

# A Meta-Modeling Approach to Describe Internet of Things Architectures

Abdessamad Saidi<sup>a</sup>, Mohamed Hadj Kacem<sup>a</sup>, Imen Tounsi<sup>a</sup> and Ahmed Hadj Kacem<sup>a</sup>

<sup>a</sup>University of Sfax, ReDCAD Research Laboratory, Sfax, Tunisia

## Abstract

Statistics say that there will be 25 billion devices interconnected to the Internet by the end of 2021. This explosive growth of connected objects has led us to the advent of a promising technology that can satisfy this need known as the Internet of Things (IoT). IoT enables digitalization around the world, but it also increases the challenges for implementing an IoT solution. Since there are no specific standards for the Internet of Things, there can be conflicts, ambiguities that can lead to poor implementation. The adequate software architecture that can be applied to Internet of Things is service-oriented architecture. SOA offers flexibility and weak coupling and allows the reuse of IoT services in such an IoT system. Our main objective is to provide a formal description of the IoT software architecture. In this work, we propose a solution based on design models for the Internet of Things with semi-formal notation using OMG's (Object Management Group) standard, the UML modeling language.

## Keywords

UML, Internet of Things, Design Pattern, Model, Meta-model, Software Architecture

## 1. Introduction

Software architecture allows us to understand the system and reason about its properties by providing a high-level model of its structure. Software and digitization have pervaded everything. The success in the competitiveness of businesses depends on the performance of software systems. The Internet of Things provides to potential to connect everything, makes machines talk to other machines (M2M) and collects data about them with a continuous real-time data stream. The complexity of software systems has many different dimensions. The software systems for the Internet of Things applications tend to have a sense of each of the complex dimensions: IoT software has a distributed and real-time nature, it interoperates with many external systems and processes large volumes of data. IoT applications are also typically built by large multidisciplinary global teams. There are many challenges when building IoT systems like interconnecting devices, communication protocols, security and power supply constraints. In software design, a Design Pattern is a reusable solution to a common problem in a specific environment. In our research study, we focused on Design Patterns solutions for software

---

*Tunisian Algerian Conference on Applied Computing (TACC 2021), December 18–20, 2021, Tabarka, Tunisia*

✉ [abdessamad.saidi@redcad.org](mailto:abdessamad.saidi@redcad.org) (A. Saidi); [mohamed.hadjkacem@isimsf.rnu.tn](mailto:mohamed.hadjkacem@isimsf.rnu.tn) (M. Hadj Kacem);


[imen.tounsi@redcad.org](mailto:imen.tounsi@redcad.org) (I. Tounsi); [ahmed.hadjkacem@fsegs.rnu.tn](mailto:ahmed.hadjkacem@fsegs.rnu.tn) (A. Hadj Kacem)

🌐 <https://www.redcad.org/members/hadjkacem/> (M. Hadj Kacem);

<https://www.redcad.org/members/imen.tounsi/> (I. Tounsi); <https://www.redcad.org/members/ahadjkacem/> (A. Hadj Kacem)



© 2020 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

IoT systems. To face these challenges, a set of Design Patterns was proposed in [1] to solve some problems. However, these models are represented with natural language and graphical notations. This representation is classified as an informal modeling. On the other hand, the use of informal language makes the modeling imprecise and sometimes ambiguous.

In this paper, we propose an approach that permit to describe IoT architecture in order to develop IoT application by providing tool, meta-models and models.

Our approach has a number of advantages: first, the approach aims to capitalize on the expressiveness of UML 2.5 standard visual notations. Second, the proposed meta-models are defined in a high level of abstraction that can be reused. Third, we have strengthened our approach by using IoT oriented design patterns in order to have high-performance applications while guaranteeing some quality attributes. Last but not least, an eclipse plug-in to assist architect in the modeling phase.

The remainder of the paper is structured as follows: we will present and discuss related works in Section 2. In Section 3, we model the structural features of IoT architecture and the behavioral features in Section 4. In Section 5, we will present an IoT application to a case study of our proposed approach. We will give a part of source code that permit to implement our use case in Section 6. Section 7 presents our tool, which implements the proposed approach. We conclude and outline some future work in Section 8.

## 2. Related works

The arrival of the Internet of Things has raised many challenges for architects and designers of complex systems. In order to assist and orient the architects, several modeling and design solutions were proposed such as tools, UML (Unified Modeling Language) profiles and Design Patterns. Several works have made specific UML extensions for the IoT [1][2][3]. Among these profiles, there are those which models the security mechanisms [4], the other allows modeling the IoT applications [5]. In another work [6], authors have created a tool which offers the possibility of modeling the architecture of an IoT system using models as well as verifying these models using a logic programming language.

