Towards Ontology-based Domain Specific Language for Internet of Things

Eman Negm
Faculty of Computer and Artificial Intelligence
Cairo University
Giza, Egypt
e.negm@fci-cu.edu.eg

Soha Makady
Faculty of Computer and Artificial Intelligence
Cairo University
Giza, Egypt
s.makady@fci-cu.edu.eg

Akram Salah
Faculty of Computer and Artificial Intelligence
Cairo University
Giza, Egypt
akram.salah@fci-cu.edu.eg

ABSTRACT
Development of Internet of Things (IoT) applications is considered as a complex task. It requires knowledge in the different software layers starting from the low level perception layer to the high level application layer. The domain expert should be involved from the start of the project to its end, to ensure that the delivered system satisfies the user needs. Such involvement results from the continuous need for the domain knowledge throughout the software development lifecycle. Such long development time along with the high cost of IoT applications, cause a slow progress in the IoT development. In this paper, a Domain Specific Language (DSL), called OntIoT, is proposed that contributes in reducing the complexity of IoT application development through providing the needed domain knowledge in an automated manner. OntIoT is an ontology-based DSL that utilizes the Semantic Sensor Network (SSN) ontology to catch the IoT domain concepts and constraints.

CCS Concepts
• Software and its engineering → Software notations and tools → Context specific languages → Domain specific languages.

Keywords
Internet of Things, Domain Specific Language, Ontology.

1. INTRODUCTION
Internet of Things (IoT) [1] is a term that refers to the interconnection of a set of things through a network. These things are able to collect and exchange data with each other. Things could be physical devices, virtual services, or anything. Sensors and actuators are examples of the physical devices that could be connected to the IoT applications. There are many successful IoT applications that promise great potential in enhancing the daily live like smart home and smart cities.

Although, IoT is a very promising area that could produce a new generation of software applications, the progress of the IoT development does not match the expected rate. CISCO provided a survey[1] in 2017 showing that only 26 percent of the companies could complete its IoT initiatives to be successful projects, and 60 percent of companies believe that the implementation of IoT applications is more complex than expected.

There are many reasons for the slow growth of IoT applications [2]. Different technical knowledge levels are required to build an IoT application. The required knowledge is varied from the low level embedded development and network protocols to the high level user friendly UI design. Across those levels, IoT development teams should focus on security, scalability, and other non-functional requirements that are mandatory for any IoT application. As a result, more development time and special skills are required to build IoT Applications. In addition, there is a lack of adequate languages and tool support for IoT programming [3].

Domain Specific Languages (DSL) are languages that provide high abstractions and optimizations for a specific domain. The problems in the given domain are represented more efficiently using the related DSL. A DSL for the IoT domain could contribute in solving some challenges of IoT development. The DSL will provide high abstractions related to the IoT domain like sensor, actuator, and observation concepts. Accordingly, the mapping between the business model of the IoT, held by the domain expert, and the technical model, held by the developer, becomes faster and more straightforward. As a result, the development time of the IoT applications will be reduced.

The DSL handles the generation of the low level code that decreases the required knowledge and increases the automated part. The DSL also provides a special editor that facilitates the development process and ensures that output program is matching the domain constraints. Finally, the involvement of domain expert becomes easier as the program is written with his terms.

The first phase for building a new DSL is the domain analysis phase which determines the concepts, relations, and constraints inside the concerned domain [4]. This phase is considered as a core phase in DSL development since a wrong analysis will lead to an invalid DSL even with a correct implementation. On the other hand, the ontology provides a formal conceptualization for a given domain that determines the concepts and the relations of this domain [5]. A lot of effort has been done to develop valid ontologies for IoT domain [6-7]. Accordingly, building a DSL from these ontologies will generate a valid DSL and it will reduce the DSL development time. In addition, the DSL could utilize existing ontology reasoning algorithms to provide semantic

ICSIE 2020, November 11–13, 2020, Cairo, Egypt
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7721-8/20/04…$15.00
DOI: https://doi.org/10.1145/3436829.3436833

[1] https://www.slideshare.net/CiscoBusinessInsights/journey-to-iot-value-76163389

reasoning services for the IoT editor. Finally, the existing documentation of the ontologies will guide the IoT developer to correctly model his problem using the corresponding DSL.

