A Services Development Approach for Smart Home Based on Natural Language Instructions

Yiyan Chen[1,2], Zhanghui Liu[1,2], Zhiming Huang[1,2], Chuanshumin Hu[1,2], and Xing Chen[1,2,*]

[1]College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China
[2]Key Laboratory of Network Computing and Intelligent Information Processing
Fuzhou University, Fuzhou 350116, China
[*]chenxing@fzu.edu.cn

***Abstract*—With development of the infrastructures supporting smart home which has entered in a new stage featured by intelligent services. The services based on natural language instructions are aimed at simplifying and improving our lives. To customize and develop these services more efficiently, this paper proposes an approach to model and execute services based on natural language instructions at runtime which introduces the knowledge graph into development process. Firstly, a concept model of knowledge graph is introduced for smart home services. Secondly, we put forward the mechanism to construct the instance of knowledge graph for smart home services. Finally, rules of transformation are provided to achieve mapping from natural language instructions to the services of knowledge graph. We evaluate our model on a prototype system and the experimental results show that our approach can develop services at runtime and effectually reduce the lines of code by 90%.**

***Index Terms*—services development, smart home, knowledge graph, runtime model, natural language**

I. Introduction

The continuous development of smart home infrastructure has ushered in a new era characterized by smart services. A large number of new devices which are complex and heterogeneous such as intelligent robots, smart wearable devices and smart home appliances will access to the network for interaction and collaboration [1] [2]. Human-computer interaction based on natural language has become the main interaction method in smart home like Amazon Echo [3] and Google Home [4].

Smart home service based on natural language commands is actually a process of decision-making and implementation according to the instructions given by users and the information of their environment [5]. At present, the management interface of the device is directly invoked by a programming language such as C or Java to implement services development of smart home. Although this kind of programming has good adaptability, it will bring high price. Developers must be familiar with the management interfaces of different smart devices in order to implement the interaction. In addition, the application system is based on the underlying code bound to a specific intelligent device, which makes it impossible to reuse its management logic. It is hard to expand the manage system when the devices change, though the management mechanism is similar.

DOI reference number: 10.18293/SEKE2019-173

Due to the gap between problem domains and system implementations, accomplishing mapping between them can create significant programming complexity. The knowledge graph is used to describe the concepts, entities, events and relationships between the object, which can serve as a bridge between system requirements and system implementation. Moreover, it turns to be easier to integrate the devices into control system of smart homes. If we want to introduce knowledge graph into the development of smart home services, there are two problems which need to be solved. One is how to define the knowledge graph model to describe the smart home scene. The other is how to map natural language utterances to services provided by devices in the knowledge graph.

In this paper the method is proposed to quickly customize and develop smart home service on the basis of natural language instructions. Our contributions are as following:

- A concept model of knowledge graph is introduced for smart home services.
- We put forward the mechanism to constructing the instance of knowledge graph for smart home services.
- Transformation rules are proposed to achieve mapping from natural language instructions to the services of knowledge graph.

II. Related work

In the development process of IoT applications, programming is usually carried out at the operating system level. This situation makes the programmer to focus on the underlaying related issues rather than the application logic [6]. Guang et al. [7] proposed a top-down IoT application development method and support system. The system could automatically generate platform-specific configuration and execution code by given the platform-independent application management logic. In addition, some research work [8] [9] proposed a service-oriented architecture-based solution that provided a device access interface in the form of RESTFul service to shield the heterogeneity and complexity of the underlying system.

The Peking University team conducted research on runtime model theory and construction methods [10] [11]. When the system meta-model and a set of management interfaces

was given, the SM@RT tool [14] can automatically generate code. When the systematic meta-model changed, SM@RT can generate new mapping code automatically.

III. Approach overview

A. Concept model of knowledge graph

The concept model of knowledge graph provides an abstract representation of smart home services, and aim to describe the concepts and relationships of abstract elements for smart home scene, as shown in Figure 1.

