Online-Journals.org

iJOE International Journal of Online and Biomedical Engineering

iJOE | eISSN: 2626-8493 | Vol. 20 No. 6 (2024) | OPEN ACCESS

https://doi.org/10.3991/ijoe.v20i06.46545

PAPER

Context-Aware IoT System Development Approach Based on Meta-Modeling and Reinforcement Learning: A Smart Home Case Study

Amal Hallou[1](✉), Tarik Fissaa[1], Hatim Hafiddi[1], Mahmoud Nassar[2]

[1]InnovatiVE REsearch on Software, Systems and data (EVEREST), National Institute for Postal and Telecommunication Studies (INPT), Rabat, Morocco

[2]IMS (IT architecture and Model Driven Systems development) Team, ENSIAS, Mohammed V University, Rabat, Morocco

hallou.amal@inpt.ac.ma

ABSTRACT

Integrating context awareness into the Internet of Things systems is essential for enhancing their adaptability to their context, particularly their user preferences and behaviors. This paper proposes an approach to model and develop context-aware self-adaptive IoT systems, capable of adapting their actions according to their users' preferences. The approach consists of three main axes. The first axis involves establishing an overview of the system architecture that provides a high-level understanding of the various components of a context-aware IoT system. The second axis concerns the creation of a context-aware IoT systems meta-model, encapsulating the essential elements, relationships, and dependencies governing context awareness within the IoT system in a domain-independent manner. The third axis proposes a reinforcement learning reasoning process to enable intelligent decision-making within context-aware IoT systems. To validate the feasibility of the proposed approach, a simulation was conducted using the OpenAI Gym framework to emulate a context-aware smart home system. The results highlight the feasibility of the approach, and its potential to enhance real-life IoT systems' awareness of their users' context.

KEYWORDS

Internet of Things, context-awareness, reinforcement learning, meta-modeling, smart home

1 INTRODUCTION

The Internet of Things, introduced by Kevin Ashton in 1999, has emerged and evolved in recent years as a paradigm of communication between interconnected devices [1]. It can be considered as a set of connected devices sharing data, without human awareness, to optimize their performance. This technology is based on smart physical objects (sensors, actuators…) that collaborate autonomously, without

Hallou, A., Fissaa, T., Hafiddi, H., Nassar, M. (2024). Context-Aware IoT System Development Approach Based on Meta-Modeling and Reinforcement Learning: A Smart Home Case Study. *International Journal of Online and Biomedical Engineering (iJOE)*, 20(6), pp. 25–42. https://doi.org/10.3991/ijoe.v20i06.46545

Article submitted 2023-11-07. Revision uploaded 2024-01-12. Final acceptance 2024-01-12.

human involvement, to access data and services remotely, and perform actions on behalf of users [2] [3].

Context-aware computing has been identified as an important IoT research prospect and a key enabler for the IoT. The context-awareness feature leverages IoT systems by giving them the ability to capture variations in the state of the environment and adapt accordingly, making them more interesting to their users [4] [5]. Context can be defined as the whole set of 'things' that interact with the physical entities of an IoT system or as any information that characterizes the situation or distinguishes the circumstances of an entity (person, place, or object) and is considered relevant to the interaction between a user and an application [6]. Combining context awareness and IoT allows the design of intelligent systems that can anticipate and fulfill user needs effectively.

In this context, our paper endeavors to introduce a novel approach to design and develop context-aware self-adaptive IoT systems capable of learning the behavior of their users and autonomously adapting their actions. Numerous researches have explored similar objectives, but their approaches are often related to a specific domain or rely on rule-based reasoning or probabilistic models. Those approaches, while valuable in certain contexts, are insufficient for addressing dynamic context changes. In contrast, our aim is to leverage IoT systems to ensure dynamic adaptability to contextual changes in accordance with user preferences and behavior. This dynamic adaptation involves automatic reconfigurations of the state of the devices in response to contextual changes, all of which meet user preferences.

The contribution of this study is a new approach based on Meta-modeling and Reinforcement Learning for the design and development of IoT systems that have adaptive capabilities to meet the requirements of their users. Meta-modeling techniques provide a comprehensive representation of the system, and Reinforcement Learning algorithms enable dynamic decision-making based on contextual changes. Our approach is based on three key axes:

- The first axis concerns an architecture overview that serves as the foundation of the IoT system design. This architecture outlines the key components and their interactions within the system.
- The second axis conceptualizes our vision for context-aware IoT systems using a comprehensive meta-model. By employing the meta-model, we can capture and represent the contextual aspects that influence the behavior and decision-making processes of IoT applications in a domain-independent manner.
- The third axis focuses on the reasoning process based on a Reinforcement Learning algorithm. The latter offers an adaptive and dynamic approach to decision-making, enabling the system to learn and optimize its actions over time. By continuously learning from the environment and leveraging rewards and penalties, the reasoning algorithm can optimize the decision-making process of the system, ensuring that it responds intelligently to varying contextual changes.

In order to validate the feasibility of our proposed approach, we conducted a smart home system simulation using OpenAI Gym [7], to demonstrate the practical implications of our methodology.

The remainder of this paper is organized as follows. Section 2 provides an overview of the relevant literature concerning context-aware IoT architectures and meta-models. In Section 3, the proposed approach is detailed, starting with an illustrative scenario and the three steps of the approach. Section 4 is dedicated to a smart home case study and its simulation using OpenAI Gym. The concluding section presents conclusions and considerations for future work.

2 RELATED WORK

Context-aware IoT systems have been well explored in the literature, with various studies proposing approaches, middleware, and frameworks to ensure the development of context-aware IoT systems.

Some of the proposed research has adopted rule-based reasoning techniques. For example: [8] introduced a control and monitoring unit for the control of an irrigation system using a classical rule-based technique of artificial intelligence to generate a set of relevant information used by the system to provide the appropriate services. [9] proposed an architecture employing joint optimization of scheduling and control, integrating ontology and reasoning technologies for context-aware scheduling and control in dynamic manufacturing environments. [10] presented a context modelling approach for IoT providing relevant information at the right time. The approach is based on a context ontology, which provides a formal context representation, relevant information retrieval, and knowledge sharing.

Alternatively, some studies have focused on probabilistic models. [11] introduced a solution based on federated learning, utilizing mobile devices to actively learn user preferences and adapt to various environments. This approach integrates a global model with federation-generated knowledge and a personalized model that caters to individual user needs, enabling fast personalization, deployment, and predictions in diverse environments.

Rule-based reasoning, while effective in certain application, is not sufficient when it comes to dynamic and complex IoT systems. This is due to the static nature of rules and their lack of flexibility to adapt to new user behaviours or contextual scenarios. Additionally, rule-based approaches are hard to generalize and scale to new contexts. On the other hand, probabilistic based models may also fail to be generalized or scaled given the uncertainty nature of probabilities.