In this paper, we are interested in the Design Patterns since they offer a reusable and flexible design. There are several types of Design Patterns, such as Service Oriented Architecture (SOA) patterns. The authors in [7][8] propose a complete approach which makes it possible to guide the architect from the phase of modeling SOA (semi-formal) patterns to the phase of their composition via a step of transforming patterns in order to have formal semantics. The approach uses the SoaML modeling language and the Event-B method based on transformation rules for the specification of SOA Design Patterns.

In this paper, we are interested in dedicated Design Patterns for the Internet of Things. Most of the Internet of Things patterns that exist are at the Cloud level. In [9], the authors help IoT system architects to build Edge applications by proposing four IoT Design Patterns that facilitate some tasks like deployment.

In [10][11][12][13], the authors cited 48 Internet of Things patterns, and then they listed it in 10 categories (Communication, Security, Bootstrapping, Registration, etc.). However, the problem is that all of these IoT patterns are expressed informally. Our approach makes it possible to

model part of the IoT Design Patterns found in the communication category proposed in [14]. Another work, the authors in [15] made a survey on existing both IoT Design and architecture Pattern. These Design and architecture Pattern are classified in three levels (Architecture style, Architecture pattern and design pattern) and applicable in three field (not dedicate for IoT, for general IoT use case and for a particular domain). We notice that the majority of architectural pattern are used in a specific IoT domain. Moreover, we find that most of the design patterns oriented for IoT are appropriate for general IoT use case. In the other side, we don't find a lot of IoT architectural style. In another research, Hachicha et al. [16] propose a UML profile that extend UML diagram in order to model behavioral and structural aspects of MAPE (Monitoring, Analysis, Planning, Execution) patterns. In a recent work [17], authors propose an approach for self-adaptive IoT systems using formal specification and the composition of MAPE patterns. In previous work, authors propose models and meta-models that combine IoT and SOA or IoT and SoS domains, they did not take into consideration the quality attributes which are important in complex systems, some works don't mention the type of exchanged messages and do not provide a modeling tool to help developers.

In this paper, we provide an approach oriented IoT that take into consideration the quality attributes, behavioural and structural views of the IoT System, a modelling tool to assists architects and specify the types of messages.

### 3. Modeling the Structural View

In this section, we give a solution to illustrate the Internet of Things architecture by employing a visual notation supported by UML language to offer comprehensible models.

As a first step, we define a Meta-model, and the next step is to instance conform models by extending UML diagrams. UML grant to express structural and behavioral views of IoT system. In our approach, we use a sequence diagram to represent behavioral aspects and the component diagram for the structural aspects of IoT architecture. We use a case study to illustrate these models.

#### 3.1. The Proposed approach

Our approach is based on the Internet of Things architecture. Figure 1 illustrates the most important components that we found in IoT architecture which are: Sensors that collect data from the environment and actuators that receive commands from the Cloud in order to react, IoT Gateway is the intermediate between other components. In the IoT cloud platform layer , we do the necessary processing, analytic, storage, and device management to offer users applications. In our approach, we have proposed three meta-models:

1. IoTStakeholder meta-model
2. IoTCommunication meta-model
3. MessageType meta-model

In follows, we details each meta-motel.

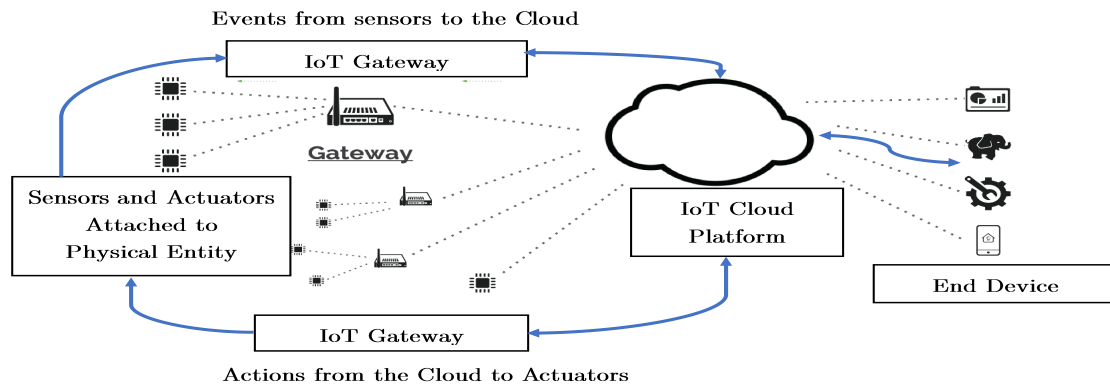


Figure 1: A basic Internet of Things architecture.

