EPiC Series in Computing

Volume 82, 2022, Pages 1–10

Proceedings of 37th International Conference on Computers and Their Applications

An Ontology Based Approach towards End User Development of IoT

Narayan C. Debnath[1], Shreya Banerjee[2*], Giau Ung Van[3], Phat Tat Quang[4], and Dai Nguyen Thanh[5]

Department of Software Engineering, Eastern International University, Binh Duong, Vietnam

narayan.debnath@eiu.edu.vn[1], shreya.banerjee@eiu.edu.vn[2]
giau.ung@eiu.edu.vn[3], phat.tat@eiu.edu.vn[4], dai.nguyen@eiu.edu.vn[5]

Abstract

Trigger-Action-Programming (TAP) is a most widely used End User Development (EUD) tool for Internet of Things (IoT). However, end users often cannot differentiate between distinct kinds of triggers and actions. They also make erroneous combinations of those. Consequently, inconsistencies, and bugs are exhibited in behavior of IoT objects. To resolve this issue, end users need to be guided to interpret different triggers, actions and their combinations effectively. In this case, precise representation of temporal and contextual aspects of triggers and actions can assist. Moreover, vast and growing numbers of IoT objects as well as increasing numbers customized rules create scalability issues. To address these drawbacks, this paper has proposed an upper level ontology named as Trigger Action Ontology (TAO) that provides meta rule semantics for TAP. The contribution of proposed ontology specification is to present formal semantics of temporal and contextual aspects of triggers and actions. Further, the ontology is implemented in Protégé. In addition, the expressiveness of the proposed ontology is illustrated using a suitable case study.

Key Words: End User Development, Trigger Action Programming, Ontology, Internet of Things

1 Introduction

The Internet of Things (IoT) offers consumers the opportunity to engage naturally with their surroundings in pervasive contexts. Although IoT objects do most of the work without any human intervention, people can interact with them for example to set them up, to give them instructions or simply to access the data [1]. In this context, End User Development (EUD) paradigm helps consumers customize technology to their individual needs and preferences [2]. In general, end users

* Corresponding Author

are non-programmers. Therefore, they require a simple and easy to use approach to define the behavior of IoT objects [3, 4]. Trigger-Action Programming (TAP) is considered as an effective EUD approach tailoring to this need. It enables users to indicate the behavior of IoT objects through specifying rules.

TAP is a simplified form of the Event Condition Action (ECA), a rule-based approach originally employed to react to different kinds of events occurring in active databases and industrial processes [3]. Rules in TAP have no condition part. These rules represent what should be triggered and what should be the associated effect [5, 6]. Thus, It takes the simple form of conditional statements, like "if <*something happens*>, then <*activate some behavior*>" [7]. IoT devices and Web services can be used both in the trigger (the 'if' part) and in the action (the 'then' part) of the rule [3].

The easy interpretation capability of rule based system helps end users to define different behaviors of IoT objects without programming experience [8]. However, the identification of triggers and related actions sometimes can be complicated towards the end users. In this regard, both commercial (IFTTT [9], Zapier [10]) and research approaches [1, 3, 11] facilitate end-user to specify the creation of trigger-action rules that determine when and how the automations should be performed. Yet, these end user development approaches still faces some challenges.

One crucial challenge is identification between event and state kind of triggers. Event kind of triggers is happened in a specific moment (For Example, "when I leave the room"). On the other hand, state kind of trigger persists for a long period ("as long as it's raining") [12]. Likewise, actions can be distinct kinds based on different timing aspects. Action can be immediate (For example, "sending an e-mail"), can persist for some time and then ended (Extended Action for example, "brewing the coffee") or can persist until other behavior is defined on the same object (Suspended action for example, "Turn on the light") [12]. End users often cannot make proper distinction between different kinds of triggers and actions. Consequently, inconsistencies, redundancies and bugs are exhibited in the behaviors of the IoT objects [13]. To represent differences between distinct kinds of triggers and actions, temporal aspects of those need to be represented precisely.

