IPSJ SIG Technical Report

Vol.2020-MBL-96 No.11
Vol.2020-UBI-67 No.11
Vol.2020-CDS-28 No.11
Vol.2020-ASD-19 No.11
2020/9/29

Towards Automated Generation of Data Models A Case Study of ECHONET Device Objects Specification

VAN CU PHAM[1,a) THANH TUNG LE[1,b) TIEN HUY NGUYEN[2,c) YASUO TAN[1,d)

Abstract: In accordance with the development of cloud computing, the term Web APIs is becoming the backbone of the Internet of Things. The publication of data models is extremely important to enhance the interoperability of Web APIs providers and Web APIs consumers. Currently, data models of a protocol are created by experts, however, humans might cause mistakes such as syntax errors, typographical errors, and inconsistencies of abbreviations. This paper introduces an idea of the automated generation of data models by software(machine) as it can handle the drawbacks of humans and it is much faster than humans. The biggest barrier of using the machine is that it lacks domain knowledge to export terms and abbreviations from protocol descriptions in natural languages. To this end, Natural Language Processing models are utilized. In this research, a case study of the ECHONET Lite protocol is introduced. The results proved that machines can mimic experts in exporting terms and abbreviations while assures the syntactic error-free and consistency of generated data models. Furthermore, the machine supports fast, reliable while enhances reusability in exporting data models for multiple platforms.

Keywords: ECHONET Lite Web APIs, Data Model, Machine Generated Data Model

1. Introduction

WebAPIs[1],[2] have been utilized as the backbone of the World Wide Web, cloud infrastructure, mobile applications, and recently the Internet of Things (IoT). WebAPIs allow the encapsulation of IoT device resources as APIs and being invoked through the network via standard Web protocols[3]. In order to avoid confusion and improve the interoperability of the API providers and API consumers, a data model that provides a standard format to document WebAPIs is essential.

Commonly, data models are proposing and creating by humans with expert knowledge. However, it is a time-consuming task and humans might make mistakes such as syntax errors, typographical errors, and inconsistencies of definitions. The automatic generation of data models by machines could handle the drawbacks of humans. Nevertheless, the biggest barrier of using a machine is that it is required the expert knowledge to generate definitions of properties of a data model. In **Table 1**, an example of device object definition from the ECHONET consortium[*1] is described. Based on the definition, a data model that describes device name *commercialAirconditionerIndoorUnit* with a property name *automaticOperationModeStatus* was created by the ECHONET consortium in the latest release (Device Description(JSON)v1.1.1).

Table 1: A Sample of ECHONET Device Objects Specification

Package-type commercial air conditioner (indoor unit) (except those for facilities)			
Property name	EPC	Contents of property / Value range (decimal notation)	...
...	...	...	...
Current function (”automatic” operation mode)	0xAE	This property indicates, when the air conditioner is opening in the ”automatic” operation mode, the function (”cooling”, ’heating”, ”dehumidification”, ”air circulation” , or ”other”) that is currently being used.	...
		The following values shall be used: Cooling: 0x42 Heating:0x43 Dehumidification: 0x44 Air circulation: 0x45 Other: 0x40	...
...	...	...	...

By utilizing the rule-based approach without human expertise, the machine could generate a data model with (i) device name: *packageTypeCommercialAirconditionerIndoorUnitExceptThoseForFacilities*, and (ii) property name: *currentFunctionAutomaticOperationMode*. Obviously, definitions created by humans are **clearer** and also **shorter** which are essential to provide data models for providing Web APIs.

Recently, machine learning (ML) evolution has achieved breakthroughs in several domains that included natural language processing (NLP)[4]. Therefore, the utilization of NLP techniques is expected to overcome the drawbacks of the machine-based approach. This research introduces a solution to generate data models from the ECHONET device objects[5] automatically to reduce human efforts to support WebAPIs for the ECHONET specification. The main contributions of this work include the

[1] Japan Advanced Institute of Science and Technology, Nomi, Ishikawa 923–1211, Japan
[2] University of Science Ho Chi Minh, Ho Chi Minh City 700-0000, Vietnam
a) cupham@jaist.ac.jp
b) lttung@jaist.ac.jp
c) ntienhuy@fit.hcmus.edu.vn
d) ytan@jaist.ac.jp
*1 https://echonet.jp/

Vol.2020-MBL-96 No.11
Vol.2020-UBI-67 No.11
Vol.2020-CDS-28 No.11
Vol.2020-ASD-19 No.11
2020/9/29

following:

- Propose and implement an NLP solution mimicking humans in generating data model definitions from descriptions. The goal of the NLP solution is to eliminate unimportant words from descriptions in order to create short and meaningful terms and abbreviations from descriptions of ECHONET device objects.
- Propose and implement a solution to syntactically serializes ECHONET device objects into JSON based data models that are compliant with ECHONET Lite Web API*2 and FIWARE smart data models*3.