In contrast, machine learning offers a more adaptive approach, allowing the systems to learn from previous experiences. Numerous researchers have proposed innovative solutions centered on machine learning. For example, [12] employed a rule-based machine learning method to discover behavioral rules of individual smartphone users in order to provide context-aware intelligent services. [13] developed a context-aware automated system, capable of collecting sensory data and predict elderly motion activity. The reasoning is based on signal processing, support vector machine learning approach, and cloud-assisted smart home architecture for context-aware healthcare services for the elderly living alone at home. [14] designed a behavioral decision tree machine learning classification approach to build a context-aware predictive model based on diverse user activities with smartphones. [15] suggested a machine learning-based framework for selecting services to create assistive services for disabled people.

Some machine learning propositions are domain-specific. For instance, [16] proposed an adaptive sensing strategy for a Healthcare Sensing System, integrating human context recognition for context-aware sensing. [17] aimed to train traffic lights agents using a Deep Q-network (DQN).

Despite the various machine learning-based approaches proposed in the literature, we have not encountered any domain-independent approach deploying reinforcement learning to develop self-adaptive IoT system, capable of learning user preferences and dynamically adapting its actions in response to contextual changes.

3 THE PROPOSED APPROACH

In this section, we outline our proposed approach for designing and developing context-aware IoT systems, centering on enhancing them with the capacity to adapt their actions, following any context change, to their users' preferences and needs. In order to explain the aim of our approach, we commence by a motivating scenario that serves as an example to a context-aware self-adaptive system. Subsequently, we present the pivotal axes of our approach: the system's architecture, its meta-model, and the reinforcement learning-based reasoning component.

3.1 Motivating scenario

Mr. and Mrs. John live in an apartment, equipped with different sensors and actuators (temperature sensors, fire sensors, etc.). They are keen to harness the full potential of their home using a context-aware system capable of automating the services provided by these devices, based on their preferences. Their goal is to have a smart home system that adapts to their personalized experience and intelligently controls objects in the home to meet their needs and daily routines.

A context-aware smart apartment should be able to learn the daily routines of its inhabitants and adapt the devices of the apartment accordingly. For example:

- If one of the inhabitants usually watches TV in the living room at 8 pm, it will turn on the TV automatically at that time and adjust the light automatically according to their preferences.
- If one or more inhabitants are in a specific room, the temperature and humidity levels will be adjusted to make the environment more comfortable for the occupants.
- If the occupants leave the room, the HVAC system is turned off to save energy.
- The system automatically adjusts the lights and blinds of all rooms based on the time of day, the level of natural light, and the occupancy of the room.

3.2 Context-aware IoT system architecture

In this section, we provide an overview of the proposed architecture for context-aware self-adaptive IoT systems (Figure 1). The architecture is structured into three tiers, each consisting of two layers. The tiers and their layers are described as follows:

- The first tier is the data acquisition tier, which includes physical and data layers. This tier is responsible for gathering data from multiple sensors, including mobile phones, and storing it in the data repository. It is important to note that our approach is domain-independent, hence the developers have the flexibility to choose communication protocols, governing the communication between devices and data exchange, which are adapted to their needs.

- The second tier is the processing tier, which encompasses the context acquisition and the context reasoning layers. The context acquisition layer is responsible for extracting contextual elements from the data received. This process relies on a context model that specifies the types of contextual information. On the other hand, the context reasoning layer evaluates whether there has been any change in one or more contextual elements, to proceed to the context-aware reinforcement learning-based reasoning component.
- The third tier is the application tier, which comprises adaptation and physical layers. The adaptation layer uses the decisions made during the reasoning process and executes the actions determined accordingly. On the other hand, the physical layer enables the system to perform physical operations based on the decisions made, such as controlling devices, triggering alerts, or sending notifications. This layer facilitates the translation of digital decisions into tangible realm-world actions.

In the rest of the paper, we focus on the context reasoning and adaptation layers, to propose a performant reasoning process that can choose, after any relevant change in contextual information, the appropriate actions to execute to meet the user preferences and behavior.

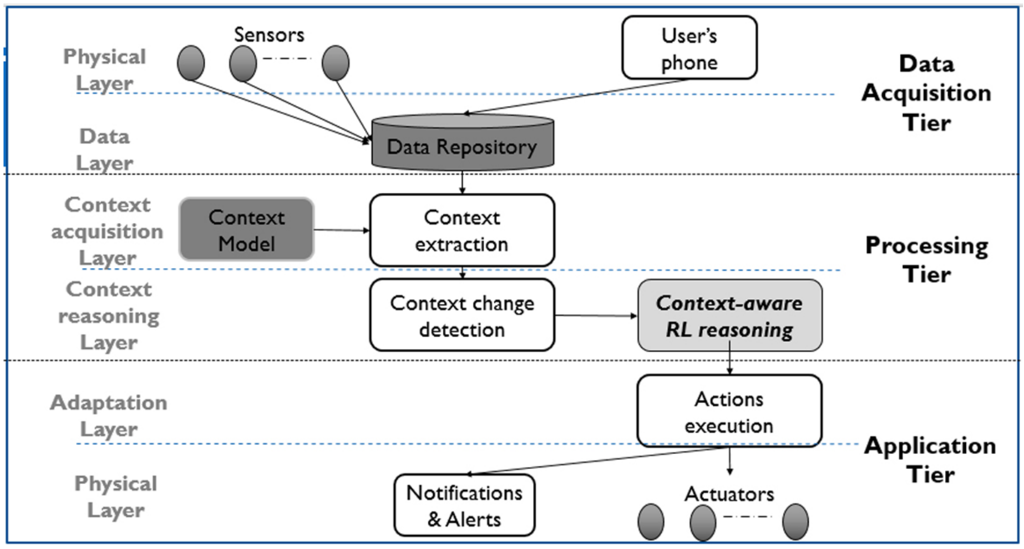


Fig. 1. Proposed context-aware IoT system architecture

3.3 Context-aware IoT system meta-model

In order to give a domain-independent representation of context-aware self-adaptive IoT systems, the UML meta-model, shown in Figure 2, is proposed. This is defined with a high level of abstraction, which makes it easy to reuse.