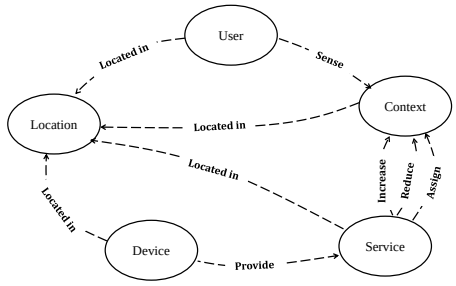

Fig. 1. Concept model of knowledge graph for services of smart home

As Table I illustrates, the concept model defines the concepts of the smart home scenario such as location, user, context, device and service. We give further explanations for those concepts and their properties in this model.

TABLE I
Concepts and properties in the conceptual model of smart home knowledge graph

Concept Name	Conceptual Properties
Location	<LName>
User	<UName, LName>
Context	<UName, LName, CType, CValue>
Device	<DName, LName, $\{Key_1, Key_2, ..., Key_n\}$ >
Service	<DName, LName, DFunction, CType, Effect>

$Location$ indicates a specific area and the name of the area is expressed by $LName$. $User$ is used to describe the service object and $UName$ is the name of user. The attribute $LName$ is the area where the user is located. $Context$ represents an environment state of users. $UName$ is the name of current service object. $LName$ represents the area where the users are located. $CType$ is to describe an type of environment status (eg brightness, temperature). $CValue$ is the value of status. $Device$ represents the equipment object and it contains three properties $LName$, $DName$ and Key_i. Key_i means the configuration parameter or system indicator of device. $Service$ indicates a service to change the environment or operating the device and it contains five properties. We need to emphasize the $DFunction$ and $Effect$ here. $DFuction$ is the function interface of the device corresponding to the service. $Effect$ is an operation of changing the environment or managing the devices meeting the instruction given by user.

The conceptual model also defines the relationship between concepts mentioned above, as shown in Figure 1. For example, $X \xrightarrow{Located} L$ indicates node $User$, $Context$, $Device$ and $Service$ are located in area L. $U \xrightarrow{Sense} C$ represents user is aware of the state of the context. $S \xrightarrow{Increase} C$ shows that the service is used to raise the state value of the context.

B. Mapping natural language to service

The implementation of the smart home services in natural language is based on the knowledge graph model of smart home. The goal is to map natural language commands to specific service and the knowledge graph contains all the information of services. Hence, we have two main tasks in this part.

Firstly, the task of information extraction is to identify the semantics in natural language commands. At present, information extraction methods are mainly divided into two categories, one is based on statistical learning method and the other is based on rule pattern matching method. Statistical learning method is portable and robust. However, it needs a lot of training data to set parameters and optimize the system. Rule-based matching method has high efficiency and accuracy. The text of smart home domain has the characteristics of domain, regularity and simple logic. The rule-based method can achieve better extraction effect.

Secondly, the conversion rules are builded to map the information to services of knowledge graph. When services matching the commands are found, it is available to execute the device functional interfaces satisfied with users' need. The specific transformation rules are given in Section V.

IV. Runtime modeling

In this part, we will introduce the runtime modeling method of knowledge graph for smart home. With the purpose of describing the context knowledge of smart home, we establish knowledge graph instances by constructing concept instances and relationship instances.

A. Concept instances modeling

The concept instances are constructed based on the developer configuration. For realizing bidirectional synchronization between concept instances and the real-time information of the scene, a set of rules for mapping are proposed. Developers need to provide relevant configurations to describe specific objective facts in the scenario, including scene information, the mapping of concept instances to smart devices and the environment of service objects.

The configuration of scene information provides a collection of locations $L = \{L_1, L_2, ..., L_n\}$. The mapping between concept instances and devices provides a collection of devices $D = \{D_1, D_2, ..., D_n\}$, a collection of services $S = \{S_1, S_2, ..., S_n\}$ and the relationship between them. The configuration of users' environment provides data which include collection of users $U = \{U_1, U_2, ..., U_n\}$, positioning devices $T = \{T_1, T_2, ..., T_n\}$, and environmental state of users $C = \{C_1, C_2, ..., C_n\}$. Then the corresponding concept instances are generated by the collections mentioned. Moreover,