Another challenge is scalability issues arise from enormous and emergent numbers of IoT devices as well as related personalization rules [5]. This creates problem to set up TAP in practical settings. A high-level abstract representation of rules can solve the issue of scalability to some extent. Since, this kind of representation reduces the core concepts [14] and enables the end users to make less numbers of rules to achieve their expected behavior from IoT objects.

Further, there is a growing demand form the end users to combine several IoT objects to get more complex and combined services. This creates the need to connect multiple triggers and multiple actions. One single trigger or action can be connected with another trigger or action respectively by different ways such as 'And', 'Or', 'Not'. In this context, some triggers can produce contradict settings. For example, rules such as "do not heat and cool at the same time"; "do not turn on both the coffeemaker and the microwave since that will blow a fuse" [8] need to be created to handle contradict settings. Context information related to triggers and actions can prevent end users to create such kinds of contradiction. Context represent surrounding information related to TAP such as "who initiates the trigger", "when the trigger will be initiated", "where the action will be performed". These kinds of information can help to understand end user's mental model, and help them to indicate correct behavior of IoT objects. In general, contextual information can be represented through 5W1H (What, Why, When, Where, Who, How) [7]. In the context of TAP, contextual information for triggers and actions related to temporal, spatial, who is responsible, what is happened and why happened can be achieved through 5W (When, Where, Who, What, Why). Further, "How" context is achieved through combination of distinct kinds of triggers and actions. Therefore, semantics of this context information (5W1H) is also need to represent rigorously. Consequently, proper knowledge can be achieved about outcomes of complex combinations of multiple triggers and multiple actions. This precise semantics further enable effective debugging of the rules created by end users.

Addressing the aforementioned challenges, the proposed work in this paper is aimed to deal with the following research questions. *Q1*. How contextual aspects of triggers and actions can be presented precisely? *Q2*. How TAP rules can be represented in high-level abstract form?

With the objective to deal with these research questions, this paper has proposed an ontology based specification named as Trigger Action Ontology (TAO) for TAP. Ontology is defined as an explicit specification of shared conceptualization. It specifies an abstract view of the world in terms of concepts and their in between relationships [14]. Ontology provides detailed semantics through axioms. Axioms can be represented formally using mathematical logic. The literature recognizes the value of semantic enrichments, through ontologies, for facilitating the event-driven programming of IoT devices also in other domains [15]. An upper level ontology can specify a high-level abstract representation [14]. In this paper, an upper level ontology TAO is specified in first order logic to represent the meta-rule semantics of TAP. This meta-rule semantics also includes the temporal and contextual aspects related to triggers and action. Based on this high-level meta-rule semantics, end users are empowered to interpret different kinds of triggers, actions. They are also enabled to make consistent combinations of multiple triggers and multiple actions. Consequently, end users are able to synthesize efficient TAP. The contribution of the proposed work is to represent formal semantics that can differentiate between distinct kinds of triggers as well as actions. In addition, formal semantics for different combinations of multiple triggers and multiple actions are also specified. This kind of semantics can further help to debug the rules created by the end users. The proposed formal specification is implemented in Protégé [14] tool. Moreover, the expressiveness of the proposed TAO is illustrated using a suitable case study.

2 Related Work

Related approaches existing in literature can be classified in two categories. In [1, 13, 16], authors have specified an ontology named as “EUPont”. The described ontology is high-level semantic model that can be able to adapt to different contextual situation. Authors have specified OWL based description of rules, context and IoT devices. Although, proposed TAO in this paper has similarity with EUPont in representing rules, context and IoT devices layers, yet proposed work has identified more concepts in each layer. EUPont has not considered about temporal aspects of both triggers and actions. Hence, this approach has a limited expressiveness with respect to the end users. Proposed TAO in this paper outperforms EUPont based on representing formal temporal aspects and contextual aspects of TAP. In [15], authors have described an approach that manages and systematize user-defined semantics. However, the authors have not provided semantics related to differences between event and state kinds of triggers, and timing aspects of actions. They also have not considered about precise semantic representation for combinations of multiple triggers or multiple actions. In [5], authors have used Compact Prediction Tree (CPT) and neural network to classify different kinds of triggers and actions. However, predictive models provided by neural networks can be improved through ontology specification. The reason is ontology helps in specifying detailed knowledge related to triggers and actions. Further, the approach has not considered about contextual information of TAP. In [3], authors have considered the fact related to wrong interpretation of event and states by end users. However, to deal with this issue, authors have not provided any formal semantics. They have handled it through user interface through “When” (event) and “While” (state) part and two questions “what happens?” and “in which configuration of states should hold for the rule to trigger?”. However, it can be happened that end users have no idea about “state” word in the second question. Although, authors have conducted usability test, but implicit meaning of event and states is the limitation of their approach. In [7], authors have considered 5W (Who, What, When, Where, Why) composition paradigms to support end users in creating TAP rules. They have also considered five questions