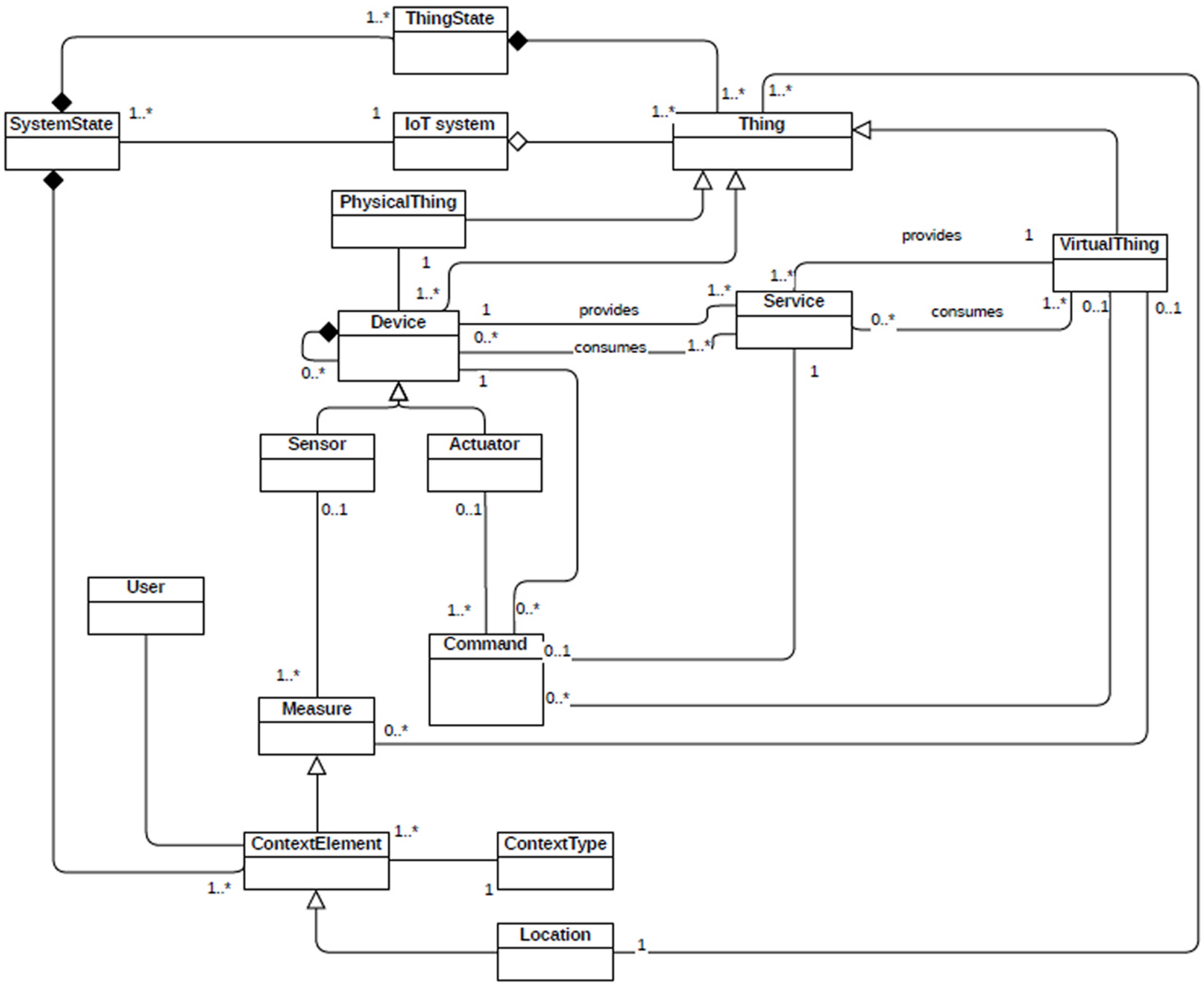


Fig. 2. Proposed context-aware IoT system meta-model

To define this meta-model, we assume the following considerations:

An IoT system comprises a set of things that communicate and exchange data, enabling the system to function as a unified network. In this work, we consider that a thing can be a physical thing, a virtual thing, or a device.

A physical thing can be any object from the real world, to which one or more devices are attached to ensure that the system effectively captures and responds to real-time information and events. A device can be either a sensor, an actuator, or a combination of both. For example, a door is a physical object, to which we can associate an actuator that can automatically open/close it, a sensor that can detect if someone is approaching it… A sensor belongs to a sensor category depending on its purpose, can gather data from the surrounding environment, and delivers various measurements. On the other hand, an actuator also belongs to an actuator category and is designed to execute commands that perform actions affecting the behavior or state of a specific thing.

In this paper, we consider that a virtual thing can be an object that sends or receives data directly to the system without the need for a third device (e.g., mobile phones), or a virtual system that is used to gather data or execute commands.

Services enable communication and interaction between the system's things. Each service is provided and can be consumed by a device or a virtual thing. A command of an actuator is essentially a service call. For example, a temperature sensor can provide a "getTemperature" service, that allows other things to request the current temperature value. Similarly, an actuator attached to a door may consume a "Close" service to automatically close the door.

This system can adapt to the context of the user and the environment in which it operates. The measures sent from a sensor or a virtual thing can be contextual elements of the user environment. Additionally, the location and information of the user are context elements that are used by the system to determine the actions to be made.

The state of the system at a given instant is determined as a combination of the state of its things and the values of predefined contextual elements. The state of a thing refers to its current operational status, such as reading from a sensor, or the availability of an actuator for command. By considering both things and contextual elements, an IoT system can make informed decisions, execute appropriate actions, and offer valuable data to users.

3.4 The context-aware reasoning component

The context-aware reasoning component plays a crucial role in automating IoT systems and minimizing the need for user intervention, by targeting the appropriate actions to execute in response to changes in context elements. An overview of this component is shown in Figure 3.

Overview of the reasoning component. Our reasoning component adopts an approach based on reinforcement learning techniques and, more specifically, a Q-learning algorithm.

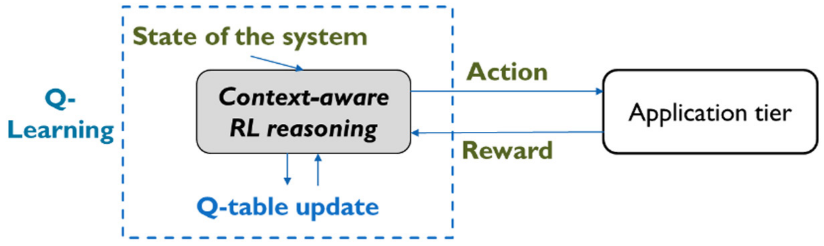


Fig. 3. Overview of the reasoning component

Reinforcement learning (RL) is an adaptive control algorithm that can be used to address decision-making challenges within IoT. This approach is grounded in experience-driven machine learning, enabling a system to autonomously gain insight into its environment without external teaching [18]. Operating through an autonomous agent, RL entails sensing environmental aspects and mapping situations to actions to maximize the reward value, following an explicit goal [19]. An RL agent learns over time by dynamically interacting with its environment, through trial-and-error. It observes its state at each time step, selects an action, transitions to a new state, and assesses the action's effectiveness through a reward [20].

Typically, problems addressed by RL are formulated as Markov Decision Processes (MDPs), which provide a means to dynamically control stochastic systems, guided by the Markovian property where the current state is independent of past states [19].

In this approach, we propose the use of a Q-learning algorithm [19] because it is an off-policy, model-free reinforcement learning, where there is no need to set probabilities for each action. However, the system learns by itself the best actions to take, using the Bellman equation: $v(s) = \mathbb{E}[r_{t+1} + \gamma v(s_{t+1} | s_t = s)]$, where: $\mathbb{E}$ is the expectation.

Problem formalization. Context-aware reasoning can be formulated as a Markov Decision Process (S, A, R, γ, s_0) where:

- ***S*:** The state of the system, which is given as the state of its things and the values of contextual elements.