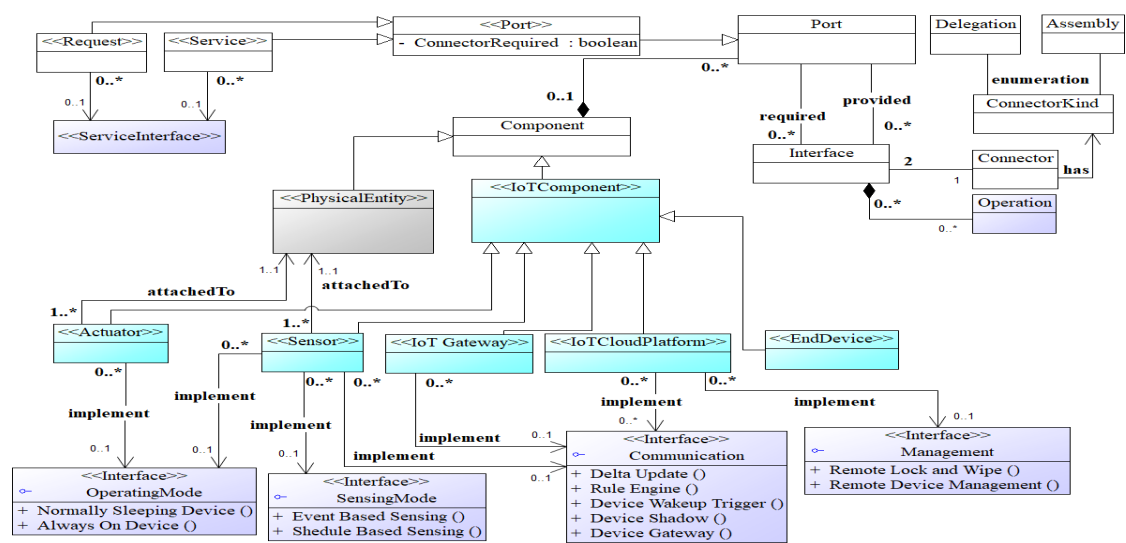


Figure 2: Meta-models: IoTStakeholder and IoTCommunication

### 3.2. IoTStakeholder meta-model

The IoTStakeholder meta-model describes the major components of an IoT system. Blocks with **Cyan color** in Figure 2 represents the proposed Meta-model in a high level of abstraction.

Our meta-model is based on the UML component diagram. The basic elements of our meta-model are:

- **IoTComponent**: The elements that form the architecture of an IoT system are: sensors, actuators, gateways, cloud and user interface.
- **Sensor**: allows data to be collected and sent to IoT Cloud platform.
- **Actuator**: reacts according to the decisions taken after the processing done at the Cloud level.
- **IoTGateway**: Sensors and Actuators are linked to Cloud by the IoTGateway.

- **IoT Cloud Platform:** The processing, storage, services and standards required for connecting, managing, and protecting various IoT devices and applications are included in an IoT cloud Platform.
- **EndDevice:** allows you to view the data collected by the Sensors as well as the state of actuators.
- **Port:** Each IoT component has a port that represents the point of interaction with the other components. There are two types of port:
  1. A port which makes it possible to offer a service <<service>>
  2. A port that expresses the need for a service <<request>>
- **PhysicalEntity:** represents the object that we want to make it connected or intelligent.
- **Interface:** communications with the environment are made through both types of interface. The Provided interface and the Required interface.

### 3.3. IoTCommunication meta-model

The Figure 2 shows the IoTCommunication meta-model. Based on the IoTStakeholder meta-model, we added three elements which are colored with **purple color**.