The rest of this paper is organized as follows. In Section **2**, concepts related to the ECHONET device objects and current effort in supporting data models for the ECHONET device objects specification are briefly introduced. Section **3** highlights the overall concept and main building blocks of the proposed solution. Section **4** and Section **5** describes the building blocks of the proposed solution in greater details. Moreover, the design, implementation, and evaluation are also described. Finally, our work is summarized in Section **3**.

2. Related Work

2.1 ECHONET Device Objects

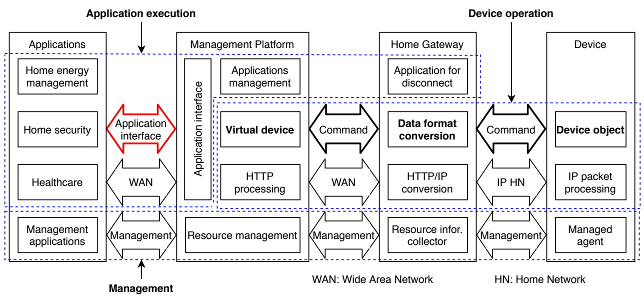


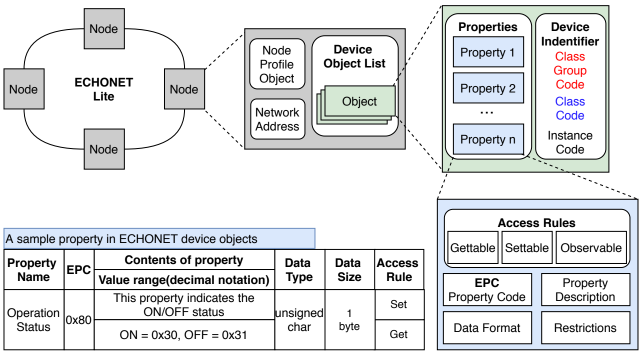
Fig. 1: The Concept of ECHONET Device Objects

ECHONET device objects are a part of the ECHONET Lite[6], a leading protocol for smart homes in Japan[7]. As illustrated in **Fig. 1**, a device object represents a logical device that is classified into seven groups and 117 classes of devices in the latest English specification released in 2020 (Release M). Device objects offer a standardized method to represent device resources and services via a list of *Property* and constraints for each property. In [8], the author proposed an idea of providing a JSON version of the ECHONET device objects and the JSON schema that supports the latest version of the device object is available at the *HEMS Interoperability Test Center* homepage*4.

2.2 Data Models for ECHONET Device Objects

As in **Fig. 2**, data models could be supported at *Device*, *Home Gateway*, and *Management Platform* layer. The ECHONET device objects specification in Section **2.1** support the data model of the *Device* layer.

In [10], an ontology-based data model was introduced at the



Property Name	EPC	Contents of property	Data Type	Data Size	Access Rule
		Value range(decimal notation)			
Operation Status	0x80	This property indicates the ON/OFF status	unsigned char	1 byte	Set
		ON = 0x30, OFF = 0x31			Get

Fig. 2: ITU-T Y.2070 Functional Architecture for Home Energy Management System [9]

Home Gateway layer. The proposed data model supports the ECHONET device objects specification (release K) and ambient assisted living platform[11]. However, the data model was syntactically generated by the software and the naming conventions, as well as abbreviation definitions, have been done manually by humans.

Since data models and descriptions are extremely important to support WebAPI[12], the developments of data models at the *Management Platform* layer (in Fig. 2) are emerging. For example, FIWARE[13] provides several JSON based smart data models for smart city infrastructure and the developments of new data models for other "smart" infrastructures are actively promoting. The oneM2M ecosystem released ontology-based semantic data model[14], and also smart device templates (syntactic). However, the support of ECHONET device objects specification has not been introduced in those platforms.

In [15], a data model, namely *JSON Device Description*, was created by the *ECHONET Lite Web API Working Group*. The total of 43/117 JSON schemes describe ECHONET device objects specification, however, only 27 out of 43 schemes are valid where the rests need to be revised and 4 of them require major reworking. As it was created by humans, common mistakes of this data model include: spell mistakes, syntactic mistakes (missing fields, misplaced fields, incorrect use of quotation marks), incorrect descriptions of data type. In a report, it took approximately 3 hours for an experienced person to create JSON schemes for 6 sensor objects with only 9 properties. It is such a time-consuming task to create data models for the ECHONET device objects specification manually.

3. Automated Generation of Data Models: The Concept

The overall concept of the proposed solution, namely *eModelGen* is illustrated as in **Fig. 3**. The input of the whole process is the specification document (ECHONET device objects specification). Fortunately, the JSON version of the specification is available as an open document so the process to digitalize the input data is unnecessary. The structure of provided input document is defined as followings:

- *metadata* indicates overall information of the documents such as version information, release number, etc.
- A list of supported data type definitions.