As modeled in the previous section (Section 3.3), the state of each physical thing is given by the state of the devices related to it, whereas the state of a virtual thing is given by the value of its attributes. In contrast, the context values are provided by some of the sensors of the system.

The state of the system at time epoch t is given by the vector of the vectors in Equation (1):

$$state_t = \begin{bmatrix} \left[device_1, state_{d_1}\right], \ldots, \left[device_n, state_{d_n}\right], \\ \left[virtal\_thing_1, state_{t_1}\right], \ldots, \left[virtual\_thing_m, state_{t_m}\right], \\ \left[contextElement_1, value\right], \ldots, \left[contextElement_v, value\right] \end{bmatrix} \tag{1}$$

where: n is the number of devices, m the number of virtual things, and v the number of contextual elements.

The state of each virtual thing or device is defined by the values of its attributes, given by the vector of Equation (2):

$$state_{t_i} = [(attribute_1, value_1), \ldots, (attribute_j, value_j)] \tag{2}$$

Where j is the number of attributes of the virtual thing/device t_i

- ***A*:** Finite set of vector actions. For each device/virtual thing, several possible actions can be executed on it.

At time epoch t, when a change occurs, the reasoning component chooses the actions to be executed on each device/virtual thing by activating or deactivating it or modifying one or more values of its attributes. An action on a device modifies the state of the related physical device.

Thus, an action at time epoch t can be formulated as shown in Equation (3):

$$action_t = \begin{bmatrix} \left[device_0, action_{d_1}\right], \ldots, \left[device_n, action_{d_n}\right], \\ \left[virtual\_thing_1, action_{vt_1}\right], \ldots, \left[virtual\_thing_m, action_{vt_m}\right] \end{bmatrix} \tag{3}$$

where $action_{d_i}$ is the action to execute on device d_i, and $action_{vt_i}$ is the action to execute on virtual thing $virtual\_thing_i$.

– $\boldsymbol{R}$: immediate reward function $\boldsymbol{R}: \boldsymbol{S} \times \boldsymbol{A} \rightarrow \mathbb{R}$

The objective of our model is to ensure that the system adapts to the user preferences. Thus, the reward must be calculated based on user the feedback.

The reward of a (state, action) pair (state_t, action_t) is calculated one minute after the execution of action a_t as shown in Equation (4):

$$r_t = -\frac{number_{user\_changes_t}}{(number_{auto\_changes_t} + number_{user\_changes_t})} \tag{4}$$

Where:

– $number_{auto\_changes_t}$: the number of things/virtual things whose states have been automatically modified as a result of the execution of the actions a_t.
– $number_{user\_changes_t}$: the number of things/virtual things whose states have been modified by the users one minute after the execution of the actions a_t.
– γ: the discount factor that measures the importance of future reward.
  $\gamma = 0$: only consider the current period
  $\gamma = 1$: makes the agent strive for future rewards
  $\gamma = \text{fraction} \in [0,1]$: balance between current and future rewards
– $\boldsymbol{s_0} \in \boldsymbol{S}$ the initial state of the system

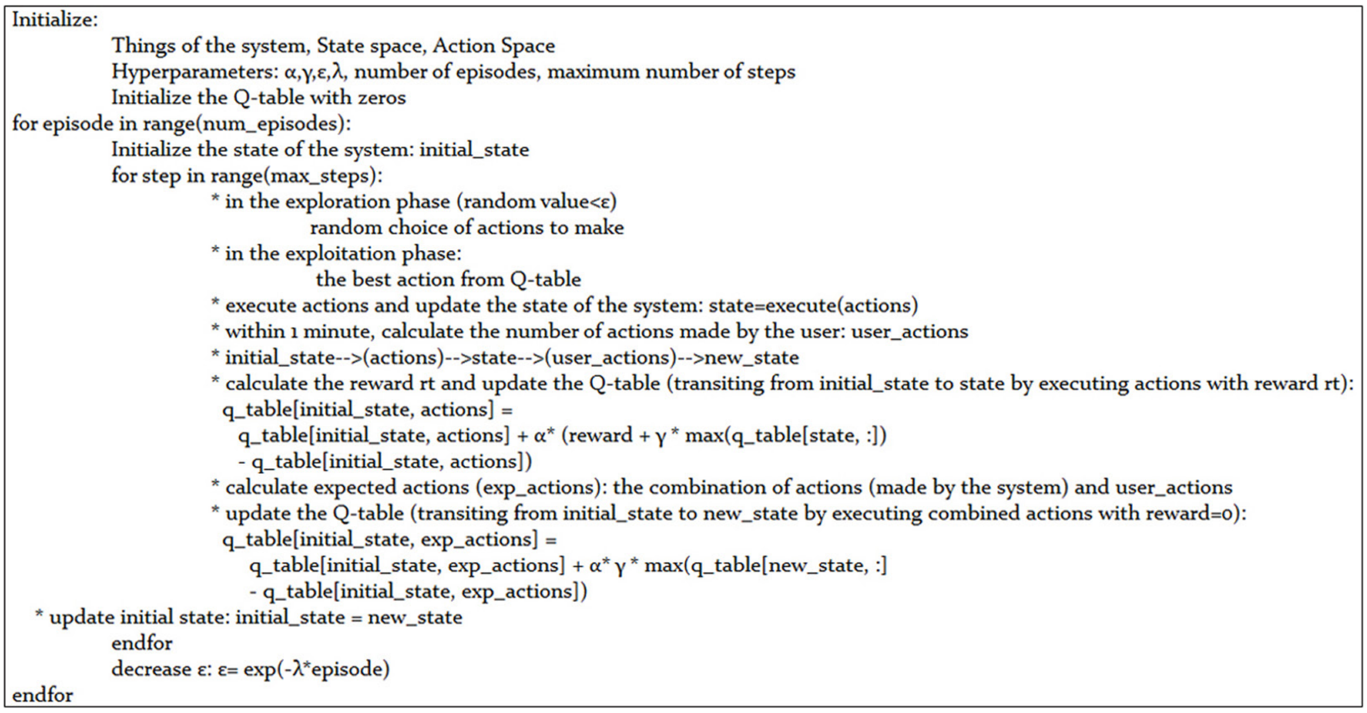
3.5 Q-learning reasoning algorithm

The Q-learning reasoning context-aware algorithm is illustrated in Figure 4.

We begin by defining the hyperparameters that govern the performance of the algorithm and impact the learning process and behavior of the agent. The first two hyperparameters are: (i) the learning rate, α, which indicates how easily the agent should accept new information over previously learned information; and (ii) the discount factor, γ, which indicates how much the agent should take into consideration the rewards it could receive in the future versus its immediate reward. We chose to utilize the epsilon-greedy exploration strategy, which involves selecting the actions with the highest Q-value during the exploitation phase, while selecting random actions during the exploration phase. Thus, two additional hyperparameters are initialized: (iii) exploration probability, ε, the probability that our agent will explore; and (iv) the decay rate for epsilon λ, which indicates the rate of graduate decrease of ε as the agent gains more experience and learns more about the environment. Finally, the number of episodes, and maximum number of steps per episode are initialized.