TABLE II
MAPPING RULES FOR MODEL OPERATIONS OF LOCATION, USER AND CONTEXT INSTANCES

	Location	User	Context
List	$List * L \rightarrow \{L_1, L_2, ..., L_n\}$	$List * U \rightarrow \{U_1, U_2, ..., U_n\}$	$List * C \rightarrow \{C_1, C_2, ..., C_3\}$
	$Get\ L_i.properties \rightarrow L_i.properties$	$Get\ U_i.properties \rightarrow U_i.properties$	$Get\ C_i.properties \rightarrow C_i.properties$
Get	$Get\ L_i.LName \rightarrow L_i.LName$	$Get\ U_i.UName \rightarrow U_i.UName$ $Get\ U_i.LName \rightarrow \mathbf{RTModel}\,(Get\ T_i.location)$	$Get\ C_i.UName \rightarrow C_i.UName$
			$Get\ C_i.LName \rightarrow Get\ U_j.LName\ \left((\exists U_j)\, U_j \xrightarrow{Sense} C_i\right)$
			$Get\ C_i.CType \rightarrow C_i.CType$
			$Get\ C_i.CValue \rightarrow Get\ D_j.key_n\ \left((\exists D_j)\, D_j \xrightarrow{Provide} S_i\right)$
Set	-	-	-

TABLE III
MAPPING RULES FOR MODEL OPERATIONS OF DEVICE AND SERVICE INSTANCES

	Device	Service
List	$List * D \rightarrow \{D_1, D_2, ..., D_n\}$	$List * U \rightarrow \{U_1, U_2, ..., U_n\}$
	$Get\ D_i.properties \rightarrow D_i.properties$	$Get\ S_i.properties \rightarrow S_i.properties$
Get	$Get\ D_i.DName \rightarrow D_i.DName$ $Get\ D_i.LName \rightarrow D_i.LName$ $Get\ D_i.key_m \rightarrow \mathbf{RTModel}\,(Get\ D_i.key_m)$	$Get\ S_i.DName \rightarrow S_i.DName\ \left((\exists D_j)\, D_j \xrightarrow{Provide} S_i\right)$
		$Get\ S_i.LName \rightarrow S_i.LName$
		$Get\ S_i.CType \rightarrow S_i.CType$
		$Get\ S_i.Effect \rightarrow S_i.Effect$
		$Get\ S_i.DFaction \rightarrow S_i.DFunction$
Set	$Get\ D_i.key_m \rightarrow \mathbf{RTModel}\,(Set\ D_i.key_m)$	-

the SM@RT tool [10] [11] is used to construct the runtime model of the smart devices and positioning devices.

The model operation methods of the concept instances mainly include three types, namely $List$, Get and Set. $List$ is used to get all instances of the same type and its properties. Get and Set are used to read and write the attribute values of instances respectively. In order to maintain the bidirectional synchronization of the concept instances and the scene realtime information, the mapping rules are defined, as shown in Table II and Table III. There are three main mechanisms for the bidirectional synchronization of knowledge graph and the real-time information of scene by configuration information, runtime models and rules respectively. As Table II shown, the attribute $LName$ of location L_i is from the configuration like $Get\ L_i.LName \rightarrow L_i.LName$. And there are similar situations in other instances of concept. The $LName$ of the user U and the Key_m of the device need to be obtained through the runtime model, such as $Get\ U_i.LName \rightarrow \mathbf{RTModel}\,(Get\ T_i.location)$. The $LName$ and $CValue$ of the context as well as $LName$ of the service need to be obtained by rules, such as $Get\ C_i.LName \rightarrow Get\ U_j.LName\ \left((\exists U_j)\, U_j \xrightarrow{Sense} C_i\right)$.

B. Relation instances modeling

As illustrated in Table IV, we further define the rules for constructing relation instance to express the relationships between concepts in a smart home scenario. When two instances of a concept meet certain preconditions, the relationship between them is constructed. For instance in Rule 4, if the area $LName$ and the $CType$ of the service S_i is equal to the context C_j, and $Effect$ of S_i is $Increase$, the relationship instance $S_i \xrightarrow{increase} C_j$ is created. In Rule 2 and 3, the relationship $Scence$ and $Provide$ involve the user's environmental information and the device's features in the real scenario, so this kind of information needs to be obtained from the developers' configuration data.