*2 https://echonet.jp/web_api/
*3 https://github.com/smart-data-models/data-models
*4 http://sh-center.org/files/156

Vol.2020-MBL-96 No.11
Vol.2020-UBI-67 No.11
Vol.2020-CDS-28 No.11
Vol.2020-ASD-19 No.11
2020/9/29

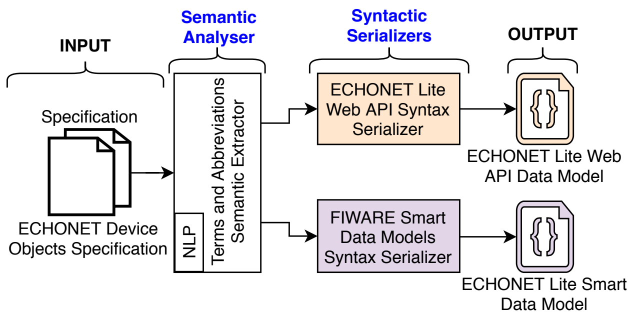

INPUT
Semantic Analyser
Syntactic Serializers
OUTPUT
Specification
ECHONET Device Objects Specification
NLP
Terms and Abbreviations Semantic Extractor
ECHONET Lite Web API Syntax Serializer
FIWARE Smart Data Models Syntax Serializer
ECHONET Lite Web API Data Model
ECHONET Lite Smart Data Model

Fig. 3: The Concept of Automated Generation of Data Models for ECHONET Device Objects Specification

- A list of device objects. A device object provides (i) the object code (EOJ), (ii) name of the object in English and Japanese, and (iii) a list of properties supported by the object.
- A property provides (i) the property code (EPC), (ii) name of the property in English and Japanese, (iii) access rules of the property, and (iv) data type of possible values of the property.

Since the document is created to support developers to interact with hardware devices, the *object code* and *property code* are acceptable. However, for the Web APIs development, the *object name* and *property name* are fundamental. Therefore, a process to extract semantics of the *object name* and *property name* descriptions to create **terms and abbreviations** is inevitable. Currently, this process is manually achieved by experts and the *eModelGen* has a *Semantic Analyser* (in Section. **4**) to mimic this process.

Even though the semantic abbreviations are the same for different models to enhance the consistency and reusability, each data model has different syntactic formats as well as data type definitions. Therefore, for a target data model, a *Syntactic Serializer* (in Section. **5**) is required.

4. Semantic Analyser

In the development of this building block, we realize that a concise representation of object and property names as abbreviations can reduce time and effort to create data models. It comes from a ton of redundancy from unnecessary information in the description string. In recent systems, a formation of property and object names is carried out by humans and it is too difficult and burdensome to deploy in the huge of samples in the practical environments. Therefore, it motivates and encourages us to build an automatic system to extract short and meaningful abbreviations from the long descriptions of the specification. Based on the requirement of this area, we prefer concise patterns to the long ones. It means that the shorter the extracted phrases are, the better results we get. In other sense, the representation needs to maintain the semantic of these original descriptions.

To build the NLP model, features of sentences and words (from description sentences) are extracted by Key-phrase Feature Extraction in Section 4.1. Then, we apply our Description Extraction algorithm to obtain the critical components from the original input in Section 4.2. Next, to prove the effectiveness of our model, the result of our model as well as the comparison with the manual samples are represented thoroughly in Section 4.3. Besides, we also present some samples and discussion in Section 4.3.4.

4.1 Key-phrase Feature Extraction

The most important task in the key-phrase extraction is mapping a sentence into a vector space. However, the representation must reflect the meaning of the original input. A sentence of natural language is the interaction among words. Each word in a sentence has its own meaning and connection with the others. Recently, in most of NLP works, BERT[16] is one of the famous and popular language models to extract the sentence features. Specifically, BERT is the pre-trained model for language understanding by stacking the multiple transformer[17] layers. The sequence of this process is as following:

- Firstly, the input sentence is segmented into a sequence of words and two special characters: [*CLS*] and [*SEP*] to mark the start and end of a sentence. Then, an embedding layer learns how to present the token, segment, and position representation of each character. By combining three components, each word contains the semantic and positional information with the others.
- Next, BERT takes advantage of deep bidirectional transformer layers to learn the interaction among words in a sentence. By the huge feed-forward neural layers and attention heads in the multi-head attention mechanism, it is considered the relationship and meaning of words in the sequence. The key strength of BERT is the first pre-trained language model in NLP. Nonetheless, it is proved its robustness and effectiveness in many NLP tasks including sentence similarities. In recent years, there are more and more approaches using BERT in the sentence embedding layers as a fundamental component of systems. Therefore, we also take advantage of BERT to integrate it into our model.

The detail of BERT for sentence feature extraction is presented as in **Fig. 4**. In this model, the input is object and property descriptions sentences and the output is the vector of special character [*CLS*].