An episode is a single trajectory of experience from the interaction between the agent and the environment. The state of the system is initialized by restoring the things to their initial state and assigning values to contextual elements. Each episode consists of a sequence of steps. Each step follows a change in the value of one or more contextual elements. To allow the system to efficiently learn the behaviors and preferences of its users, the learning process would take a few hours to a few days or even weeks.

In the training phase, the Q-learning algorithm iteratively updates the Q-values stored in the Q-table to approximate the optimal action values for each state-action pair. The process of calculating the reward is initiated by counting the number of changes introduced by the system, transitioning from the initial state, denoted as s_t, to the subsequent state s_t', within the following 1 minute, the user actions are tracked. If the user performs any action, the resulting state s_{t+1} is tracked. The reward is then calculated using the Equation (4) (Section 3.4), this reward is attributed to the transition from state s_t to state s_t'. Furthermore, the user actions and random system actions are combined to determine the overall impact on the system state transitioning from the initial state s_t to the resulting state s_{t+1}, where the reward is zero, representing the maximum possible reward value. As a result of each step, two rows are added to the Q-table: [initial state, actions] that leads to an intermediate state with the reward calculated using equation (4), and [intermediate state, combination of agent actions and user actions] with reward zero, to consider that this combination of actions is the best to meet the user's preference. We consider an episode to be terminated if the reward calculated by the system is equal to zero.



```
Initialize:
        Things of the system, State space, Action Space
        Hyperparameters: $\alpha$,$\gamma$,$\varepsilon$,$\lambda$, number of episodes, maximum number of steps
        Initialize the Q-table with zeros
for episode in range(num_episodes):
        Initialize the state of the system: initial_state
        for step in range(max_steps):
                * in the exploration phase (random value<$\varepsilon$)
                        random choice of actions to make
                * in the exploitation phase:
                        the best action from Q-table
                * execute actions and update the state of the system: state=execute(actions)
                * within 1 minute, calculate the number of actions made by the user: user_actions
                * initial_state-->(actions)-->state-->(user_actions)-->new_state
                * calculate the reward rt and update the Q-table (transiting from initial_state to state by executing actions with reward rt):
                  q_table[initial_state, actions] =
                    q_table[initial_state, actions] + $\alpha$* (reward + $\gamma$ * max(q_table[state, :])
                    - q_table[initial_state, actions])
                * calculate expected actions (exp_actions): the combination of actions (made by the system) and user_actions
                * update the Q-table (transiting from initial_state to new_state by executing combined actions with reward=0):
                  q_table[initial_state, exp_actions] =
                     q_table[initial_state, exp_actions] + $\alpha$* $\gamma$ * max(q_table[new_state, :]
                     - q_table[initial_state, exp_actions])
   * update initial state: initial_state = new_state
        endfor
        decrease $\varepsilon$: $\varepsilon$= exp(-$\lambda$*episode)
endfor
```

Fig. 4. Context-aware Q-learning algorithm

4 SMART HOME SYSTEM CASE STUDY

4.1 Overview of the system

This case study concerns a compact smart home, equipped with a multitude of sensors and actuators that seamlessly interact with the environment. To make the system automatically adaptive to its users' preferences and behaviors, these devices are strategically distributed to provide precise information about occupants and their activities. The key features integrated into the smart home environment are detailed below, and their spatial distribution through the apartment illustrated in Figure 5. The model of the system conforms to the meta-model introduced in Section 3.2, as shown in Figure 6.