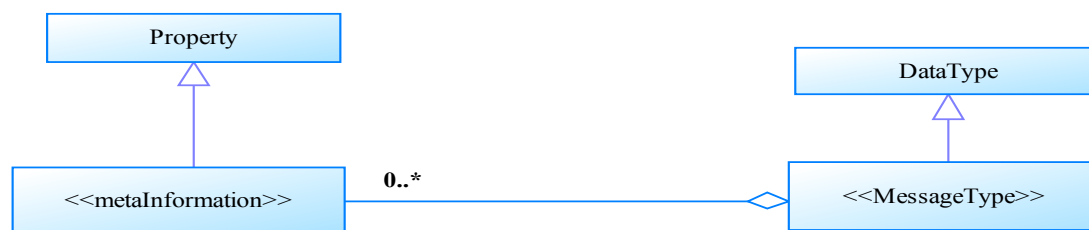
1. **ServiceInterfaces:** are used to explicitly model provided and required operations.
2. **Operation:** describes the operations provided by an interface.
3. **Design Pattern:** based on the work [14], we added the design patterns as interfaces to implement, they allow us to improve the quality attributes in complex architecture such as IoT system.

For example, Delta Update which is a communication design pattern guarantee these qualities attributes:

1. **Message Size:** As a result, messages are now smaller and don't contain any extraneous data. They only exclude information that hasn't changed since the previous time they communicated.
2. **Bandwidth:** With a smaller message, less bandwidth is used.
3. **Energy Consumption:** Communication consumes most energy. Due to the fact that updates are smaller, devices may deliver them more quickly. It also means that devices may turn off their connectivity modules for longer duration's, which reduces their energy usage.

### 3.4. MessageType meta-model

In any system, there are a lot of messages going around. For that we must define the type of each message in order to be able to differentiate between the messages exchanged by the different entities of the system. The MessageType meta-model contains the elements that support data modeling. It describes the type of messages exchanged between the different entities of IoT architecture. Its structure is described by the Figure 3. The <<MessageType>> is used to explicitly identify the data elements exchanged between the components of the system. It extends the DataType metaclass. An <<Attachment>> is a part attached to the message. It extends the Property metaclass.



**Figure 3:** Message Type meta-model

## 4. Modeling the Behavioral View

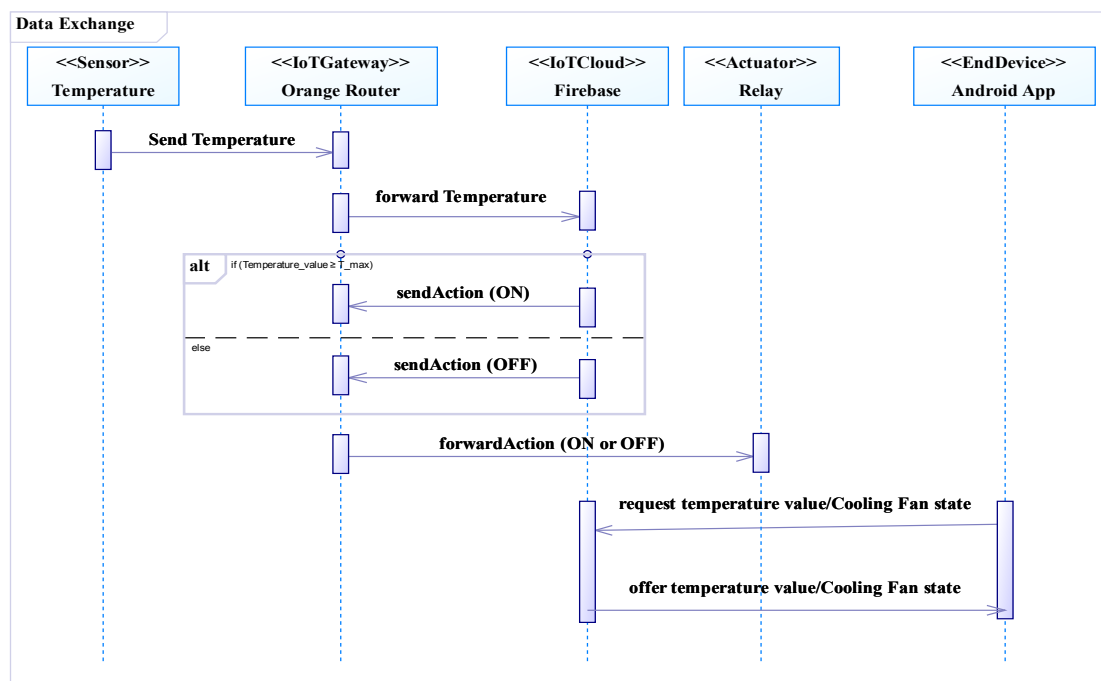
In this section, in order to model the behavioral properties of the IoT system we use the part of the meta-model of the sequence diagram of the UML 2.5 version. This diagram allows us to illustrate the different reciprocal actions between the different components of an IoT system. Figure 4 illustrates a sequence diagram of our case study, it shows the exchanged messages between the major components of IoT architecture. The use case consist on collecting the temperature of the room with a Sensor and sending it to the IoT Cloud Platform through the IoT Gateway. In the IoT Cloud Platform, we process the temperature value in order to decide how to react using the Actuator. In the other side, we have a mobile application that permit to see the temperature of the room and the status of the cooling fan in real-time.