The main motivation of our work is to utilize the current standard IoT ontologies to build a DSL for IoT domain, this DSL benefits from the aforementioned advantages of ontology and DSL integration. This paper proposes OntIoT DSL that models part of the IoT domain. The structure of OntIoT is automatically generated from Semantic Sensor Network[2] (SSN) ontology [8]. The implementation approach of OntIoT is generic, it could be applied on any ontology or domain. Furthermore, OntIoT is based on projectional editing technique [9], which allows the language to be extended to support different editors and mixing among textual, tabular, and graphical notations. Moreover, the projectional editing technique allows OntIoT to support language extension and composition with other DSLs.

The rest of the paper is structured as follows. Section 2 illustrates some basics that our research depends on; Section 3 lists the related work; Section 4 shows a scenario that describes why do we need to use OntIoT?; Section 5 illustrates the approach that we follow to implement OntIoT DSL; Section 6 is devoted to the evaluation of OntIoT; and Section 7 concludes the paper.

2. RESEARCH BASICS

2.1 Domain Specific Language

General Purpose Languages (GPLs) are languages that are designed to solve problems in any domain. They do not contain specific domain abstractions or optimizations. Examples of GPLs are Java, Python, and C++. Domain Specific Languages (DSLs) are considered as an extension for GPLs. DSL is a language specialized in solving a specific class of problems which represents the domain of the language. Each DSL has a specific editor that provides editor services and error messages that are related to the concerned domain. Hyper Text Markup Language (HTML) is a DSL for representing web pages. Structured Query Language (SQL) is another DSL for relational databases.

Developing a new DSL includes five stages [4]: domain analysis, design, implementation, evaluation, and maintenance. Language workbench [9-10] is an environment for DSL development. It provides high level tools to support the previous stages. Language workbench aims to reduce the development time and complexity of building a new DSL. The DSL development process contains two main stockholders: First, the language designer who is responsible for developing the DSL through a language workbench. Second, the developer who utilizes the DSL editor created by the language designer to create concrete programs.

One of the promising techniques for implementing DSLs is the projectional editing technique. It is a parse-less technique that defines the structure of the language using meta-models not grammar rules. The Abstract Syntax Tree (AST) represents the core representation of the program that is stored and manipulated by the workbench. The developer edits the AST through a projection process that is defined by the language designer.

The core advantages of projectional editing technique are: First, it allows to mix different notations since it is a parser-less technique. Second, the language designer could define multiple projection processes for the same AST. Finally, it facilitates the language extension and composition since they are based on merging ASTs not grammar rules [11].

[2] https://www.w3.org/ns/ssn/

2.2 SSN Ontology

Ontology is a formal specification that is used to conceptualize a specific domain [5]. It consists mainly of four components: classes, instances, attributes, and relations. For instance, within the IoT domain, *Sensor* is a class that represents all sensors. *LM35* is an instance of class *Sensor*, it represents an individual sensor that measures temperature. Sensor may have attributes like *name*, *version,* and *observation*. Sensor has some relations with other objects, e.g., *Sensor* is a subclass of the *Device* class.

A well-known language for representing an ontology is the Ontology Web Language [12] (OWL). It a family of languages that are built by World Wide Web Consortium (W3C) based on the Description Logics [13]. Semantic Sensor Network (SSN) ontology is an ontology to model the domain of sensors, actuators, samplers and their observations. The current version of SSN ontology is developed by W3C and Open Geospatial Consortium's (OGC). SSN ontology contains a core ontology called SOSA (Sensor, Observation, Sample, and Actuator). SOSA follows one pattern to represent sensors, actuators, and samplers. Figure 1 describes this pattern by representing part of the SSN ontology that models the data of the following scenario:

Alice has a temperature sensor LM35 that measures the temperature of the kitchen. It follows analog procedure to measure the temperature. It measured value 37 at 12/12/2018 20:23:45.