related to 5W. They have managed 5W information through user interface. However, authors have not considered about "How" context, which can give answers how complex behavior of IoT objects can be defined using TAP. In [11], authors have modeled contextual information through user, environment, technology and social interaction. However, they have not provided any formal semantics to prescribe the interpretation of triggers and actions towards end users. Authors in [17] have used Petri Nets to represent temporal aspects of different kinds of events trigger. Yet, they have not considered about temporal aspects of actions. Further, in [18], authors have followed a data flow approach that enables end users to composite multiple triggers and actions. Still, they have not considered about formal semantics of "And", "Or", "Not" connection between multiple triggers.

Most of the existing approaches have not provided precise semantics that can empower end users to interpret the distinction between events and states. In addition, they also have not provided temporal aspects of different kinds of actions. Further, very few have specified about contextual information related to triggers and actions. Yet, those approaches have the drawbacks to provide exact detailed semantics that can be mapped with end users mental model. Besides these, majority of the approaches are not providing a high-level abstract form of the rules. Consequently, tools based on those approaches cannot handle scalability issues effectively. However, the proposed approach in this paper is capable to manage all of these mentioned issues through proposed TAO.

3 Proposed Trigger Action Ontology (TAO)

This paper has proposed an upper level ontology named as Trigger Action Ontology (TAO) to represent meta rules for TAP. The proposed upper level ontology is generic in nature. Consequently, it is high-level abstract description of end user defined rules for different IoT based applications such as Home Automation System, Smart Farming, HealthCare, Smart Parking etc.

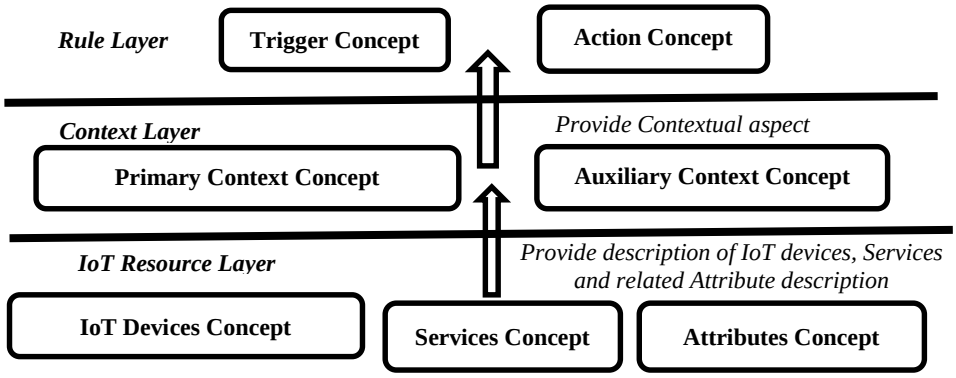

Figure 1: Proposed Trigger Action Ontology (TAO) Model

Proposed TAO is consisting of three layers – *Rules*, *Context* and *IoT Resources*. Figure 1 has demonstrated the conceptual model of proposed TAO. The bottom most layer of proposed TAO (*IoT Resources*) provides ontology-based descriptions for IoT devices, services and related attributes. Based on those descriptions, middle layer (*Context*) of proposed TAO represents the contextual information related to triggers and actions. This contextual information is classified as primary context and auxiliary context. Further, based on the contextual information, the top most layer (*Rules*) provides the precise semantics towards different kinds of triggers, actions, multiple triggers and multiple actions. Formal description of each layers are specified in the following sub sections.