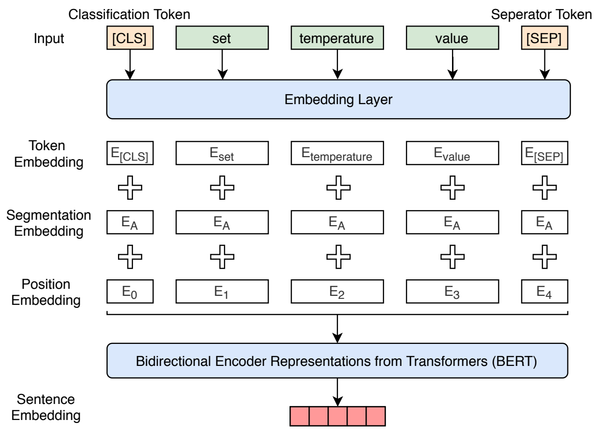


Fig. 4: Sentence Feature Extraction Model

The [*CLS*] vector indicates the start of sequence which is considered to have an ability to generate the rest of the sequence.

Vol.2020-MBL-96 No.11
Vol.2020-UBI-67 No.11
Vol.2020-CDS-28 No.11
Vol.2020-ASD-19 No.11
2020/9/29

Therefore, in most of NLP researches, the $[CLS]$ features are used to represent the sentence.

4.2 Description Extraction

After obtaining the features of the description as vectors, we need to determine the important scores for each word. In the NLP area, a similarity of words and sentences is often evaluated in the geometric measure in Euclidean space such as cosine similarity. Therefore, in this model, we use Algorithm 1 to extract the meaningful set of words through the cosine similarity measurement.

Algorithm 1: Description Extractor: Select Important Words from Description **d** Under The Budget **b**

```
   Input: description d, budget b
   Result: List of words
1  $v_d$ := BERT(d);
2  while $i < |d|$ do
3      $v_i$ := BERT($d_i$);
4      $s_i$ := sim($v_i, v_d$);
5  end
6  $\bar{W}_i := sorted(\{w_i, s_i\}_{i=0}^{|d|})$ ;
7  $c := \emptyset$ ;
8  $k := 0$ ;
9  while $((f(c) < b) \wedge (k \le |d|))$ do
10     $c := c \cup \bar{s}_k$ ;
11     $k := k + 1$ ;
12 end
13 $out$ := reorder($c, p_d$) ;
```

First, we get the representation of descriptions via the key-phrase feature extraction sub-module in Equation 1.

$$v_d = BERT(d) \in \mathbf{R^N} \tag{1}$$

Then, we get the relation score of v_d and each word's representation v_i through Equation 2. These similarities are used to sort the order of words in the priority of choice.

$$s_i = sim(v_i, v_d) = \frac{v_i v_d}{||v_i||||v_d||} = \frac{\sum_n^N v_i^n v_d^n}{\sqrt{\sum_n^N v_i^n}\sqrt{\sum_n^N v_d^n}} \tag{2}$$

To get the important words, we prefer higher similarity words to lower ones. However, we expand the selected set c through the budget b with the cost function $f(.)$. Since the standard cost function has not been defined in this step, we propose a budget ratio that corresponds to the percentage of selected words and the original description as in Equation 3.

$$f(c) = 100\frac{|c|}{|d|} \tag{3}$$

Finally, we assume that the order of selected word is similar to its position in the original description, which is controlled by the $reorder(.)$ function.

4.3 Experiments

4.3.1 Dataset

Dataset for the experiment is extracted from the *JSON Device Description* provided by the ECHONET consortium. From the *JSON Device Description*, samples are extracted as followings:

- Device name:
  - Input: The *descriptions.en* field of a JSON schema which is a sentence to describe the object.
  - Output: The *deviceType* field of the JSON schema which is a abbreviation manually created by humans.
- Property name:
  - Input: The *property.descriptions.en* field of a JSON schema which is a sentence to describe the property.
  - Output: The *propertyName* field of the JSON schema which is a abbreviation manually created by humans.

The summary of the dataset is as in **Table 2**.

Table 2: The Detail of Dataset

	Training Set	**Testing Set**
Number of Samples	100	470
Avg Number of Word: Input	4.94	4.63
Avg Number of Word: Output	3.16	3.24
Budget Ratio	73.54	77.46

4.3.2 Evaluation Metrics

Firstly, deviations of (i) machine-generated outputs and (ii) human-created outputs from original inputs are calculated. As for the result, the smaller deviation score shows better performance. The **Edit Distance**[18] and the **Word Error Rate**[19] are selected as the first evaluation metric.

- **Edit Distance** (ED)-Levenshtein distance: A method to quantify how dissimilar two strings are by counting the minimum number of operations required to transform one string into the other. The ED works at the phoneme level of a character string with the transform function including substitution, deletion, and insertion.
- **Word Error Rate** (WER): This is a deviation of the Levenshtein distance which deals with the word level instead of the phoneme level.