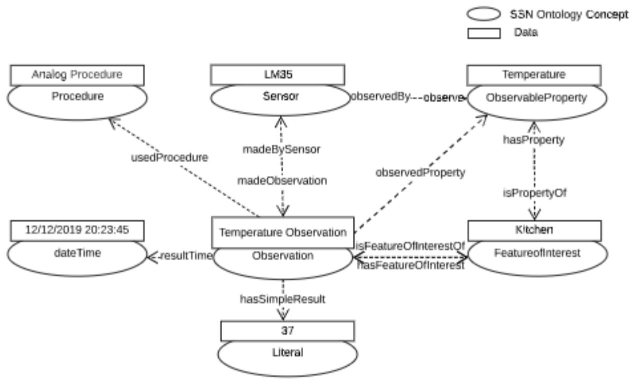


Figure 1. SSN Ontology Example

3. RELATED WORK

Our work combines three disciplines: IoT, DSL, and ontology. In the following sections, we will give a short survey about the current IoT DSLs and how ontology is currently used in the DSL development.

3.1 IOT DSLs

There is a set of DSLs that are proposed to model the IoT applications. Some of them provides textual notations and the others provide visual notations. DOS-IL [14] is a textual DSL for IoT with a lightweight interpreter. It enhances the scalability and extensibility of IoT applications since the development is done in an intermediate layer not in the target devices. Sneps-Sneppe and Namiot [15] propose another textual web based DSL that facilitates the integration of the data that are generated from the IoT devices and the web applications. IoTDsL [16] is also a textual DSL, it provides high level constructs to define the IoT devices and their network configuration. In addition, IoTDsL allows the user to define the business rules in a declarative way. CHARIOT [17] is a textual DSL that targets to enhance the extensibility of Cyber-Physical Systems.

DSL-4-IoT [18] is a visual domain specific modeling language. It enhances the programing complexity of IoT by providing a visual editor. Node-RED[3] is an open source visual editor that enables the designer to design IoT applications by connecting different devices, APIs, or online services. Eterovic et al. [19] propose a visual DSL that utilizes UML notations for allowing non-technical users to develop IoT applications. Midgar [20] is another visual DSL that allows non-technical users to connect IoT devices.

Up to our knowledge, no existing IoT DSL benefits from the current standard IoT ontologies. Our work is different from the previous work since OntIoT is an ontology-based DSL that is based on SSN ontology and projectional editing technique.

3.2 Using Ontology in DSL

DSLs and ontologies are utilized to model a specific domain. Consequently, there are a lot of work to utilize the ontology in enhancing the DSL development process [21]. Čeh et al. [22] propose Ontology2DSL framework that utilizes the ontology in the DSL design phase to model the domain. Then, it uses the given ontology to automatically generate the grammar of the DSL in the implementation phase. Similar work is presented by Pereira el al. [23] with more enhancements in the generation process and the output grammar. Walter et al. [24] integrate the ontology and the domain specific modeling in OMG four layered architecture [25] to enhance the DSL tooling and the learning curve. Ulitin et al. [26] focus on using ontology in DSL evolution. Another approach for ontology and DSL alignment is proposed by Ojamaa et al. [27]. They provide automated generation of design templates for DSL meta-models based on a domain ontology.

None of the previous work studies the merging of the ontology with the projectional editing DSL implementation technique. The structure of OntIoT is automatically generated from the mapping between the ontology and the AST which represents the core unit in the projectional editing.

4. OntIoT USAGE SCENARIO

OntIoT is a DSL that is specialized to model problems in the IoT domain using SSN ontology. Consider the following scenarios from the language designer and the developer perspectives:

Bob is a DSL designer in DesignBoost company. He has been assigned to design a DSL to help developers build IoT applications.

As Bob is not an expert in the IoT domain concepts that need to be included within the target DSL, he decides to gather knowledge from various resources to come up with a comprehensive DSL. By comprehensive, Bob aims for a DSL that would cover all the IoT domain concepts that a developer would need when building an IoT application. As covering all IoT concepts is such a challenging task, Bob attempts to build OntIoT DSL in a generic manner, so that such a DSL would extract the IoT domain concepts automatically from an existing ontology. Such an automated approach for concepts extract would give Bob the opportunity to release newer versions of OntIoT easily whenever new IoT concepts, and hence IoT ontologies arise.