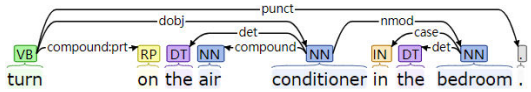

Fig. 2. Instance of dependency parsing tree

V. MAPPING NATURAL LANGUAGE TO SERVICES

In this chapter, we first apply dependency parser to information extraction. Secondly, knowledge reasoning rules are specified to transform the results of information extraction into the search of service nodes in the knowledge graph.

Dependent syntax explains the syntactic structure of sentence by analyzing the dependencies between components within a language unit, as shown in Figure 2. According to the characteristics of commands in smart home scene, we chose dependency parsing to extract the information from natural language instructions.

The final goal is to determine the $Service$ instance S_i based on the properties $DName(D)$, $LName(L)$, $CType(C)$ and $Effect(E)$ to invoke the $DFuction$. Therefore, the task of information extraction is to identify four kinds of the mentioned information. This will allow us to convert any natural language utterance to a canonical representation, which we can map to services.

A. Rules of Extraction

In this paper, Stanford Parser is applied for dependency parsing. The generation of the extraction rules mainly depends on the Part of Speech (POS) of the extraction task and the dependency characteristics.

TABLE IV
RULES FOR CONSTRUCTING RELATION INSTANCES

	Relation Instance	**Precondition**
1	$X_i \xrightarrow{Locate} L_j$	$X_i.LName = L_j.LName$
2	$U_i \xrightarrow{Sense} C_j$	–
3	$D_i \xrightarrow{Provide} S_j$	–
4	$S_i \xrightarrow{increase} C_j$	$S_i.LName = C_j.LName \wedge S_i.CType = C_j.CType \wedge S_i.Effect =$ "$Increase$"
5	$S_i \xrightarrow{Reduce} C_j$	$S_i.LName = C_j.LName \wedge S_i.CType = C_j.CType \wedge S_i.Effect =$ "$Reduce$"
6	$S_i \xrightarrow{Assign} C_j$	$S_i.LName = C_j.LName \wedge S_i.CType = C_j.CType \wedge S_i.Effect =$ "$Assign$"

1) Extraction Rules of "Effect": $Effect$ is the operation of changing the environment or managing the device in commands given by users. By summarizing the commands for the smart home scene, we found that an operation is usually represented by a single verb or a verb phrase, like "increase" and "turn on". There is a specific dependency "cpmpound:prt" between words in a verb phrase. The dependency path of the verb part of the sentence is shown in Figure 3. In algorithm 1, we elaborate how to extract the "Effect" according to the dependency relationship. The identifier p is used to represent the nodes in the dependency parsing tree. For example, every word in Figure 2 can be expressed as p. We use $p.word$ to describe content of the node p. $p{\rightarrow}son$ illustrates the child node of p. The part of speed of p is represented by $p.pos$. $p{\rightarrow}head$ expresses the parent node of p. Above all, we imply $r(p, q)$ to describe the dependency relationship between node p and q.

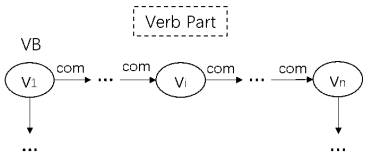

Fig. 3. Dependency paths of verb part

Algorithm 1 Extracting the "Effect" information

Input: Dependency parsing of the command;
Output: "Effect" information E
Find the node p whose part of speech is "VB", E=p.word;
while p has a child node $p{\rightarrow}son$ and the relationship $r\,(p, p \rightarrow son)$ is "compound:prt" **do**
  E=E+$(p \rightarrow son)$.word;
  p=p$\rightarrow$ son;
end while

2) Extraction Rules of DName, LName and CType: Mapping the attributes $DName$, $LName$ and $CType$ to the commands is the information of device, area and attribute of environment or device. The expression of these three kinds of information can be summarized in the following two ways.