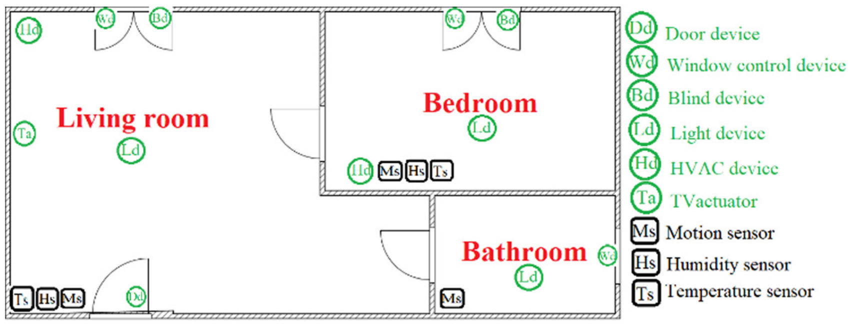


Fig. 5. Distribution of devices in the smart apartment

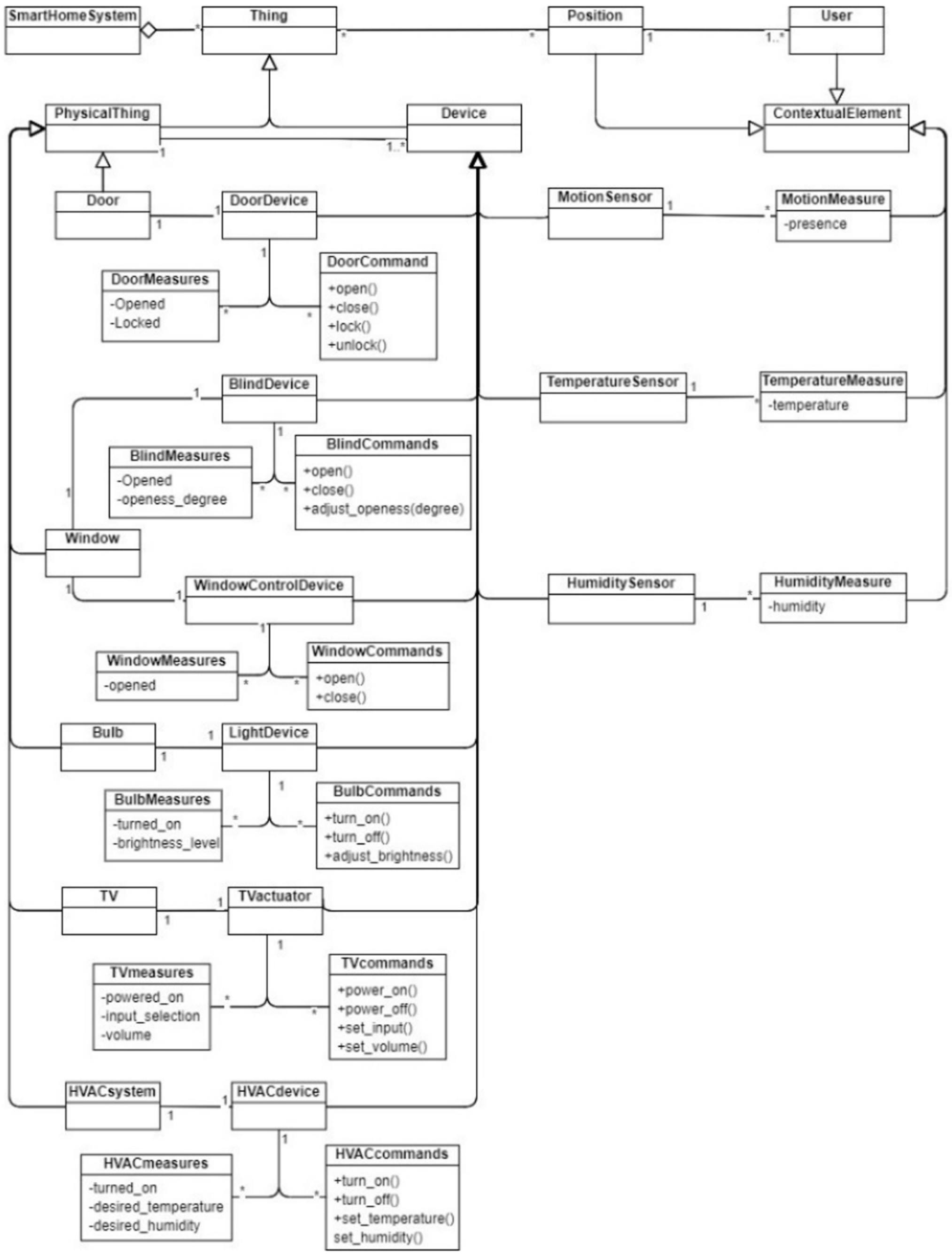


Fig. 6. Model of the smart home system

The smart home system is supposed to be composed of the following physical things related to one or more devices:

- An entrance door, related to a door device, that captures its state (opened/closed, locked/unlocked) and commands its opening/closing and locking/unlocking.
- Two windows, each related to a window control device that captures its state (opened/closed) and commands its opening/closing, and a blind device that captures the state of its blind (opened/closed, degree of openness) and commands the opening/closing and openness degree of its blind.
- A window, related to a window control device;
- Three bulbs, each of which is related to a light device that captures its state (turned on/off, brightness degree) and commands turning it on/off and adjusting its brightness degree.
- Two HVAC systems, each of which is related to an HVAC device that captures its state (turned on/off, desired temperature and humidity) and commands turning it on/off and adjusting the temperature and humidity.
- A TV, which is related to a TV actuator that commands power it on/off, adjusting the volume and input selection.

Additionally, the smart home is equipped with the following sensors that capture the following contextual data: the position of the user, temperature, and humidity inside the house.

- Three motion sensors that accurately detect user presence and movements within their designated spaces;
- Two humidity sensors, that measure moisture levels in the air within their designated spaces;
- Two temperature sensors, that provide accurate and continuous measurements of ambient temperature within their designated spaces.

We assume that the time, date, and user are the other contextual elements of our system.

4.2 Context reasoning simulation

The smart home system was implemented in Google Colab a cloud-based platform that offers a Jupyter notebook environment, and free access to GPU resources, allowing running Python code in the browser using OpenAI Gym, a toolkit for reinforcement learning (RL) research.

OpenAI Gym [7] is a software package that gathers a collection of environments with a unique interface to control them, and is compatible with machine learning libraries such as TensorFlow and PyTorch. OpenAI Gym allows developers to test and benchmark the RL algorithms in the same set of environments.

4.3 Definition of the environment

To implement our smart home case study system, we developed a custom environment, called SmartHomeSystemEnv, that simulates the behavior of various devices and users within a smart home.

Devices implementation. The first step was to define all devices that composes the system. Their implementation was showcased through a collection of Python classes, each representing a specific device type. These classes inherit from an abstract base class, named Device, which defines the fundamental attribute, the position of the device, and the methods that can be common across all the devices (get_position(), get_state() and perform_action()).

The various device types, described earlier (Section 4.1), were then derived from the base class and formed the building blocks of our smart home environment system.

Smart home environment initialization. The basic method for the environment is the init() function, in which its essential components are defined. We begin by declaring the number of physical things in our system, and then instantiate various devices related to those physical things as well as the sensors, capturing their initial states and positions within the smart home.

Next, we define both the observation and action spaces:

- The observation space represents a set of possible environmental states. We define it as a dictionary that encompasses the contextual elements—user, position of the user (detected using the motion sensors), time of the day, temperature and humidity readings in each room—and the states of the physical things, retrieved from the devices related to them using their get_state() functions.
- The action space offers a set of possible actions that can be performed on each device type. We define this as a dictionary grouping of the number of actions possible for each device type.

4.4 Implementation of the Q-learning algorithm

Given that this case study is a primary simulation serving as a proof of feasibility, we used a table, named 'user_actions', containing contextual values—user's action pairs. This table constitutes the preferences of the user. Thus, in every step, the system chooses, using an ε-greedy policy and the best actions to take, then we compare those actions made by the system with the ones that the user can do in a similar context from the user_actions table. The reward is then calculated.

The hyperparameters were chosen by carefully considering their effects on convergence, exploration, and learning efficiency. Table 1 displays the value chosen for each hyperparameter, and provides a brief justification for this specified value.

Table 1. Values of hyperparameters

Value of the Hyperparameter	Rationale of the Choice
The learning rate $\alpha = 0.9$	Allows the agent to adjust quickly the Q-values during the initial training phase
The discount factor $\gamma = 0.5$	Balances preference for immediate and future rewards (since in each step, two rewards are added to the Q-table)
The exploration rate $\varepsilon = 1$	Ensures thorough exploration during early episodes
The decay rate of ε: $\lambda = 0.05$	Ensures that exploration remains relevant for a longer time

5 RESULTS AND DISCUSSION

To evaluate the feasibility and efficiency of the approach, we opted for three performance metrics in the training phase: the cumulative reward per episode, the average rewards per episode, and the maximum steps of an episode. An episode is terminated if the reward calculated by the system is equal to zero, which is the optimal reward.

After many simulations with different numbers of episodes and steps, we found that the reasoning algorithm is efficient when considering 5000 episodes, with 50 steps each and above. This is explained by the high dimensionality of the observation space and the action space, and thus the number of possible states and actions. The results obtained are plotted in Figures 7, 8 and 9, respectively. The figures show that the system is learning since the reward values are improving considerably from the beginning of the 3000th episodes.

As illustrated in Figure 7, the cumulative reward per episode exhibits a noticeable improvement beginning at around 3000 episodes, gradually increasing to zero within the final 1000 episodes. In Figure 8, the average rewards per episode display a fluctuating pattern, and eventually converge to zero near the end of the simulation spanning the last episodes. Figure 9 depicts the gradual decrease of the number of steps per episode starting at around the 3000th episode, and its reduction to just one step by episode within the final episodes. Collectively, these metrics highlight the agent's capacity to effectively balance exploration and exploitation to navigate the smart home environment, thereby demonstrating its progressive learning ability.

In order to provide additional evidence of the efficacy of our methodology, we employed the number of steps per episode as a key metric during the testing phase. As depicted in Figure 10, the 5000 episodes conducted in the testing phase consistently concluded within fewer than 50 steps, none of them exceed 35 steps. This observation signifies that the optimal reward of 0 was attained prior to the completion of each episode. Such a result indicates that the system exhibits commendable learning capabilities, successfully achieving the desired outcome within a relatively short timeframe.