## 5. Use Case: A Smart room service: an automatic ventilation system

We choose a smart house as IoT domain application. The Internet of Things has made the world utterly digital by interconnecting the physical world with the digital world. The smart house is a representation of this technology, and these houses are equipped with sensors and actuators to control everything that exists in the house. The cooperation of these components makes it possible to offer services. In our case, we took a part of the smart house system which is smart room to make experimentation in order to validate our approach. The system is composed by four layers witch is compatible with the general IoT architecture with the aim of having a scalable smart environment that sustain a variety of services.

The used software and hardware components:

- **The physical layer:** of the application is composed by a temperature sensor, it permits to sense the temperature of the room and it is attached to a wall and a relay that plays the role of an actuator and it is linked to the cooling fan, the actuator can turn on and off the cooling fan. Both Sensor and Actuator are connected to an ESP8266 and it is programmed with embedded C language and support Wi-Fi technology.
- **The processing layer:** contain the IoT Cloud platform, it permits to do the necessary processing, storage and management. If the temperature value is greater than a predefined value (T\_MAX) we turn the cooling fan ON else OFF. We choose Firebase as IoT cloud



**Figure 4:** Sequence diagram of the modeled system

platform because it offers a real-time database, free with big community and offer APIs that help in the implementation.

- **The network layer:** is a gateway that it is an intermediary between the physical layer and the processing layer. We opt for Wi-Fi as communication technology because it is supported by a lot of devices. In our system, we have an Orange router.
- **The application layer:** through this layer, we can visualize the data collected by the sensor and the state of the actuator. In our use case, we develop an Android application.

Now, we will instantiate models from the proposed meta-models in order to model our case study. As shown in the Figure 5, we have specified the type of port for each component of the system as well as the operations offered by each interface. Next, we will instance the MessageType diagram, it is used to model the messages exchanged between the different entities of the system. The diagram of MessageType is shown in Figure 6. There are five MessageType which are used to define all the information exchanged. Our implemented system respects the three dimensions of IoT:

1. Any time connection: We can visualize the data in real-time at any time (night, daytime ...)
2. Any place connection: We can see the data in any place in the world.
3. Anything connection: We are able to make the Cooling Fan as a connected thing.