- Entity of a single noun, like "light", "temperature", "bedroom".
- Entity composed by phrase. It can be divided into two categories. One is the form of "noun+noun" and their relationship is "compound", like "air condition". The other is the form of "adjective+noun", their relationship is "amod", like "sitting room".

The data stored in the knowledge map is matched with the extracted information to classify. In summary, the extraction rules for the three kinds of information as algorithm 2 shows.

Algorithm 2 Extracting the "device", "location" and "attribute" information

Input: Dependency parsing of the command;
Output: "device" information D, "location" information L, "attribute" information A
Find the node p whose part of speech is "NN", C=p.word;
while p has a child node $p{\rightarrow}son$ and the relationship $r\,(p, p \rightarrow son)$ is "amod" or "compound" **do**
  N=N+$(p \rightarrow son)$.word;
  p=p$\rightarrow$ son;
end while
Look for knowledge graph node S_i;
Judge the type of information N is $DName$, $LName$ or $CType$.

B. Rules for Knowledge Reasoning

After information extraction is done, we need to map extracted information to service in knowledge graph through transformation mechanism. Universal rules of knowledge reasoning is proposed in term of different extraction results for matching the services. Rules are listed in Table V.

Rule 1, 2 and 3 outline that the user directly specifies the operation of a device. For example, when the user specifies ""D, "L", "C" and "E", construct rule 1. If there is service instance S_i and the value of attribute $LName$ is L, $DName$ is D, $CType$ is C, and $Effect$ is E, call the function interface $DFunction$ of service S_i.

Rules 4 and 5 describe the knowledge reasoning based on the user's environment when the information given by the user is insufficient to locate the specific service. For example, when the user gives the operation and device, construct rule 4. Firstly, the location information of the device is obtained according to the $LName$ of the current user U_i. Then, we

need to find a service S_j that meets the following conditions. The attribute $LName$ value of S_i is equal to $LName$ of U_i, $DName$ of S_i is D, and $Effect$ is E. Finally, the function interface pointed to by the attribute $DFunction$ of the service S_i is called.

VI. Evaluation

In this section, we build a prototype system of smart home for the actual scene to evaluate the method. Firstly, we evaluate validity of the modeling method for knowledge graph instances. Secondly, we compare the development difficulty and execution performance of the service with the traditional method. Finally, we discuss the precision of information extraction.

A. Study Case

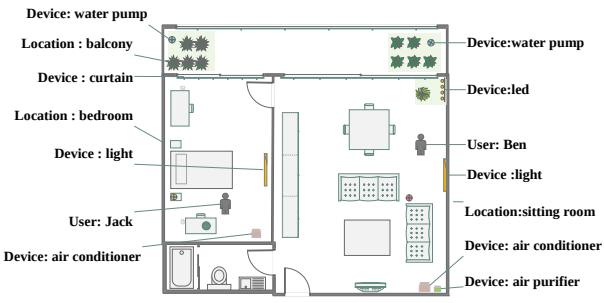

Fig. 4. Example scenario of smart home

There are 3 areas in the scene, which are the sitting room, bedroom and balcony, as shown in Figure 4. For example, there are some smart devices such as an air conditoner, lights and an air purifier in the sitting room. The scene includes four types of environmental states: temperature, humidity, brightness and particulate matter. There are services such as “turn on”, “turn off”, “increase”, “reduce” and so on for various functions of different devices. Fig. 5 shows the instance of the knowledge graph which is generated with respect to the smart home scene.

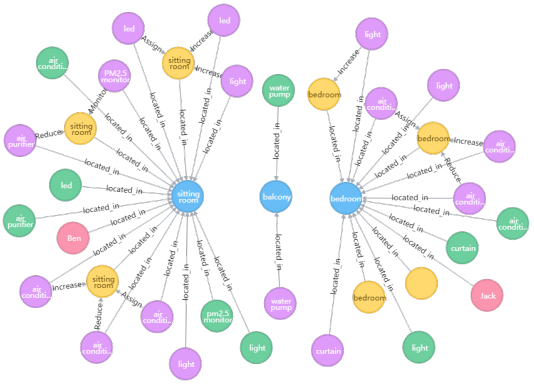

Fig. 5. Instance model of the knowledge graph for smart home

B. Evaluation of the Model

In this section, we develop smart home services based on Java to verify the effectiveness of the method, comparing the process and the difficulty of services development with the method we proposed.