Alice is a developer in another company, she has been assigned to implement new business requirements for building a set of IoT applications. *Alice* does not have a good knowledge about IoT. In addition, she is required to finalize the implementation in a very limited time.

Alice has two options to solve the above situation. First option (Figure 2) is to use one of the GPLs like C++ and Java to implement the new requirements. In this option, he does not have to learn new programming language. However, he has to gain a detailed knowledge about the different layers of IoT development from the perception layer to the application layer. Additionally, *Alice* needs a lot of time to model the required IoT applications using the selected GPL which is not valid in this scenario.

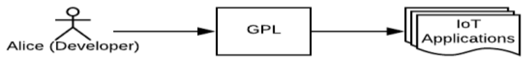


Figure 2. Using General Purpose Language (Option 1)

Second option (Figure 3) is to use one of the existing DSLs that models the IoT domain. This option will hide a lot of IoT development complexity by providing high abstractions which facilitate the modeling of the required IoT applications. In addition, the DSL will provide *Alice* with a specialized editor and automatic generation for the low level layers. However, in this option, *Alice* needs some time to learn the new DSL which is not required in the first option.

Finally, *Alice* decided to choose the second option to satisfy her knowledge and time constraints. The next decision is which IoT DSL to use. *Alice* decided to use OntIoT for many reasons: It is based on SSN standard ontology which is a high level ontology that could model the different required IoT applications. Furthermore, OntIoT can be extended with emerging IoT domain concepts in an automated manner through updated IoT ontologies. Moreover, the constructs of OntIoT DSL is mapped to SSN ontology which is well documented and meaningful for him. In addition, the OntIoT editor is a user friendly.

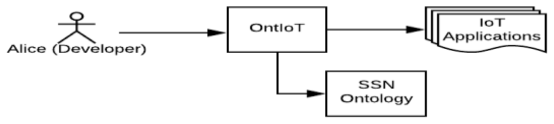


Figure 3. Using OntIoT DSL (Option 2)

5. OntIoT IMPLEMENTATION

In this section, we will describe the approach that is used to implement OntIoT. Figure 4 describes the steps for building OntIoT and its editor from the SSN ontology. OWL is used to represent the SSN ontology. The transformation engine is responsible for transforming the OWL SSN ontology to the target language structure based on a set of general mapping rules (Section 5.1). The input for the engine is the SSN OWL file and the output is the abstract syntax tree for the OntIoT language. The latter is used inside the language workbench to generate the final DSL and its editor by the language designer.

[3] https://nodered.org/

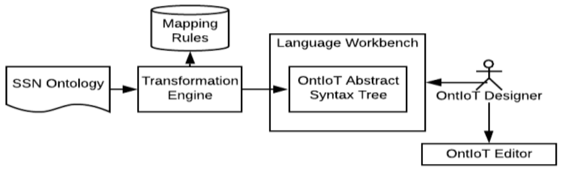


Figure 4. OntIoT Implmentation Approach

Our ontology-based approach for building OntIoT is a generic approach that can build a DSL from any OWL ontology. SSN ontology is selected to build OntIoT since it is a W3C and OGC standard. In addition, it is a high level ontology that could be used in a wide range of IoT applications. Accordingly, OntIoT could be used in many IoT sub domains like smart home, smart healthcare, and agriculture monitoring. SSN ontology is considered as a core ontology for a set of more specific ontologies like IoT-Lite[4] and OpenIoT[5]. The extension of OntIoT to cover these ontologies is a straightforward using our generic approach without need to rebuild the language.