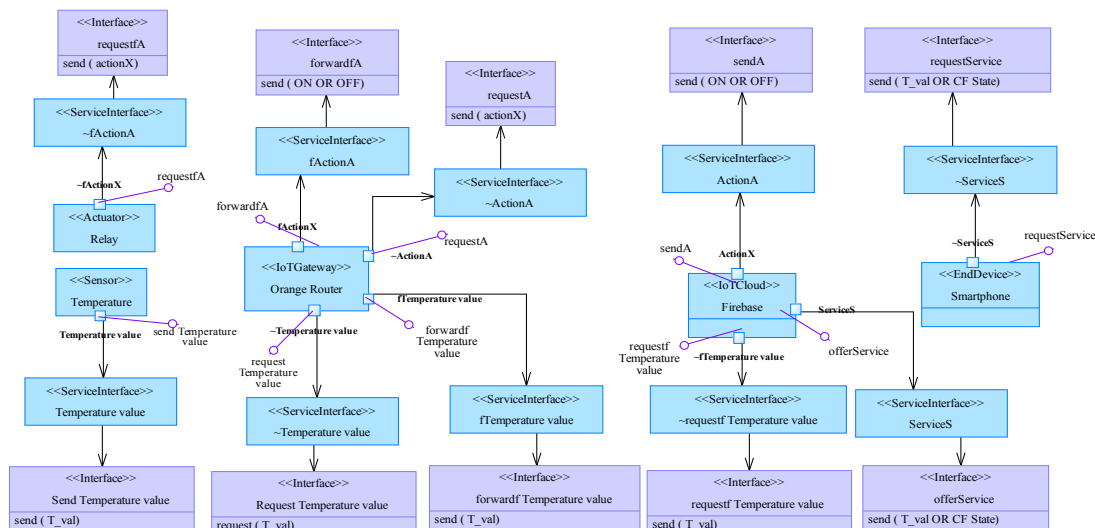


Figure 5: IoTCommunication model

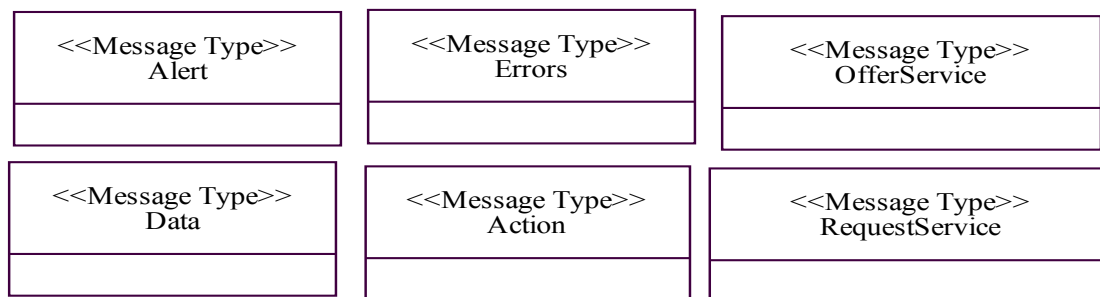


Figure 6: MessageType model

## 6. Implementation

In this Section, we present some important lines of the source code of the Embedded part. In Listing 1, line 1 indicates the library of Firebase Cloud Platform, it contains functions that permit to connect to the Database, to read and write the data. Line 2 represents a library to use the WiFi module of the Micro-controller (NODE MCU) and the library of DHT 11 sensor in line 3 which offers two functions to capture the temperature and humidity of the environment. In lines 4 and 5, we have the link of the Database and the authentication key. The Service Set Identifier (SSID) and the password of the router in lines 6 and 7 respectively.



Listing 1: Sketch of source code 1

```
1 #include "FirebaseESP8266.h"
2 #include <ESP8266WiFi.h>
3 #include <DHT.h>
4 #define FIREBASE_HOST "smartroom-ee16e-default-rtdb.firebaseio.com"
5 #define FIREBASE_AUTH "W2C5EfElHzgbGlauyznQBu0A9XhJBSd6Pr4VdBMZ"
6 #define WIFI_SSID "Fixbox_01"
7 #define WIFI_PASSWORD "IoT"
```

Listing 2 represents the initialisation of temperature sensor. We define the type of DHT sensor (DHT 11 or DHT 22) in line 1 and we start the sensor with the instruction mentioned in line 2. Finally, we collect the temperature value of the room with the instruction in line 3.

Listing 2: Sketch of source code 2

```
1 DHT dht(DHTPIN, DHTTYPE);
2 dht.begin();
3 h = dht.readTemperature();
```

Now, we give the implementation of the actuator and how to interact with the Database. The line 1 of Listing 3 permits to store the temperature value and the second line permits to read the temperature value from the Database.

From line 3 to 9 of Listing 3, we compare the temperature value with a constant, if it is hot we turn the cooling Fan ON and make its status ON in the Cloud Platform otherwise the cooling Fan turns OFF and the status will be updated to OFF in the Cloud Platform.

Listing 3: Sketch of source code 3

```
1 Firebase.setFloat(firebaseData, "Room/Temperature", t);
2 float T1 = Firebase.getFloat(fbdo, "Room/Temperature") ? fbdo.floatData():0.0;
3 if (T1 >= T_MAX ){
4     digitalWrite(DCMPIN, 0);
5     Firebase.setString(fbdo,"Room/Status", "ON");
6 }
7     else {
8     digitalWrite(DCMPIN, 1);
9     Firebase.setString(fbdo,"Room/Status", "OFF");
10 }
```

In the processing layer, Figure 7 shows the real time database of Firebase Cloud Platform. It contains two parts: the first one contains the collected data (the temperature of the room) as well as the state of Cooling Fan and the second part shows the users who have access to the database.