3.1 IoT Resource layer

IoT Resource layer has represented concepts and relationships associated with different types of IoT devices and services in an IoT based domain. Since, TAP represents the behavior of IoT devices and related services; a dedicated layer is needed to represent those concepts. Description of concepts included in this layer is described as follows:

(a) ***IoT Devices***: Distinct domains based on IoT are consisting of different kinds of devices, such as sensors, actuators, tag devices etc. *IoT Devices* concept represents those different kinds of devices. This concept is further categorized as *Sensors*, *Actuators*, *Tag Devices* and *other devices*. Following are the axioms related to *IoT Devices* concept.

F1: $\forall iotde(((Iot\_Devices(iotde)) \wedge (iotde \in Device\_List)) \rightarrow M(GetDevice(Device\_List)))$

F2: $\forall iotde(Iot\_Devices(iotde) \rightarrow$
$(Sensor(iotde) \oplus Actuator(iotde) \oplus Tag\_Device(iotde) \oplus Other\_Device(iotde)))$

Explanation: In *F1*, *Iot_Devices* is a predicate that represents a single instance (*iotde*) of IoT devices. Device*_List* is a set that is consisting of instances of IoT devices such as *iotde*. *M* is a predicate over function *GetDevice()* taking the set *Device_List* as argument and returns an instances of IoT devices such as *iotde*. *F2* specifies that an *IoT* device instance *iotde* can be *Sensor, Actuator, Tag_Device* or *Other_Devices*.

(b) ***Services***: IoT devices can provide and consumes several services such as sensor services (for example, sense temperature value); Actuator services (for example, switch on the light); Tag device services (for example, open the lock of an RFID tag based door). Following are the axioms.

F3: $\forall se((Services(se) \wedge (se \in Service\_List)) \rightarrow M(GetService(Service\_List)))$

F4: $\forall se(Services(se) \rightarrow$
$(SensorServices(se) \oplus ActuatorServices(se) \oplus Tag\_DeviceServices(se)))$

Explanation: *F3* and *F4* specify similar kind of semantics for *Services* as specified in *F1* and *F2*.

(c) ***Attributes****:* This concept represents properties of different IoT devices and Services. Examples of these properties are name of the device, service name, battery capability of the device, service input data, sensor measured value etc. This concept further classified as *IoT Devices Attributes* and *Service Attributes*. The axioms of *Attributes* will be similar kind as specified in *F1* and *F2*.

Besides these three concepts, this layer also includes three relationships – *Provide*, *Consume* and *Has_Attribute*. Following are the descriptions of those relationships.

(d) *Provide*: This relationship represents the fact that one IoT device can produce a service.

F5: $\forall iotde \exists se((Iot\_Devices(iotde) \wedge Services(se)) \rightarrow provide(iotde, se))$

Explanation: In *F5*, *Provide( )* predicate specify the relationship from *iotde* to *se*.

Formal axioms for *Consume* and *Has_Attribute* can be defined in the same way as in *F7*. However, in case of *Consume* relationship, the direction will be from *se* to *iotde*. Further, *Has_Attributes* relationship will be from *IoT Devices* to *Attributes* or from *Services* to *Attributes*.

3.2 Context Layer

Context layer represents the concepts and relationships useful to describe 5W (Where, When, Who, What and Why) information related to triggers and actions. This 5W term is presenting the basic information related to trigger and actions as follows. "Who" represents who is responsible for triggering or performing action. Who can be an IoT device, a service or an end user. "When" represents, the temporal aspects that when trigger or action can be happened. Further, "Where" provides the location information related to the trigger and action. "What" represents that what the trigger and the action specifies. "Why" describes the reason of the trigger and performing the action.

Context layer is consisting of two concepts and their in between relationships. Two concepts are *Primary Context* (*PC*) and *Auxiliary Context (AC*). *PC* is further categorized according to 5W concepts. *AC* provides additional information relevant for *PC*. An example of *AC* can be described in

this way – "An IoT device (representing "who" *PC*) can have *AC* such as brand name, performance criteria etc". Formal axioms of *PC* and *AC* are similar as specified in axioms *F1* and *F2*. Following are the axioms for 5Ws.

