technologies”, *IEEE Internet of Things Journal*, Vol. 4, No. 1, pp. 1–20, 2017.

[8] U. Alegre, J. C. Augusto, and T. Clark, “Engineering context-aware systems and applications: A survey”, *Journal of Systems and Software*, Vol. 117, pp. 55–83, 2016.

[9] S. A. Chelloug and M. A. El-Zawawy, “Middleware for Internet of Things: Survey and Challenges”, *Intelligent Automation & Soft Computing*, pp. 1–9, 2017.

[10] E. de Matos, R. T. Tiburski , C. R. Moratelli, S. J. Filhoa, L. A. Amaral, G. Ramachandran, B. Krishnamacharif, and F. Hessel, “Context

information sharing for the Internet of Things: A survey", *Computer Networks*, Vol. 166, p. 106988, 2020.

[11] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a Better Understanding of Context and Context-Awareness", In: *Proc. of International symposium on handheld and ubiquitous computing*, Springer, Berlin, Heidelberg, pp. 304–307, 1999.

[12] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques", *Pervasive Mobile Computer*, Vol. 6, No. 2, pp. 161–180, 2010.

[13] C. M. Keet, "An introduction to ontology engineering", *University of Cape Town*, 2018.

[14] O. Cabrera, X. Franch, and J. Marco, "Ontology-based context modeling in service-oriented computing: a systematic mapping", *Data & Knowledge Engineering*, Vol. 110, pp.24-53, 2017.

[15] P. Temdee and R. Prasad, *Context-Aware Communication and Computing: Applications for Smart Environment*, Cham: Springer International Publishing, 2018.

[16] A. Sivieri, L. Mottola, and G. Cugola, "Building Internet of Things software with ELIoT", *Computer Communication*, Vol. 89–90, pp. 141–153, 2016.

[17] A. Colin, E. Gerbert-Gaillard, G. Vega, P. Lalanda, and S. Chollet, "Autonomic Service-Oriented Context for Pervasive Applications", In: *Proc. of 2016 IEEE International Conference on Services Computing (SCC)*, San Francisco, CA, USA, pp. 491–498, 2016.

[18] A. Averian, "A Programming Model for Context-Aware Applications in Digital Ecosystems", In: *Proc. of 17th International Multidisciplinary Scientific GeoConference(SGEM), Bulgaria*, pp.37-44, 2017.

[19] M. R. Hoque, M. H. Kabir, and S. H. Yang, "Development of a Cooperative Middleware to Provide Context-Aware Service in Smart Home", *International Journal of Smart Home*, Vol. 11, No. 5, pp. 33–40, 2017.

[20] P. V. Borges, C. Taconet, S. Chabridon, D. Conan, T. Batista, E. Cavalcante, and C. Batista, "Mastering Interactions with Internet of Things Platforms through the IoTVar Middleware", *Proceedings*, Vol. 31, No. 1, p. 78, 2019.

[21] M. Al-Osta, A. Bali, and A. Gherbi, "Event driven and semantic based approach for data processing on IoT gateway devices", *Journal of Ambient Intelligence and Humanized Computing*, Vol. 10, pp. 4663–4678, 2019.

[22] M. Elkady, A. M. El-Korany, and R. Bahgat, "Towards a Semantic-based Context-as-a-Service For Internet of Things", *International Journal of Computer Science and Information Security*, Vol. 15, No. 8, pp. 284–293, 2017.

[23] D. J. Cook, "Learning Setting-Generalized Activity Models for Smart Spaces", *IEEE Intelligent Systems*, Vol. 2010, No. 99, p. 1, 2010.

[24] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation", In: *Proc. of Australasian joint conference on artificial intelligence*, Berlin, Heidelberg, pp. 1015-1021, 2006.

[25] N. Bokde, G. Asencio-Cortés, F. Martínez-Álvarez, and K. Kulat, "PSF : Introduction to R Package for Pattern Sequence Based Forecasting Algorithm", *The R Journal*, Vol. 9, No. 1, pp. 324-333, 2016.

[26] N. Sarma, S. Chakraborty, and D. S. Banerjee, "Activity Recognition through Feature Learning and Annotations using LSTM", In: *Proc. 2019 11th International Conference on Communication Systems & Networks (COMSNETS)*, Bengaluru, India, pp. 631 - 636, 2019.

[27] H. Y. Yatbaz, S. Eraslan, Y. Yesilada, and E. Ever, "Activity Recognition Using Binary Sensors for Elderly People Living Alone: Scanpath Trend Analysis Approach", *IEEE Sensor Journal*, Vol. 19, No. 17, pp. 7575-7582, 2019.

[28] A. De Paola, P. Ferraro, S. Gaglio, G. L. Re, M. Morana, M. Ortolani, and D. Peri, "An Ambient Intelligence System for Assisted Living", In: *Proc. of 2017 AEIT International Annual Conference*, Cagliari, Italy, pp. 1-6, 2017.

[29] B. Quigley, M. Donnelly, G. Moore, and L. Galway, "A Comparative Analysis of Windowing Approaches in Dense Sensing Environments", *Proceedings*, Vol. 2, No. 19, p. 1245, 2018.