1) Process of Service Development: To develop services based on Java, developers must be familiar with the management interfaces of different intelligent devices and realize their interaction. Moreover, because services are based on these management interfaces, their management logic cannot be reused.

We built the runtime model of the smart devices with the help of the SM@RT tool [10] [11], which realizes the data read-write and function invocation of the devices in a unified manner (SM@RT is available from the reference [12]). SM@RT(supporting model at runtime) is a tool for constructing the runtime software architecture by model-driven, including domain-specific modeling language (SM@RT language) and code generator (SM@RT generator).

SM@RT language allows users to define the meta-model and accessing model of run time software architecture. The meta-model defines the structure and manageable elements of the target system. The accessing model declares the methods of managing these elements in meta-model.

SM@RT generator can automatically generate and maintain the infrastructure of run time software architecture based on meta-model and access model, and reflect the real-time state of the underlying system to the run time model. At the same time, the intelligent device run time model only needs to be constructed once then it can be reused in different smart home scenarios. Therefore, it does not bring extra work.

Based on the run time model of smart devices, developers can automatically build voice control services in smart homes by describing Scenario-Oriented knowledge, configuring the mapping relationship between conceptual instances and smart devices, and describing the environment of service objects.

2) Difficulty in Service Development: Table VI compares lines of code for smart home services developed by the two methods. We develop services in Java, and each service is developed independently. When using the method we proposed, developers need to finish the scenario oriented configuration, and the basic code are 115 lines.

For example, it takes 197 lines of code to implement the service S_{11} for temperature adjustment performed by Jack. The workload for these two parts in service S_{11} can be 126 and 71 lines respectively. It only takes 6 lines of configuration code to achieve the same functionality in our method. Implementing these services by traditional method needs 188 lines of code in average, but only 16 lines are required using the proposed method, reducing the workload by 90%.

C. the Accuracy of Information Extraction

We convene 50 volunteers to give natural language instructions based on the prototype system. Because the smart home scene we built is small, the corpus obtained is limited. In the end, we receive 1321 responses regarding what the respondents

TABLE V
RULES FOR KNOWLEDGE REASONING

Condition	Reasoning Rules
D+L+A+O	$(\exists S_i)\,((S_i.DName = D) \wedge (S_i.LName = L) \wedge (S_i.CType = A) \wedge (S_i.Effect = O)) \Rightarrow S_i.DFunction$
D+L+O	$(\exists S_i)\,((S_i.DName = D) \wedge (S_i.LName = L) \wedge (S_i.Effect = O)) \Rightarrow S_i.DFunction$
L+A+O	$(\exists S_i)\,((S_i.LName = L) \wedge (S_i.CType = A) \wedge (S_i.Effect = O)) \Rightarrow S_i.DFunction$
D+O	$(\exists S_j)\,(\exists U_i)\,((U_i.UName = uname) \wedge (S_j.LName = U_i.LName) \wedge (S_j.DName = D) \wedge (S_j.Effect = O)) \Rightarrow S_j.DFunction$
A+O	$(\exists U_i)\,(U_i.UName = uname) \wedge (\forall S_j)\,((S_j.LName = U_i.LName) \wedge (S_j.CType = A) \wedge (S_j.Effect = O)) \Rightarrow S_j.DFunction$

TABLE VI
COMPARISON IN LOC OF TWO APPROACHES

	$Basic$	S_{11}	S_{12}	S_{13}	S_{21}	S_{22}	S_{23}	S_{33}	S_{34}	S_{42}	S_{43}	$Avarage$
Java	0	197	231	181	210	189	140	216	184	140	195	188
Our Approach	115	6	6	6	6	6	6	6	6	6	6	16

want their smart home to do. We extracted the information from these commands and the results of the evaluation are shown in Table VII.