Secondly, semantic preservation scores of (i) machine-generated outputs and (ii) human-created outputs from original inputs are calculated. As for the result, the greater score shows better performance because one of the critical requirements of the automated generation solution is to maintain the semantics of descriptions through the extraction phase. The **Cosine Similarity** that shows the distance in the geometric space of the input and the output vectors by calculating angles between these vectors, has been utilized for this metric.

Lastly, the correlation coefficient of (i) machine-generated outputs and (ii) human-created outputs from original inputs are calculated. This coefficient implies the correlation with human judgments in grammar and meaning preservation[20] of machine-generated outputs. In Machine Translation, **BLEU**[21] calculates the n-grams overlap of automatic and manual samples to evaluate the similarity between the references (human-created samples) and prediction (machine-generated samples). In this experiment, **BLEU** is selected as the last evaluation metric.

4.3.3 Results

Experiment results are separated into 3 metrics includes (i) Deviation Scores, (ii) Semantic Preservation Scores, and (iii) Machine Generation Correlation Coefficient as followings:

Deviation Scores: Differences at the character level as well as

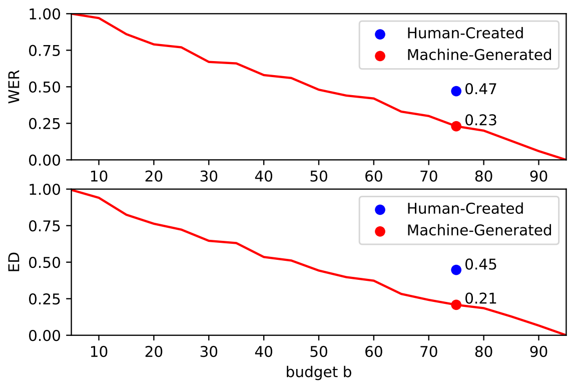


Fig. 5: Word Error Rate and Normalized Edit Distance Between human-Created and Machine-Generated Samples Comparing to Original Descriptions

the word level of machine-generated samples, and human-created samples are visualized in **Fig. 5**. WER and normalized ED Scores of human-created samples are constant values **0.47** and **0.45** (at 75% of the budget) respectively. At the same rate of 75% of the budget as human-created samples, WER and normalized ED scores of the machine-generated samples are **0.23** and **0.21** which show fewer deviations, and hence achieve the better performance.

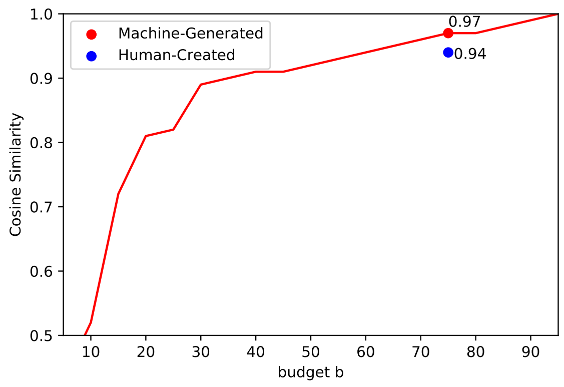


Fig. 6: Cosine Similarity Between Human-Created and Machine-Generated Samples Comparing to Original Descriptions

Semantic Preservation Scores: Scores that imply the level of semantic preservation after shorting original descriptions to generate *device name* and *property name* of machine-generated approach and human-created approach are summarized in Fig. 5. At the constant rate of 25% of the budget, human-created samples preserve **94%** the meaning of original descriptions. Meanwhile, at the same budget ratio, machine-generated samples preserve **97%** of the original descriptions. The proposed NLP model achieves the same score **94%** as humans at approximately **60%** of the budget. Results show that machine-generated samples can (i) preserve **higher level of semantic** at the same length of human-created samples or (ii) preserve the **same level of semantic** as humans with the **shorter** outputs.

Machine Generation Correlation Coefficient: The correlation coefficient of machine-generated samples and human-created

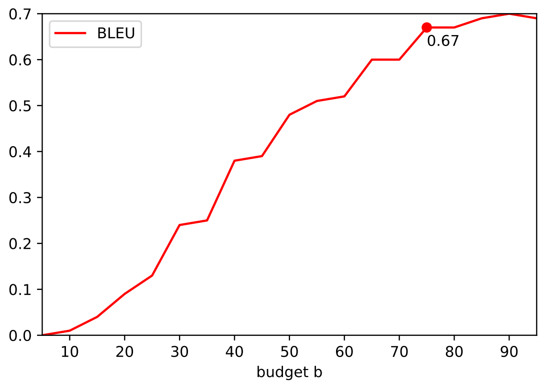


Fig. 7: Correlation Coefficient of Machine-Generated Samples and Human-Created Samples using BLEU

samples that shows the degree of similarity between the implemented NLP model and humans, is visualized in **Fig. 7**. Currently, the BLEU score at **0.35** is considered as human-like solutions in Machine Translation[22]. At the rate **75%** of the budget, the BLEU score of the implemented model is **0.67**. Even though the implemented NLP model is an extractive approach for short descriptions, the result shows that the implemented model can totally imitate humans in exporting creating definitions for data models.