F6: $\forall pc \exists c((PrimaryContext(pc) \wedge Why(pc) \wedge cause(c)) \rightarrow M(specifyCause(pc, c)))$

Explanation: In *F6*, *Why()* predicate specify that the context is why context; *cause()* specify that *c* is a cause. Further, *M()* specifies that *specifyCause()* function returns the reasons why triggering is happened or why action is performed based on that trigger.

F7: $\forall pc \exists des((PrimaryContext(pc) \wedge What(pc) \wedge Description(des)) \rightarrow M(provideDescription(pc, des)))$

Explanation: In *F7*, *What()*specifies a one line textual description of the trigger and action.

F8: $\forall pc \exists name((PrimaryContext(pc) \wedge Who(pc) \wedge name(name)) \rightarrow M(provideName(pc, name)))$

Explanation: In *F8*, *Who()* predicate specifies that name of the IoT device, Service, or the user who is responsible for happening of the trigger or action.

F9: $\forall pc \exists time((PrimaryContext(pc) \wedge When(pc) \wedge (TimeStamp(time) \vee (EndTime(time) \wedge StartTime(time))) \rightarrow M(specifyTime(pc, time)))$

Explanation: In *F9*, *When()* specifies that the trigger or the action will be happened either at a time stamp or during a time period .

F10: $\forall pc \exists lc((PrimaryContext(pc) \wedge Where(pc) \wedge location(lc) \rightarrow (M(specifyLocation(pc, lc)))$.

Explanation: In *F10*, *Where()*specify the location, where the trigger or action will be happened.

IoT devices, and *Services* in *IoT Resource* layer can mapped towards *Who* context. *Attribute* can be mapped towards *AC*. Context layer is working as a link between *Rules* and *IoT Resources* layer.

3.3 Rule Layer

Rule layer represents the concepts and relationships specific to the semantics of a TAP rule. In general, a TAP rule is consisting of two parts – trigger and action. Based on this structure, ***Rules*** layer is consisting of following concepts.

(a) ***Trigger* (*T*):** Trigger concept represents the causes, which are responsible for performing actions. Triggers can be further categorized as *Event* and *State*. It can be formally represented as a member of the set *Trigger List*. The axiom related to trigger concept is similar like *F1* and *F2*.

(b) ***Event (E)***: Events represent those triggers, which can occurs at a given time. Further, events are associated with some conditions. For example,"When I have entered room, brighten the room light". Here, the trigger part "When I have entered the room" is representing an event. Since, it occurs at a given moment. The axiom related to Event concept is as follows:

F11:$\forall E \exists t1 \exists c((Event(E) \wedge Time(t1) \wedge Condition(c)) \rightarrow (occurs(E, t1) \wedge associated(E, c)))$

Explanation: In *F11*, *Event()* implies a single event *E*. *Time()* implies a particular time. *Condition()*implies a single condition *c*. Further, *occurs()* represents a single event *E* is happening at a particular time *t1*. Besides this, *associated()* represents that a single event is associated with a particular condition *c*. *Condition* represents the what aspects of a trigger in rule layer.

(c***) State (S)***: State represents those triggers, which tends to persist. For example, "If it is raining, close the window". Here, "if it is raining" representing a state. Since, it tends to persist. The axiom related to State concept is as follows:

F12: $\forall S \exists t1 \exists t2((State(S) \wedge StartTime(t1) \wedge EndTime(t2)) \rightarrow (persiatance(S, t1, t2)))$

Explanation: In *F12*, *State* is a predicate implying a single state *S*. *StartTime()* is a predicate implying the initiation time of the particular state. *EndTime()* is a predicate implying the ending time of the state. *persistance()*implies the relationship between a state and its existence duration.

(d) ***Action***: Action represents some functionalities those are initiated due to triggers. Actions can be further categorized as *Immediate*, *Extended*, and *Suspended*. The axiom of Action is similar as

defined in *F1* and *F2*. Further, *F13* has presented the triggering relationship from *Trigger* to *Action*. This axiom provides the general semantics of a TAP rule.