We will discussed the results according to the evaluation data. The accuracy of $DName$, $LName$, $CType$ and $Effect$ extraction from the table is 93%, 90%, 84% and 80%. Because users have different expressions of the same concept or entity. For example, "sitting room" and "living room". Therefore, when the user does not use the same word in the knowledge map to perform operations on the device, it will lead to failure of information extraction. The information extraction method of this paper achieves better results when the user follows the scene information.

TABLE VII
THE ACCURACY OF INFORMATION EXTRACTION

Type	Sentence Number	Right Back Number	Precision
DName	867	771	89%
LName	906	815	90%
CType	532	447	84%
Effect	766	612	80%

VII. CONCLUSION

To customize and develop smart home services more efficiently, an approach is proposed to model and execute services based on natural language instructions at runtime. Due to the gap between problem domains and system implementations, accomplishing mapping between them can create significant programming complexity. In order to solve the problem we introduce the knowledge graph serve as a bridge between system requirements and system implementation. Experimental results show that the proposed method can greatly reduce the difficulty and complexity of developing smart home services. But, due to the lack of technical and design capabilities there are some function implementation could be optimized. Firstly, the detection module [13] should be added to check the correctness and completeness of the instance model. Secondly, the information extraction method need to be improved by entity linking to increase the precision.

VIII. ACKNOWLEDGMENT

This work was supported by the National Key R&D Program of China under Grant No. 2017YFB1002000, the Talent Program of Fujian Province for Distinguished Young Scholars in Higher Education and the Guiding Project of Fujian Province under Grant No. 2018H0017.

REFERENCES

[1] K. Xu, X. L. Wang, W. Wei, H. B. Song, and B. Mao, Toward software defined smart home, IEEE Communications Magazine, 2016, 54(5):116-122.

[2] C. Dixon, R. Mahajan, S. Agarwal, A. J. Brush, B. Lee, S. Saroiu and P. Bahl. An operating system for the home. Usenix Conference on Networked Systems Design and Implementation, 2012:25-25.

[3] T. Edwards, J. Edwards, E. Dot. The Amazon Echo Dot User Guide: Newbie to Expert in 1 Hour! The Echo Dot User Manual That Should Have Come In the Box. Tim Edwards & Jenna Edwards.

[4] P. Dempesey. 2017. The teardown: Google Home personal assistant. Engineering & Technoloy 12, 3(Apr. 2017), 80-81.

[5] G. Campagna, R. Ramesh, S. Xu, F. Michael, S. L. Monica. Almond: The Architecture of an Open, Crowdsourced, Privacy-Preserving, Programmable Virtual Assistant. In International World Wide Web Conferences Steering Committee, 2017.

[6] L. Mottola and G. P. Picco. Programming wireless sensor networks:Fundamental concepts and state of the art. Acm Computing Surveys,2009,43(3):1-51.

[7] G. Y. Guan, W. Dong, Y. Gao, K. B. Fu and Z. H. Chen. TinyLink: A Holistic System for Rapid Development of IoT Applications. In International Conference on Mobile Computing and NETWORKING. New York: ACM, 2017:383-395.

[8] P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L. M. S. D. Souza and V. Trif. SOA-Based Integration of the Internet of Things in Enterprise Services. In IEEE International Conference on Web Services. 2009:968-975.

[9] K. Janowicz, A. Brring, C. Stasch, S. Sachade, T. Everding and A. Llaves. A RESTful Proxy and Data Model for Linked Sensor Data. International Journal of Digital Earth, 2013, 6(3):233-254.

[10] G. Huang, H. Song and H. Mei, SM@RT:towards architecture-based runtime management of Internetware systems, In Asia-paci?c Symposium on Internetware, pp.1-10, 2009.

[11] H. Song, G. Huang, F.Chauvel, Y. F. Xiong, Z. J. Hu, Y. C. Sun and H. Mei. Supporting runtime software architecture: A bidirectional transformation based approach. Journal of Systems & Software. 2011,85(5):711-723.

[12] Peking University. SM@RT: Supporting models at runtime. http://code.google.com/p/smatrt/, 2009.

[13] H. He, X. Chen, S. Cai, Y. Zhang, G. Huang. Testing bidirectional model transformation using metamorphic testing. Information and Software Technology, 2018, 104:109-129.