4.3.4 Discussion

In this section, three typical cases include **Good**, **Bad**, and **Fair** of machine-generated property names are discussed. A good case is as followings:

- Input: *vegetable compartment temperature setting*
- Human-Created Result: *vegetableTemperature*
- Machine-Generated Result: *vegetableCompartmentTemperature*

The property created by humans is shorter and understandable however to describe a property of a refrigerator, the word *compartment* is important and should not be eliminated.

A bad case is as followings:

- Input: *remaining stored electricity 2*
- Human-Created Result: *remainingCapacity2*
- Machine-Generated Result: *remainingStoredElectricity*

The machine-generated property name is semantically better than the human-created one. However, the number *2* is important to identify the property. To handle other cases, an effort to create rules for word choice is required.

A typical fair case is as followings:

- Input: *illuminance level*
- Human-Created Result: *brightness*
- Machine-Generated Result: *illuminanceLevel*

This is a fair case because the generation of synonyms is not yet supported by the NLP model.

5. Syntactic Serializers

To enhance the reusability and the extensibility of the *eModelGen*, the *Semantic Analyzer* and the *Sentactic Serializers* are decoupled. By implementing the corresponding serializer, a target data model could be exported. In the scope of this paper, a data model for ECHONET Lite Web APIs (namely *Device Description*), and a data model that supports ECHONET Lite in the FIWARE (namely *EL Smart Data Model*) are introduced. Both data models are based on JSON, however, XML, RDF, YANG, and ontology-based models could be extended in the same manner. The structure of JSON schemes of *Device Description* and *EL Smart Data Model* are summarized in **Fig. 8** and **Fig. 9** respectively. Both of the JSON schemes share the same *deviceObjectName* and *propertyName* which are outputted from Section **4**. The *data type* definition is the last piece of the puzzle.

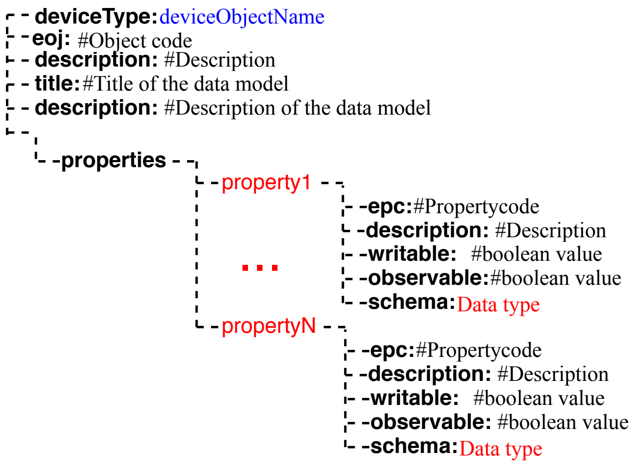

deviceType: deviceObjectName
eoj: #Object code
description: #Description
title: #Title of the data model
description: #Description of the data model
properties
property1
epc: #Propertycode
description: #Description
writable: #boolean value
observable: #boolean value
schema: Data type
...
propertyN
epc: #Propertycode
description: #Description
writable: #boolean value
observable: #boolean value
schema: Data type

Fig. 8: JSON Schema of the Data Model for the ECHONET Lite Web APIs (*Device Description*)

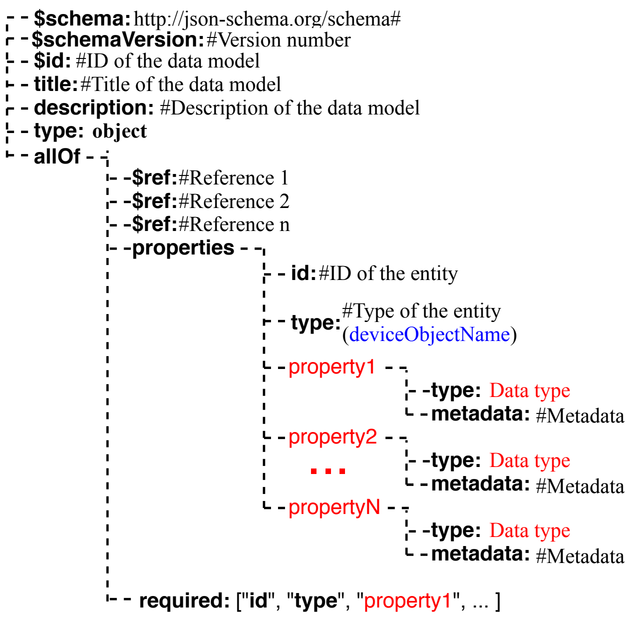

$schema: http://json-schema.org/schema#
$schemaVersion: #Version number
$id: #ID of the data model
title: #Title of the data model
description: #Description of the data model
type: object
allOf
$ref: #Reference 1
$ref: #Reference 2
$ref: #Reference n
properties
id: #ID of the entity
type: #Type of the entity (deviceObjectName)
property1
type: Data type
metadata: #Metadata
property2
type: Data type
metadata: #Metadata
...
propertyN
type: Data type
metadata: #Metadata
required: ["id", "type", "property1", ...]