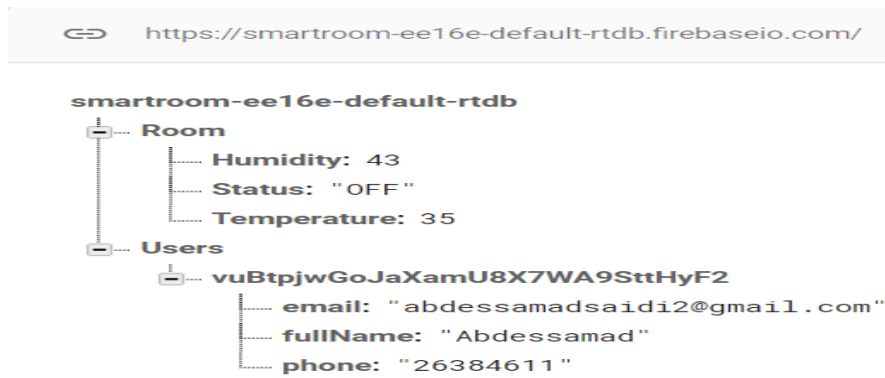


Figure 7: The Real Time Database

## 7. Tool Modeling

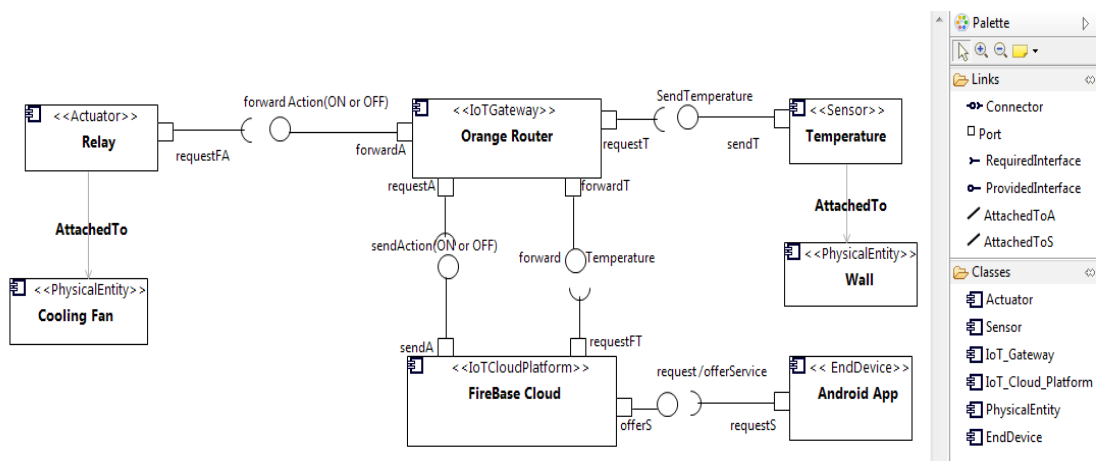


Figure 8: The Modeling Tool

In order to create a graphical editor with Eclipse IDE, we need to use Eclipse Modeling Framework (to create the meta-model) and Graphical Modeling Framework (to generate the graphical editor) frameworks. Our Eclipse plug-in supports our methodology and allows users to model in a simple and efficient manner. By using the Eclipse plug-in that we recommend, the architect can model the architecture of the IoT system with a drag-and-drop way. Based on the proposed IoTStakeholder meta-model, we present in Figure 8 the instantiated diagram which describes our system using our tool.

## 8. Conclusion

In this paper, we propose an approach that permits the modeling of the Internet of Things architecture. Our approach consists of offering a three Meta-models which represent a high level of abstraction. Then, we instantiate models that are conformed to the proposed meta-models to describe both structural and behavioral view of the Internet of Things architecture. Moreover, we implement the proposed use case. Also, we develop a tool that can help architects in modeling of the IoT system using Eclipse modeling frameworks.

Future work will include the Event-B method to specify both structural and behavioral features to have valid and correct models. Also, we will specify formally the quality attributes offered by Design Patterns.

## Acknowledgments

This work was partially supported by the LABEX-TA project MeFoGL: "Méthodes Formelles pour le Génie Logiciel".