In our implementation, we used Meta Programming System[6] (MPS) language workbench for building OntIoT. MPS is an open source workbench that implements the projectional editing technique for DSL development. The projectional editing technique is selected to implement OntIoT for many reasons: First, OntIoT is based on SSN ontology. There are many similarities between the ontology structure and the AST structure. The AST is the core unit in the projectional editing. These similarities facilitate the transformation process. Second, the projectional editing technique will allow to build different editors types for OntIoT (e.x. textual and graphical editors). Currently, we provide only a textual editor but we plan to build a graphical one. Third, It allows OntIoT to contain mixed notations (i.e. textual, symbols, and tabular) which is not applicable in other techniques. Finally, projectional editing technique allows OntIoT to be easily extended to map the different ontologies that utilize or extend SSN ontology.

5.1 OWL - DSL Mapping

In this section, we will describe the mapping rules used by the transformation engine to transform the SSN ontology into the AST of OntIoT DSL. This mapping is a generic mapping that is not restricted to SSN ontology or OntIoT DSL.

OWL ontology represents the domain by a set of classes, instances, properties, and relations. In addition, it may contain a set of restrictions on the above entities. On the other hand, AST contains a set of nodes, attributes, and references. The definition of AST in MPS language workbench is very similar to object oriented programming. Each node is defined by a concept (i.e. class). The attributes of the node are defined as concept attributes. The references are links among the defined concepts.

The classes in the OWL ontology are mapped to the MPS concepts. The superclass and subclass relations in OWL are implemented in workbench using concept extension. The OWL primitive properties are mapped to concept attributes. And the OWL relations are represented as concept links.

[4] http://iot.ee.surrey.ac.uk/fiware/ontologies/iot-lite#

[5] http://sensormeasurement.appspot.com/ont/sensor/openIoT.owl

[6] http://www.jetbrains.com/mps/

5.1.1 Mapping Challenges

The mapping between OWL and AST has a set of challenges that should be resolved to complete the mapping, below a list of challenges and the workaround done by OntIoT to resolve it.

- The definition of AST in MPS does not support multiple inheritance, whereas OWL superclass/subclass relation allows this. Currently, OntIoT selects a random parent in case of more than one parent exists.
- OWL allows the same link present in the parent, and the child to have a different target which is not valid in MPS. Accordingly, a manual feedback is needed to resolve this issue.
- AST should have a root concept (a topmost node in the tree) which is not defined by the OWL ontology. OntIoT automatically generate a root concept called "*IoTScript*" and a manual action is done to add links among the root concept and other concepts.
- Not all OWL restriction types are applicable to be implemented using MPS AST. MPS only supports cardinality (0-1), (1-n), (0-n), and (1). OntIoT supports *AllValuesFrom*, *MinCardinality* with minimum value zero or one only, *MaxCardinality* and *Cardinality* with max value one only. We are working in adding the missing restriction mapping by adding constraints on the AST.

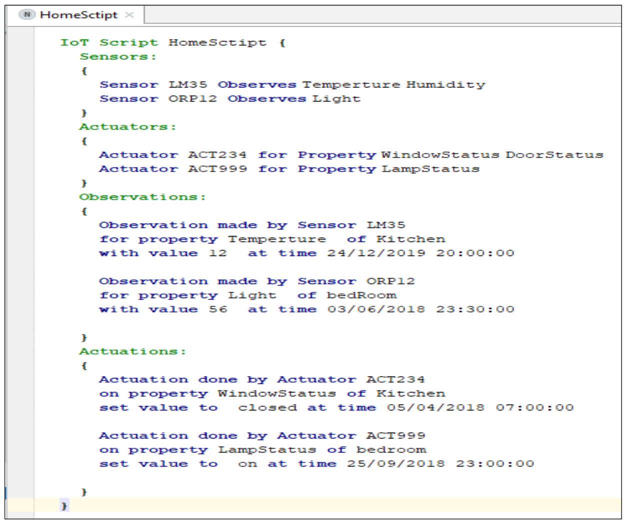
6. OntIoT EVALUATION

Table 1 lists the number of concepts, restrictions, and relations of SSN ontology that are covered in the current structure of OntIoT. All SSN ontology classes are covered by OntIoT. Class restrictions on data type properties are fully mapped to OntIoT properties. OntIoT covers 86.8% of SSN restrictions on object properties. The missing restrictions are the restrictions on the inverse of a property not the property itself which is not covered by the current mapping. OntIoT covers 85.7% of the class hierarchy in SSN. The lost relation are due to the multiple inheritance which is not supported in OntIoT up to know. Table 1 shows that OntIoT successfully covers most of the SSN objects and relations. Consequently, OntIoT DSL allows the developer to write IoT programs that match most of the concepts and the constraints of the SSN ontology.