$F13{:}\forall A \exists T((Action(A) \wedge Trigger(T)) \rightarrow Triggering(T, A))$

(e) ***Immediate Action***: Immediate actions represents the actions which have happened in a particular moment. "Sending an e-mail" is the example of immediate action.

$F14{:}\forall A \exists t1((Action(A) \wedge ImmediateAction(A) \wedge TimeStamp(t1)) \rightarrow happens(A, t1))$

(f) ***Extended Action***: An extended action represents the actions, which are persisted for some duration and then terminated. "Brewing the coffee" is the example of extended action.

$F15{:}\forall A \exists t1 \exists t2((Action(A) \wedge ExtendedAction(A) \wedge StartTime(t1) \wedge EndTime(t2)) \rightarrow persisted(A, t1, t2))$

(g) ***Sustained Action***: a Sustained action represents the actions, which are continued until a new behavior is initiated on the same IoT object. "Close the door" is the example of Sustained action.

$F16{:}\forall A \exists t1 \exists t1 \exists poc \exists T((Action(A) \wedge SustainedAction(A) \wedge StartTime(t1) \wedge EndTime(t2) \wedge trigger(T)) \rightarrow (started(A, t1) \wedge terminated(A, T, t2)))$

Explanation: In *F16*, *terminated()* imply that the activity will be ended when the trigger *T* occurs. Further, *continued()* represents the sustained activity will be continued for the specified time duration.

(h) ***Multiple Triggers*** and ***Multiple Actions***: Multiple triggers and multiple actions connect with each other through relationships *connected trigger* and *connected action***.** These relationships can be further *And Connected*, *Or Connected*, *ExclusiveOr Connected*, and *Not Connected*.

$F17{:}\forall T1 \exists T2((Trigger(T1) \wedge Trigger(T2) \wedge CombinedTriggers(T1, T2)) \rightarrow (And(T1, T2) \vee Or(T1, T2) \vee ExcusiveOr(T1, T2) \vee NotConnected(T1, T2)))$

Explanation: *F17* represent different kinds of connections between multiple triggers. Multiple actions can be represented using similar kind of axioms.

"When "context in context layer provides the temporal information related to triggers and actions. Further, other contexts are also related with both triggers and actions in *Rule* layer through *Has Why*, *Has Who*, *Has What*, and *Has Where* relationships. Moreover, "How" context are represented through combining multiple triggers and multiple actions.

3.4 Implementation of Proposed TAO Using Protégé [14]

The proposed ontology TAO is implemented in an Ontology editor tool Protégé OWL language. Protégé can help to validate the proposed TAO initially. All the concepts of proposed TAO represented as classes in Protégé. Further, all relationships of proposed TAO are represented as object

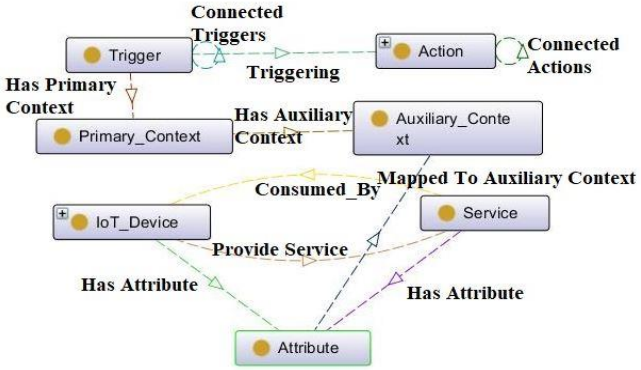

Figure 2: Ontology Graph of proposed TAO obtained through OntoGraf plugin of Protégé **[14]**

properties in Protégé. Figure 2 demonstrates the ontology-based graph of proposed TAO obtained from Protégé. The rectangle nodes in the graph represent classes and the edges represent different