## References

- [1] F. Fleurey, B. Morin, Thingml: A generative approach to engineer heterogeneous and distributed systems, in: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), 2017, pp. 185–188.
- [2] K. Thramboulidis, F. Christoulakis, Uml4iot-a uml-based approach to exploit iot in cyber-physical manufacturing systems, *Comput. Ind.* 82 (2016) 259–272. URL: <https://doi.org/10.1016/j.compind.2016.05.010>. doi:10.1016/j.compind.2016.05.010.
- [3] Costa, Bruno and Pires, Paulo F. and Delicato, Flávia C., Modeling SOA-Based IoT Applications with SoaML4IoT, in: 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), 2019, pp. 496–501. doi:10.1109/WF-IoT.2019.8767218.
- [4] D. A. Robles-Ramirez, P. J. Escamilla-Ambrosio, T. Tryfonas, Iotsec: Uml extension for internet of things systems security modelling, in: 2017 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE), 2017, pp. 151–156.
- [5] B. Costa, P. F. Pires, F. C. Delicato, Modeling IoT Applications with SysML4IoT, in: 2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2016, pp. 157–164. doi:10.1109/SEAA.2016.19.
- [6] S. Ogata, H. Nakagawa, Y. Aoki, K. Kobayashi, Y. Fukushima, A tool to edit and verify iot system architecture model, in: MODELS, 2017.
- [7] T. Imen, M. Hadj Kacem, A. Hadj Kacem, An approach for modeling and formalizing soa design patterns, in: 2013 Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2013, pp. 330–335. doi:10.1109/WETICE.2013.26.
- [8] I. Tounsi, M. Hadj Kacem, A. Hadj Kacem, K. Drira, A refinement-based approach for building valid SOA design patterns, *IJCC* 4 (2015) 78–104. URL: <https://doi.org/10.1504/IJCC.2015.067705>. doi:10.1504/IJCC.2015.067705.
- [9] S. Qanbari, S. Pezeshki, R. Raisi, S. Mahdizadeh, R. Rahimzadeh, N. Behinaein, F. Mahmoudi,

- S. Ayoubzadeh, P. Fazlali, K. Roshani, A. Yaghini, M. Amiri, A. Farivarmoheb, A. Zamani, S. Dustdar, Iot design patterns: Computational constructs to design, build and engineer edge applications, in: 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI), 2016, pp. 277–282.
- [10] L. Reinfurt, U. Breitenbücher, M. Falkenthal, F. Leymann, A. Riegg, Internet of things patterns, in: Proceedings of the 21st European Conference on Pattern Languages of Programs (EuroPLoP), ACM, 2016. doi:10.1145/3011784.3011789.
- [11] L. Reinfurt, U. Breitenbücher, M. Falkenthal, F. Leymann, A. Riegg, Internet of things patterns for device bootstrapping and registration, in: Proceedings of the 22nd European Conference on Pattern Languages of Programs, EuroPLoP '17, Association for Computing Machinery, New York, NY, USA, 2017. URL: <https://doi.org/10.1145/3147704.3147721>. doi:10.1145/3147704.3147721.
- [12] L. Reinfurt, U. Breitenbücher, M. Falkenthal, F. Leymann, A. Riegg, Internet of things patterns for devices: Powering, operating, and sensing, *International Journal on Advances in Internet Technology* (2017) 106–123.
- [13] L. Reinfurt, U. Breitenbücher, M. Falkenthal, P. Fremantle, F. Leymann, Internet of things security patterns, in: Proceedings of the 24th Conference on Pattern Languages of Programs, The Hillside Group, 2017. URL: <http://dl.acm.org/citation.cfm?id=3290281.3290305>.
- [14] L. Reinfurt, U. Breitenbücher, M. Falkenthal, F. Leymann, A. Riegg, Internet of things patterns for communication and management, *Transactions on Pattern Languages of Programming* 4 (2019) 139–182. URL: [https://doi.org/10.1007/978-3-030-14291-9\\_5](https://doi.org/10.1007/978-3-030-14291-9_5). doi:10.1007/978-3-030-14291-9\_5.
- [15] H. Washizaki, S. Ogata, A. Hazezama, T. Okubo, E. B. Fernandez, N. Yoshioka, Landscape of Architecture and Design Patterns for IoT Systems, *IEEE Internet of Things Journal* 7 (2020) 10091–10101. doi:10.1109/JIOT.2020.3003528.
- [16] M. Hachicha, R. Ben Halima, A. Hadj Kacem, Modelling, specifying and verifying self-adaptive systems instantiating mape patterns, *International Journal of Computer Applications in Technology* 57 (2018) 28–44.
- [17] M. Hachicha, R. Ben Halima, A. Hadj Kacem, Modeling and specifying formally compound MAPE pattern for self-adaptive IoT systems, *Innovations in Systems and Software Engineering* (2021). URL: <https://doi.org/10.1007/s11334-021-00409-3>. doi:10.1007/s11334-021-00409-3.