Table 1 SSN objects covered by OntIoT DSL

Object		SSN	OntIoT
Class		22	22
Restriction	Data Type Property	3	3
	Object Property	53	46
Subclass / Superclass Relation		7	6

Figure 5 shows the editor of OntIoT with a sample code that defines sensors, actuators, actuations, and observations. The code defines a new IoT Script called ***HomeScript*** (the auto generated root concept). The script contains four groups. First, the ***Sensors*** group includes the definition of the sensors. Sensor LM35 observes two properties temperature and humidity. Second, the ***Actuators*** group defines the actuators. Actuator ACT234 changes window status and door status properties. Third, the ***Observations*** group defines the observations made by the above sensors. LM35 measured value 12 at 24/12/2019 20:00:00 for the temperature of the kitchen. Finally, the ***Actuations*** group describes the actions done by the previously defined actuators. Actuator ACT99 changed the status of the window in the kitchen to closed at 25/04/2018 07:00:00:00.



```
IoT Script HomeSctipt {
  Sensors:
  {
    Sensor LM35 Observes Temperture Humidity
    Sensor ORP12 Observes Light
  }
  Actuators:
  {
    Actuator ACT234 for Property WindowStatus DoorStatus
    Actuator ACT999 for Property LampStatus
  }
  Observations:
  {
    Observation made by Sensor LM35
    for property Temperture  of Kitchen
    with value 12  at time 24/12/2019 20:00:00

    Observation made by Sensor ORP12
    for property Light  of bedRoom
    with value 56  at time 03/06/2018 23:30:00

  }
  Actuations:
  {
    Actuation done by Actuator ACT234
    on property WindowStatus of Kitchen
    set value to  closed at time 05/04/2018 07:00:00

    Actuation done by Actuator ACT999
    on property LampStatus of bedroom
    set value to  on at time 25/09/2018 23:00:00

  }
}
```

Figure 5. OntIoT Editor

7. CONCLUSION

In this paper, we propose a DSL for IoT domain, called OntIoT, which is based on the SSN ontology. The approach, used to build OntIoT, is generic that could be applied to build a DSL from any ontology. OntIoT is based on projectional editing technique.

Our ongoing research is to evaluate the proposed DSL in a real scenario. We plan to complete the mapping rules between OWL and DSL to cover the missing OWL constructs. Furthermore, we plan to utilize OWL reasoning algorithms to enrich the OntIoT editor with more semantic features. In addition, a graphical editor will be implemented for the proposed DSL.

8. REFERENCES

[1] ASHTON, K., 2009. That 'internet of things' thing. RFID journal 22, 7, 97-114.

[2] ČOLAKOVIĆ, A. and HADŽIALIĆ, M., 2018. Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues. Computer Networks 144, 17-39.

[3] TAIVALSAARI, A. and MIKKONEN, T., 2017. A roadmap to the programmable world: software challenges in the IoT era. IEEE Software 34, 1, 72-80.

[4] MERNIK, M., HEERING, J., and SLOANE, A. M., 2005. When and how to develop domain-specific languages. ACM computing surveys (CSUR) 37, 4, 316-344.

[5] GRUBER, T. R., 1995. Toward principles for the design of ontologies used for knowledge sharing? International journal of human-computer studies 43, 5-6, 907-928.

[6] BAJAJ, G., AGARWAL, R., SINGH, P., GEORGANTAS, N., and ISSARNY, V., 2017. A study of existing Ontologies in the IoT-domain. arXiv preprint arXiv:1707.00112.

[7] SZILAGYI, I. and WIRA, P., 2016. Ontologies and Semantic Web for the Internet of Things-a survey. In IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society IEEE, 6949-6954.