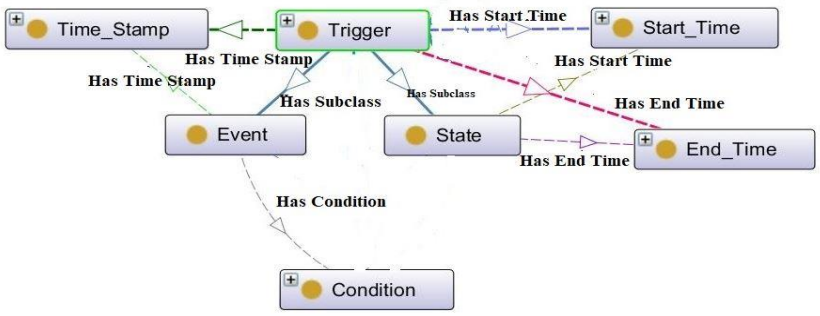

Figure 3: Classification of triggers in proposed TAO

object properties. Further, Figure 3 has demonstrated the classification of triggers. Likewise, we also get ontology-based graph for the classifications of actions.

4 Illustration of proposed TAO using a Case Study

In this section, proposed TAO is illustrated using a fictional case study related to smart home described in [11]. In the smart home domain, users typically personalize and control the appliances through a home controller application. Initially, the sensors and devices in the application are in the form as specified in Table 1. Table 2 has mapped the different concepts of Smart Home application

Part - A		Part -B	
Sensors	Values	Devices	Location
Temperature	19 ° C	TV	Living Room
Balcony Door	Open	Radio	
Users' Stress Level	30	Coffee Machine	
Presence in Living Room	False	Oven	Kitchen
Entrance Door	Close	Light	
Light	52 lx	Fan	Bed Room

Table 1: Initial value of sensors [Part –A] and Positions of Devices [Part –B] in the case study **[11]**

Smart Home concepts specified in the case study [11]		Corresponding *IoT Resource* layer concepts of Proposed TAO	Corresponding *Context Layer* concepts of proposed TAO
Sensor	Light		
	Temperature	Sensor Devices	
	User's Stress level		Name of devices ("Who" Concept)
	Entrance Door	Tag Devices	
Devices	TV		
	Radio	Actuator Devices	
Kitchen			Location of devices ("Where" concept)
BedRoom		Attributes	
Living Room			
Values of sensors and Devices		Attributes	Provide auxiliary context
Sensing room temperature, Open the door, Light is on		Services	Name of services ("Who" Concept)

Table 2: Mapping between Concepts of Smart Home Case Study **[11]**, *IoT Resource* layer and *Context Layer* concepts of proposed TAO**.**

case study [11] to the *IoT Resource* layers and further to *Context layer* of proposed TAO. Entries in the column 1 of Table 2 will be represented as individuals in Protégé. Further, entries in column 2 of Table 2 will represent the types of those individuals. Let assume a rule "*When the user stress level exceeds value 50 and the user is sitting close to the living room TV, then turn off the living room TV and turn on the living room radio*". End users can apply this rule to get the corresponding behavior from IoT objects and Services specified in Table 1. Distinct multiple triggers and multiple actions are recognized. Table 3 has specified this mapping between the example rule and proposed TAO. Thus, Table 2 and Table 3 have exhibited that proposed TAO can provide suitable semantics towards TAP.

Constructs of the Example Rule	Corresponding Proposed TAO Concepts
User' stress level exceeds value 50	Event Trigger
Living room	"Where" aspect
user is siting close to the TV	State Trigger
Turn on the Radio	Sustained Action
Turn off the TV	Sustained Action
User stress level exceeds value 50 and user is sitting close to the TV	Multiple triggers (and Combination)
Turn on the Radio and turn off the TV	Multiple actions (and Combination)

Table 3: Mapping between constructs of Example rule and Proposed TAO concept

5 Conclusion

In this paper, an upper level ontology named as Trigger Action Ontology (TAO) is proposed to specify the meta rule semantics of TAP. TAO is capable to resolve some crucial challenges related to TAP. Those challenges are related to precise and formal representation of (i) differences between state and event trigger; (ii) differences between immediate, extended, and sustained actions; (iii) multiple triggers and multiple actions and further to manage scalability issues. The contribution of the proposed work is to provide formal semantics of temporal and contextual aspects of triggers and actions. The proposed semantics are domain independent and thus it provides a high-level abstract form. Consequently, proposed TAO manages the scalability issues to some extent. Future work includes prescribing an automated framework that can facilitate end users to define correct behaviors of IoT objects. Proof the usability of that framework will be another prime consideration.