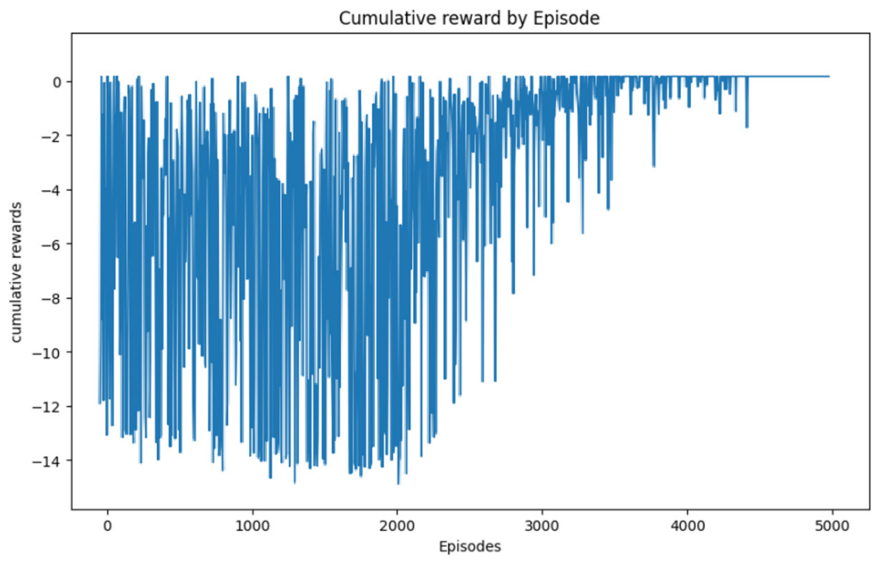


Fig. 7. Cumulative reward per episode

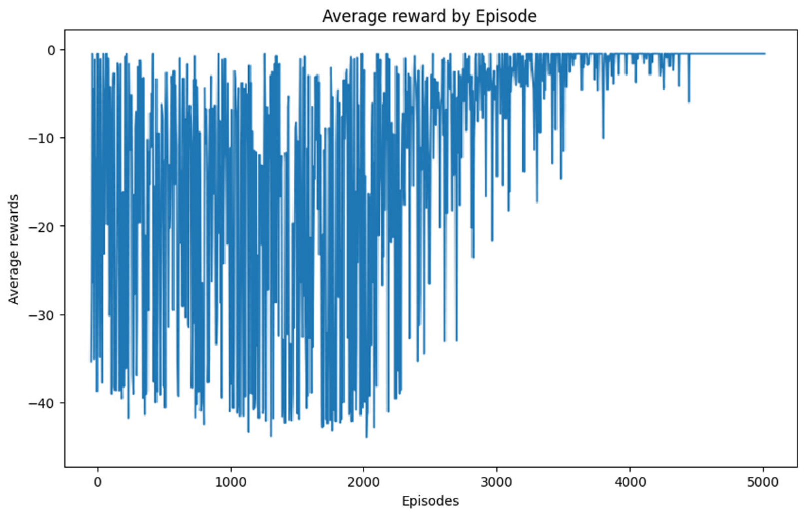


Fig. 8. Cumulative reward per episode

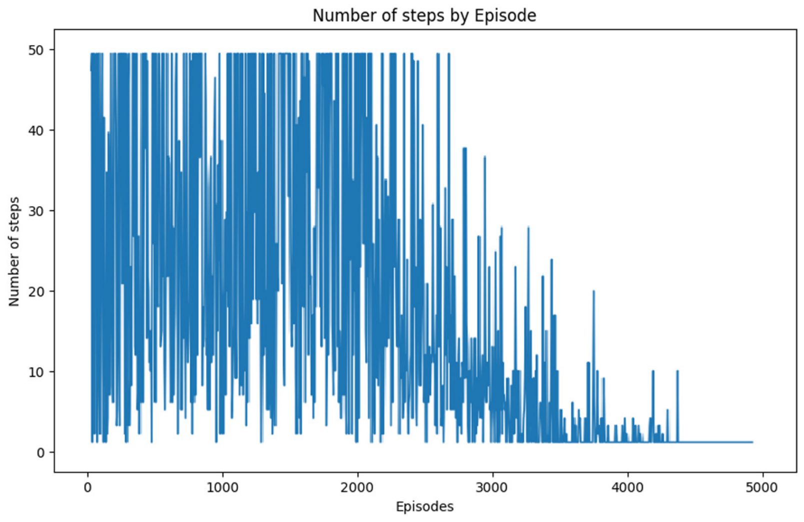


Fig. 9. Number of steps per episode

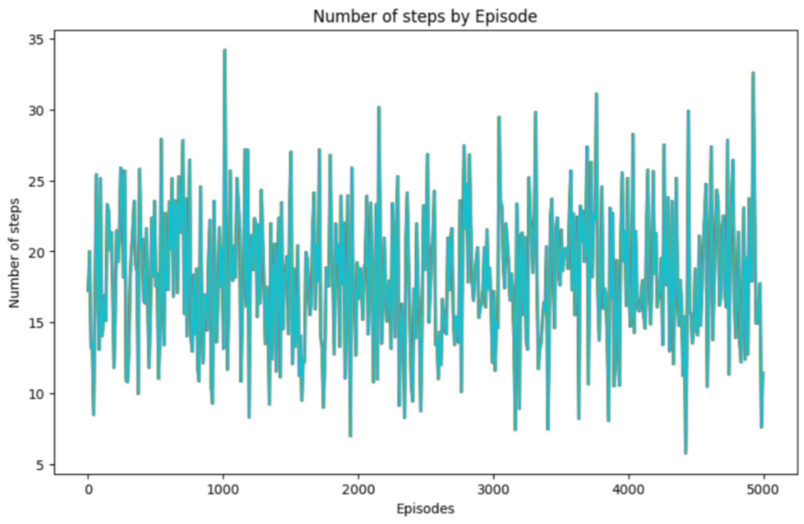


Fig. 10. Number of steps per episode (Testing phase)

These outcomes provide evidence that the algorithm effectively learns the optimal decisions to properly adjust to user actions.

The smart home was obviously small and serves as a basis for further research in this area, which leads to a relatively rapid convergence to the optimal reward. In contrast, in a real-world scenario, the system will need days or weeks to learn the behavior of each of its users.

Although the performance of the approach needs to be validated through a real study case involving real interactions with different users, this small study proves the overall utility of our proposed approach to design and develop context-aware self-adaptive IoT systems.

6 CONCLUSION

In this paper, a novel approach to conceive and develop context-aware self-adaptive IoT systems was introduced. The proposed approach consists of three axes. The first axis involves the design of a context-aware IoT system architecture, building the groundwork for subsequent steps. The second axis consists of a meta-model that captures the different components of context-aware IoT systems was presented. The third axis entails the design and implementation of the context-aware reasoning component, which ensures the system's ability to adapt dynamically to changing contexts and user requirements. The reasoning process is based on Reinforcement Learning; more specifically, the Q-learning algorithm learns user behaviors and preferences iteratively through trial and error. In order to validate the feasibility of our proposed approach, we conducted a case study centered on a context-aware smart home system. The development of the system was simulated using the OpenAI Gym environment. The results demonstrate the feasibility and efficiency of our approach, highlighting its potential applicability to real-world IoT systems.