[8] HALLER, A., JANOWICZ, K., COX, S. J., LEFRANÇOIS, M., TAYLOR, K., LE PHUOC, D., LIEBERMAN, J., GARCÍA-CASTRO, R., ATKINSON, R., and STADLER, C., 2019. The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation. Semantic Web, Preprint, 1-24.

[9] FOWLER, M., 2005. Language workbenches: The killer-app for domain specific languages.

[10] ERDWEG, S., VAN DER STORM, T., VÖLTER, M., BOERSMA, M., BOSMAN, R., COOK, W. R., GERRITSEN, A., HULSHOUT, A., KELLY, S., and LOH, A., 20. The state of the art in language workbenches. In International Conference on Software Language Engineering Springer, 197-217.

[11] VOELTER, M. and SOLOMATOV, K., 2010. Language modularization and composition with projectional language workbenches illustrated with MPS. Software Language Engineering, SLE 16, 3.

[12] MCGUINNESS, D. L. and VAN HARMELEN, F., 2004. OWL web ontology language overview. W3C recommendation 10, 10, 2004.

[13] BAADER, F., CALVANESE, D., MCGUINNESS, D., PATEL-SCHNEIDER, P., and NARDI, D., 2003. The description logic handbook: Theory, implementation and applications. Cambridge university press.

[14] NEGASH, B., WESTERLUND, T., RAHMANI, A. M., LILJEBERG, P., and TENHUNEN, H., 2017. DoS-IL: A domain specific Internet of Things language for resource constrained devices. Procedia Computer Science 109, 416-423 %@ 1877-0509.

[15] SNEPS-SNEPPE, M. and NAMIOT, D., 2015. On web-based domain-specific language for internet of things. In 2015 7th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT) IEEE, 287-292.

[16] AMRANI, M., GILSON, F., DEBIECHE, A., and ENGLEBERT, V., 2017. Towards User-centric DSLs to Manage IoT Systems. In MODELSWARD, 569-576.

[17] PRADHAN, S. M., DUBEY, A., GOKHALE, A., and LEHOFER, M., 2015. Chariot: A domain specific language for extensible cyber-physical systems. In Proceedings of the workshop on domain-specific modeling ACM, 9-16.

[18] SALIHBEGOVIC, A., ETEROVIC, T., KALJIC, E., and RIBIC, S., 2015. Design of a domain specific language and IDE for Internet of things applications. In 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) IEEE, 996-1001.

[19] ETEROVIC, T., KALJIC, E., DONKO, D., SALIHBEGOVIC, A., and RIBIC, S., 2015. An Internet of Things visual domain specific modeling language based on UML. In 2015 XXV International Conference on Information, Communication and Automation Technologies (ICAT) IEEE, 1-5.

[20] GARCÍA, C. G., ESPADA, J. P., VALDEZ, E. R. N., and DÍAZ, V. G., 2014. Midgar: Domain-specific language to generate smart objects for an internet of things platform. In 2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing IEEE, 352-357.

[21] SUTII, A., VERHOEFF, T., and VAN DEN BRAND, M., 2014. Ontologies in domain specific languages: a systematic literature review. Computer science reports 1409.

[22] ČEH, I., ČREPINŠEK, M., KOSAR, T., and MERNIK, M., 2011. Ontology driven development of domain-specific

languages. Computer Science and Information Systems 8, 2, 317-342.

[23] PEREIRA, M. J. V., FONSECA, J., and HENRIQUES, P. R., 2016. Ontological approach for DSL development. Computer Languages, Systems & Structures 45, 35-52.

[24] WALTER, T., PARREIRAS, F. S., and STAAB, S., 2014. An ontology-based framework for domain-specific modeling. Software & Systems Modeling 13, 1, 83-108.

[25] OMG, UML 2.0 Infrastructure Specifications. In OMG Document formal/05-07-05.

[26] ULITIN, B., BABKIN, E., and BABKINA, T., 2018. Ontology-based DSL development using graph transformations methods. Journal of Systems Integration 9, 2, 37-51.

[27] OJAMAA, A., HAAV, H.-M., and PENJAM, J., 2015. Semi-automated generation of DSL meta models from formal domain ontologies. In Model and Data Engineering Springer, 3-15.