Fig. 9: JSON Schema of FIWARE Smart Data Models (*EL Smart Data Model*)

The ECHONET device objects specification includes data type definitions, however, those definitions are incompatible with the JSON-Schema[23] definitions which are being used to provided data formats for most of the current Web APIs. Since data type formats of the JSON-Schema are also supported in the FIWARE ecosystem and ECHONET Lite Web APIs, *data type* fields (in Fig. 8 and Fig. 9) are mapped from the ECHONET device objects specification as in **Table 3**.

Table 3: Data Type Mapping Rules from ECHONET Device Objects Specification into JSON-Schema

	ECHONET Device Objects Specification	**FIWARE EL Smart Data Model**	**Web APIs Device Description**
Naming Conventions	Device Object Name	EntityType	deviceType
	Property Name	propertyName	propertyName
Data Type Definitions	Raw type	String type	
	Number type Numerical value type Level type	Number type	
	State type (2 values)	Boolean type	
	State type (>2 values)	Array type	
	Object type Bitmap type	Object type	
	Date-time type	Date-time type	
	Time type	Time type	

Because the target data models are in the form of JSON schemes, JSON serializers are created using a Java library, namely *JSON.simple*. As a result, the total number of **117** JSON schemes of each data model has been generated within several seconds and **100%** of generated schemes have been passed the JSON syntactic checker.

6. Concluding Remarks

In accordance with the developments of Web APIs for the IoT, data models that allow both API providers and consumers to be able to interoperate, are emerging. This paper proposed a solution to pave the way to generate data models automatically using machines to assist and reduce human efforts. The proposed solution provides (i) a *semantic analyzer* to imitate humans in creating definitions by utilizing NLP techniques, and (ii) *syntactic serializers* to generate data models in a fast and reliable manner. By decoupling those building blocks, the reusability and extensibility in supporting data models for different platforms are assured.

The proposed semantic analyzer is able to mimic humans to create property names and object names from the ECHONET device objects specification. Experimental results show that definitions generated by the machine are better than humans in preserving the semantic of original descriptions while keeping it shorter. Nevertheless, there are cases where only semantic is not enough, and the verification as well as corrections by experts are needed at the last step to release data models. Therefore, the NLP model could be utilized as a recommendation system.

The proposed syntactic serializers have beat humans in syntactic exporting because machines have the ability to work tirelessly and speedily. In this paper, two JSON serializers have been implemented to export data models for ECHONET Lite Web APIs and FIWARE Smart Data Models. As a result, within several seconds, 117 JSON schemes for each data model have been exported and 100% of the generated JSON schemes pass the syntactic checker.

The extension of an ontology serializer is desired as future work.

References

[1] Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N. and Weerawarana, S.: Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI, *IEEE Internet Computing*, Vol. 6, No. 2, pp. 86–93 (online), DOI: 10.1109/4236.991449 (2002).

[2] Vinoski, S.: RESTful Web Services Development Checklist, *IEEE Internet Computing*, Vol. 12, No. 6, pp. 95–96 (online), DOI: 10.1109/MIC.2008.130 (2008).

[3] Tan, W., Fan, Y., Ghoneim, A., Hossain, M. A. and Dustdar, S.: From the Service-Oriented Architecture to the Web API Economy, *IEEE Internet Computing*, Vol. 20, No. 4, pp. 64–68 (online), DOI: 10.1109/MIC.2016.74 (2016).

[4] Cambria, E. and White, B.: Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article], *IEEE Computational Intelligence Magazine*, Vol. 9, No. 2, pp. 48–57 (online), DOI: 10.1109/MCI.2014.2307227 (2014).

[5] ECHONET CONSORTIUM: *Detailed Requirements for ECHONET Device objects* (2020).

[6] Kodama, H.: The ECHONET Lite Specifications and the Work of the ECHONET Consortium, *New Breeze - Quarterly of the ITU Association of Japan*, Vol. 27, No. 2, pp. 4–7 (2015).

[7] Umejima, M.: Japan's Power Meter Deployment with ECHONET Lite over IPv6, *New Breeze - Quarterly of the ITU Association of Japan*, Vol. 27, No. 2, pp. 12–13 (2015).

[8] Fujita, H., Sugimura, H., Hamamoto, M. and Isshiki, M.: Improvement of the Device Descriptions of ECHONET Lite by adding version specific information, *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, pp. 791–793 (2019).

[9] TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU: *Requirements and architecture of the home energy management system and home network services* (2015).

[10] Pham, V. C., Lim, Y., Sgorbissa, A. and Tan, Y.: An Ontology-driven ECHONET Lite Adaptation Layer for Smart Homes, *Journal of Information Processing*, Vol. 27, pp. 360–368 (online), DOI: 10.2197/ipsjjip.27.360 (2019).