In future work, we intend to put our simulation-driven insights into practice by engaging a real-world IoT system. The direct interaction between the system and its users further refines the learning and adaptability of the system. In addition, we propose the integration of a pre-RL step, where the Q-table is initialized based on the outputs of existing experiences within the IoT system domain. This can help expedite the convergence of the reasoning process, thereby optimizing the overall performance of the system.

7 REFERENCES

[1] M. Díaz, C. Martín, and B. Rubio, "State-of-the-art, challenges, and open issues in the integration of Internet of Things and cloud computing," *Journal of Network and Computer Applications*, vol. 67, pp. 99–117, 2016. https://doi.org/10.1016/j.jnca.2016.01.010

[2] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012. https://doi.org/10.1016/j.adhoc.2012.02.016

[3] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010. https://doi.org/10.1016/j.comnet.2010.05.010

[4] B. Afzal, S. A. Alvi, G. A. Shah, and W. Mahmood, "Energy efficient context aware traffic scheduling for IoT applications," *Ad Hoc Networks*, vol. 62, pp. 101–115, 2017. https://doi.org/10.1016/j.adhoc.2017.05.001

[5] E. de Matos, L. A. Amaral, R. Tiburski, W. Lunardi, F. Hessel, and S. Marczak, "Context-aware system for information services provision in the Internet of Things," in *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, Luxembourg: IEEE, Sep. 2015, pp. 1–4. https://doi.org/10.1109/ETFA.2015.7301624

[6] K. K. Nithya, M. V. Prathap, and K. R. Remesh Babu, "Cluster oriented sensor selection for context-aware Internet of Things applications," in *International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018*, J. Hemanth, X. Fernando, P. Lafata, and Z. Baig, Eds., in Lecture Notes on Data Engineering and Communications Technologies, Cham: Springer International Publishing, vol. 26, 2019, pp. 981–988. https://doi.org/10.1007/978-3-030-03146-6_113

[7] G. Brockman *et al.*, "OpenAI Gym". arXiv, Jun. 05, 2016. Accessed: Aug. 18, 2023. [Online]. Available: http://arxiv.org/abs/1606.01540

[8] R. Dobrescu, D. Merezeanu, and S. Mocanu, "Context-aware control and monitoring system with IoT and cloud support," *Computers and Electronics in Agriculture*, vol. 160, pp. 91–99, 2019. https://doi.org/10.1016/j.compag.2019.03.005

[9] G. Wan, X. Dong, Q. Dong, Y. He, and P. Zeng, "Context-aware scheduling and control architecture for cyber-physical production systems," *Journal of Manufacturing Systems*, vol. 62, pp. 550–560, 2022. https://doi.org/10.1016/j.jmsy.2022.01.008

[10] P. Pradeep, S. Krishnamoorthy, R. K. Pathinarupothi, and A. V. Vasilakos, "Leveraging context-awareness for Internet of Things ecosystem: Representation, organization, and management of context," *Computer Communications*, vol. 177, pp. 33–50, 2021. https://doi.org/10.1016/j.comcom.2021.06.004

[11] R. Rentero-Trejo, D. Flores-Martín, J. Galán-Jiménez, J. García-Alonso, J. M. Murillo, and J. Berrocal, "Using federated learning to achieve proactive context-aware IoT environments," *JWE*, 2021. https://doi.org/10.13052/jwe1540-9589.2113

[12] I. H. Sarker and A. S. M. Kayes, "ABC-RuleMiner: User behavioral rule-based machine learning method for context-aware intelligent services," *Journal of Network and Computer Applications*, vol. 168, p. 102762, 2020. https://doi.org/10.1016/j.jnca.2020.102762

[13] B. L. Sujaya and R. S. Bhaskar, "A modelling of context-aware elderly healthcare eco-system-(CA-EHS) using signal analysis and machine learning approach," *Wireless Pers Commun*, vol. 119, no. 3, 2021. https://doi.org/10.1007/s11277-021-08341-2

[14] I. H. Sarker, A. Colman, J. Han, A. I. Khan, Y. B. Abushark, and K. Salah, "BehavDT: A behavioral decision tree learning to build user-centric context-aware predictive model," *Mobile Netw Appl*, vol. 25, no. 3, 2020. https://doi.org/10.1007/s11036-019-01443-z

[15] A. Namoun, A. A. Abi Sen, A. Tufail, A. Alshanqiti, W. Nawaz, and O. BenRhouma, "A two-phase machine learning framework for context-aware service selection to empower people with disabilities," *Sensors*, vol. 22, no. 14, 2022. https://doi.org/10.3390/s22145142

[16] L. Wang, S. Xi, Y. Qian, and C. Huang, "A context-aware sensing strategy with deep reinforcement learning for smart healthcare," *Pervasive and Mobile Computing*, vol. 83, p. 101588, 2022. https://doi.org/10.1016/j.pmcj.2022.101588

[17] R. Zhang, A. Ishikawa, W. Wang, B. Striner, and O. Tonguz, "Using reinforcement learning with partial vehicle detection for intelligent traffic signal control," arXiv:1807.01628 [cs], 2020. [Accessed: Nov. 20, 2021]. [Online]. Available: http://arxiv.org/abs/1807.01628.

[18] K.-L. A. Yau, P. Komisarczuk, and P. D. Teal, "Reinforcement learning for context awareness and intelligence in wireless networks: Review, new features and open issues," *Journal of Network and Computer Applications*, vol. 35, no. 1, 2012. https://doi.org/10.1016/j.jnca.2011.08.007

[19] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," p. 352.

[20] L. Buşoniu, R. Babuška, and B. De Schutter, “Multi-agent reinforcement learning: An overview,” in *Innovations in Multi-Agent Systems and Applications* – 1, vol. 310, D. Srinivasan and L. C. Jain, Eds., in Studies in Computational Intelligence, vol. 310, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 183–221. https://doi.org/10.1007/978-3-642-14435-6_7

8 AUTHORS

Amal Hallou, InnovatiVE REsearch on Software, Systems and Data (EVEREST), National Institute for Postal and Telecommunication Studies (INPT), Rabat, Morocco (E-mail: hallou.amal@inpt.ac.ma).

Tarik Fissaa, InnovatiVE REsearch on Software, Systems and Data (EVEREST), National Institute for Postal and Telecommunication Studies (INPT), Rabat, Morocco.

Hatim Hafiddi, InnovatiVE REsearch on Software, Systems and Data (EVEREST), National Institute for Postal and Telecommunication Studies (INPT), Rabat, Morocco.

Mahmoud Nassar, IMS (IT architecture and Model Driven Systems Development) Team, ENSIAS, Mohammed V University, Rabat, Morocco.