Acknowledgements

This research is financially supported by Eastern International University, Binh Duong Province, Vietnam.

References

[1] F. Corno, L. D. Russis and A. M. Roffarell, "A high-level semantic approach to end-user development in the Internet of Things," *International Journal of Human-Computer Studies,* vol. 125, pp. 41-54, 2019.

[2] B. A. Chages, D. F. Redmiles and C. S. Souza, "End-user development for the Internet of Things

OR How can a (smart) light bulb be so complicated?," in *In 2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Raleigh, NC, USA, 2017.

[3] G. Desolda, F. Greco, F. Guarnieri, N. Mariz and M. Zancanaro, "SENSATION: An Authoring Tool to Support Event–State Paradigm in End-User Development," in *Human-Computer Interaction – INTERACT 2021. INTERACT 2021, Lecture Notes in Computer Science*, 2021.

[4] R. Zeng, A. Bandi and A. Fellah, "Designing a Brain Computer Interface Using EMOTIV Headset and Programming Languages," in *In Proceedings of 2018 Second Int. Conf. on Computing Methodologies and Communication (ICCMC)*, 2018.

[5] A. Mattioli and F. Paternò, "Recommendations for creating trigger-action rules in a block-based environment," *Behaviour & Information Technology,* vol. 40, no. 10, pp. 1024-1034, 2021.

[6] A. Bandi and A. Fellah, "Design issues for converting websites to mobile sites and apps: A case study," in *In Proceedings of 2017 Int. Conf. on Computing Methodologies and Communication (ICCMC)*, 2017.

[7] G. Desolda, C. Ardito and M. Matera, "End-User Development for the Internet of Things: EFESTO and the 5W Composition Paradigm," in *Rapid Mashup Development Tools. RMC 2016. Communications in Computer and Information Science*, Lugano, Switzerland, 2017.

[8] S. P. Reiss, "IoT End User Programming Models," in *2019 IEEE/ACM 1st International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT)*, Montreal, QC, Canada, 2019.

[9] "IFTTT," [Online]. Available: https://ifttt.com/home. [Accessed 8 February 2022].

[10] "Zapier | The easiest way to automate your work," [Online]. Available: https://zapier.com/. [Accessed 7 February 2022].

[11] G. Ghiani, M. Manca, F. Paternò and C. Santoro, "Personalization of context-dependent applications through trigger-action rules," *ACM Transactions on Computer-Human Interaction (TOCHI),* vol. 24, no. 2, pp. 1-33, 2017.

[12] J. Huang and M. Cakmak, "Supporting mental model accuracy in trigger-action programming," in *In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, Osaka Japan, 2015.

[13] F. Corno, L. D. Russis and A. M. Roffarello, "A Semantic Web Approach to Simplifying Trigger-Action Programming in the IoT,," *Computer,* vol. 50, no. 11, pp. 18-24, 2017.

[14] M. Horridge, "A Practical Guide to Building OWL Ontologies Using Protégé 4 and COODETools. Edition 1.3," The University of Manchester, 2011.

[15] C. Ardito, G. Desolda, R. Lanzilotti, A. Malizia, M. Matera, P. Buono and A. Piccinno, "User-defined semantics for the design of IoT systems enabling smart interactive," *Personal and Ubiquitous Computing,* vol. 24, no. 6, pp. 781-796, 2020.

[16] F. Corno, L. D. Russis and A. Monge Roffarello, "RecRules: Recommending IF-THEN Rules for End-User Development," *ACM Transactions on Intelligent Systems and Technology,* vol. 10, no. 5, pp. 1-27, 2019.

[17] F. Corno, L. D. Russis and A. Monge Roffarello, "Empowering end users in debugging trigger-action rules," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, Glasgow Scotland Uk, 2019.

[18] S. Eun, J. Jung, Y. S. Yun, S. S. So, J. Heo and H. Min, "An end user development platform based on dataflow approach for IoT devices," *Journal of Intelligent & Fuzzy Systems,* vol. 35, no. 6, pp. 6125-6131, 2018.