[11] Hanke, S., Mayer, C., Hoeftberger, O., Boos, H., Wichert, R., Tazari, M.-R., Wolf, P. and Furfari, F.: Berlin, Heidelberg (2011).

[12] Ed-douibi, H., Cánovas Izquierdo, J. L. and Cabot, J.: Example-Driven Web API Specification Discovery, *Modelling Foundations and Applications* (Anjorin, A. and Espinoza, H., eds.), Cham, Springer International Publishing, pp. 267–284 (2017).

[13] Cirillo, F., Solmaz, G., Berz, E. L., Bauer, M., Cheng, B. and Kovacs, E.: A Standard-Based Open Source IoT Platform: FIWARE, *IEEE Internet of Things Magazine*, Vol. 2, No. 3, pp. 12–18 (2019).

[14] Li, H., Seed, D., Flynn, B., Mladin, C. and Di Girolamo, R.: Enabling Semantics in an M2M/IoT Service Delivery Platform, *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, pp. 206–213 (2016).

[15] ECHONET CONSORTIUM: *ECHONET Lite Web API* (2020).

[16] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, Association for Computational Linguistics, pp. 4171–4186 (online), DOI: 10.18653/v1/N19-1423 (2019).

[17] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u. and Polosukhin, I.: Attention is All you Need, *Advances in Neural Information Processing Systems 30* (Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S. and Garnett, R., eds.), Curran Associates, Inc., pp. 5998–6008 (2017).

[18] Ristad, E. S. and Yianilos, P. N.: Learning string-edit distance, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 5, pp. 522–532 (1998).

[19] Popovic, M. and Ney, H.: Word error rates: Decomposition over POS classes and applications for error analysis, *Proceedings of the Second Workshop on Statistical Machine Translation*, pp. 48–55 (2007).

[20] Xu, W., Napoles, C., Pavlick, E., Chen, Q. and Callison-Burch, C.: Optimizing Statistical Machine Translation for Text Simplification, *Transactions of the Association for Computational Linguistics*, Vol. 4, pp. 401–415 (online), DOI: 10.1162/tacl_a_00107 (2016).

[21] Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J.: Bleu: a Method for Automatic Evaluation of Machine Translation, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA, Association for Computational Linguistics, pp. 311–318 (online), DOI: 10.3115/1073083.1073135 (2002).

[22] Edunov, S., Ott, M., Auli, M. and Grangier, D.: Understanding Back-Translation at Scale, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, Association for Computational Linguistics, pp. 489–500 (online), DOI: 10.18653/v1/D18-1045 (2018).

[23] Pezoa, F., Reutter, J. L., Suarez, F., Ugarte, M. and Vrgoc, D.: Foundations of JSON Schema, Republic and Canton of Geneva, CHE, International World Wide Web Conferences Steering Committee, (online), DOI: 10.1145/2872427.2883029 (2016).

Van Cu Pham received his B.Eng. degree in software engineering from FPT University in 2014. He received his M.S. and Ph.D. degree in Information Science from the Japan Advanced Institute of Science and Technology (JAIST) in 2016 and 2020, respectively. Currently, he is a post-doc researcher at the Center for Trustworthy IoT Infrastructure, JAIST. His research interests include home network system, home network management, network simulation. He is a member of IEICE and IPSJ

Thanh Tung Le is currently a Ph.D student of Information Science at the Japan Advanced Institute of Science and Technology, Ishikawa, Japan (JAIST). He received the B.S degree in Computer Science (2012) of Honour Program from University of Science, Vietnam National University, Ho Chi Minh city, Vietnam and the M.S degree in Information Science (2018) from JAIST. His current research interests include information retrieval, deep learning, natural language processing.

Tien Huy Nguyen is currently an Lecturer of Computer Science at the University of Science, Ho Chi Minh City, Vietnam (HCMUS). He leads the lab on Natural language Processing and Speech Processing at Zalo, VNG. He received the B.S degree in Software Technology (2010) and the M.S degree in Computer Science (2015) from HCMUS. He received his Ph.D. degree in Information Science from School of Information Science, Japan Advanced Institute of Science and Technology (JAIST) in 2019. His current research interests include information retrieval, deep learning, speech recognition and text to speech.

Yasuo Tan was born in 1965. He received his Ph.D from Tokyo Institute of Technology in 1993. He joined Japan Advanced Institute of Science and Technology (JAIST) as an assistant professor at school of information science in 1993. He has been a professor since 2007. He is interested in IoT Systems especially Smart Home Systems. He is the chair of Technology Standards subcommittee of Smart IoT Acceleration Forum, member of Information and Communication Council, the chairman of Green Grid Platform at Home Alliance, an advisory fellow of ECHONET Consortium, and a member of IEEE, ACM, IPSJ, IEICE, IEEJ, JSSST, and JNNS.