Original Research Paper

Modelling IoT Systems with UML: A Case Study for Monitoring and Predicting Power Consumption

[1,2]Marla Teresinha Barbosa Geller and [2,3]Anderson Alvarenga de Moura Meneses

[1]*CEULS/ULBRA-Lutheran University Center, Santarém, PA, Brazil*
[2]*Laboratory of Computational Intelligence, Federal University of Western Pará, Santarém, PA, Brazil*
[3]*Graduate Program in Amazon Natural Resources, Federal University of Western Pará, Brazil*

Article history
Received: 26-10-2020
Revised: 19-01-2021
Accepted: 22-01-2021

Corresponding Author:
Marla Teresinha Barbosa Geller
CEULS/ULBRA – Lutheran University Center, Santarém, PA, Brazil
Email: marla.geller@gmail.com

Abstract: Software Engineering has evolved to meet the growing complexity of current systems and the resources of the Unified Modeling Language (UML) enable the modeling of such systems in different views. The Internet of Things (IoT) paradigm appears with very peculiar characteristics such as the heterogeneity of its physical and virtual components that must be integrated. Designing systems of this nature is a challenge and modeling using UML is consolidating itself as a resource to overcome this challenge. The objective of this work is to present some proposals for UML extensions already available in the literature, to represent IoT systems. Then, we present a case study with those models for representing a small energy monitoring system with artificial intelligence for power consumption forecast.

Keywords: Unified Modelling Language, Internet of Things, Software Engineering, Power Consumption, Energy Efficiency

Introduction

The internet is the basis for the organization in networks in the information age, extending its ubiquitous characteristics to various technologies in order to form the Internet of Things (IoT; Serpanos and Wolf, 2017). The IoT interconnects systems and objects in different scales, consisting in complex systems called Cyber-Physical Systems (CPSs). The IoT technology enables the integration of several heterogeneous objects, for example a simple object with Radio Frequency Identification (RFID) or an autonomous vehicle; or yet, objects with great computational power or with limited computational resources.

The integration of such objects must be based on standardized communication protocols which use intelligent interfaces in a transparent way, for generating data in exponential quantity and variety (Bacquet *et al*., 2018), besides enabling prediction for decision making when implemented with Artificial Intelligence (AI) models. Such IoT characteristics represent a great opportunity for improving work and life conditions, with benefits in areas such as transportation, health care and electric power. However, new challenges also emerge such as heterogeneity, reusability, adaptability, security, analysis of massive amount of data, demanding the investigation of methods for the development of such systems (Ciccozzi and Spalazzese, 2016).

The usage of software engineering principles for the implementation of IoT systems is still being adapted. Although formal bases and a well structured project must be necessary, modeling IoT systems is still challenging. According to (Pressman, 2006), software engineering objectives include the best understanding of the problem, using resources such as Unified Modelling Language (UML) diagrams for modelling complex systems. According to (Zambonelli, 2016), software engineering, as a discipline, needs to identify resources and more general problems that characterizes IoT systems, for representing the integration of its elements in a model.

The UML uses diagrams that enable a graphical vision of the systems elements and their integration, although it must be extended and adapted for an IoT system representation. Some authors (Thramboulidis and Christoulakis, 2016; Robles-Ramirez *et al*., 2017; Reggio, 2018) propose the use of the UML for IoT systems, with the suggestion of extensions. One of the benefits of using UML for IoT is the great number of resources available. The challenge of modelling an IoT system is the sufficiency level of details for the engineers to implement it and at the same time abstracting complexities for a high-level understanding for system configuration.

In this sense, developing IoT systems for power monitoring is a challenging task, taking into account the diversity of internet-enabled devices, sensors, as well as systems requirements and the type of decision-making that is targeted. Thus, the main contributions of the present work are: (1) Demonstrating how researchers are developing UML for IoT systems; and (2) proposing an IoT model for a power consumption prediction and monitoring software, with Artificial Intelligence (AI).

IoT and UML

UML Background

The UML is a language of general usage for specifying, visualizing, constructing and documenting artifacts of the software system (Booch *et al*., 1999). One of its objectives is the standardization of different methods that already existed for representing object-oriented systems. Its current version (2.5.1) has 14 diagrams which represent static and behavior aspects of a system (OMG, 2017). With the increasing complexity of current systems, the UML has aggregating extension for specifying several systems characteristics, which includes pervasive computing, distributed computing, multiplatform systems, etc.

IoT Background

The IoT has emerged as a technology with application in several areas such as health care, business and transportation. The IoT is based on existing technologies such as sensor networks and internet protocols (Serpanos and Wolf, 2017).

The main elements that compose an IoT system are sensors, software, communication system and actuators, which generate a massive amount of data. The way such devices interact is given by the architecture of the system. Issues related to data collection, storage, representation, retrieval and usage are implicit to such systems.

The main input of IoT-based applications is data continuously generated in several physical or virtual devices in order to offer services for the users. Such big data generated and made available by an IoT system is a helpful input for decision-making, with the possibility of application of AI for tasks such as preprocessing and data analysis. Thus, AI is the technology that helps an IoT system to give sense to the overwhelming amount of data.

Modeling IoT Systems with UML

Representations of IoT systems using UML resources are discussed in the present section. Although there is not a standard and sufficiently representative language for IoT systems, the UML is one of the visual modelling resources that is making possible the usage of extensions for representing such systems.

According to (Eterovic *et al*., 2015), an IoT system may be represented by two types of languages: Textual and visual. The authors propose the usage of UML resources as visual language, for representing the various parts of an IoT system as:

- *Things* – Basic element of an IoT system represented by the UML diagram of components. In this sense, components communicate and build an IoT system
- *Annotation* – Resource for specifying the type of objetcs as <<*virtual*>>, or a collection of objects as <<*subsystem*>>, etc
- Encapsulation and subsystems – Collection of objects that are part of a same context
- Items – Components of an object, which are classified in three groups: Input (e.g., sensors), output (e.g., actuators, switches and SMS messages) and software components, represented by classes with their respective stereotype (<<input>>, <<*output*>>, or <<*component*>>) and are grouped inside the objects. The items communicate through interfaces represented by three forms: Circle, semicircle and the stereotype <<interface>>
- *Ports* – The internal structure of a system with its objects, items and relationships is represented by *ports*, which show how the subsystems interact with each other. A subsystem may be represented as a black box or a white box
- *Rules* – Rules are represented as methods inside a UML class and relate to items and ports

Figure 1 shows such elements of the model proposed by (Eterovic *et al*., 2015). Two subsystems, House and Work are interconnected by the port pTemp which connects the input device <<*input*>> Temperature through the interface iTemp. The subsystem work is represented as a black box whereas the subsystem House is represented as a white box.

Thramboulidis and Christoulakis (2016) state that the IoT brings along great opportunities for companies to reach better performance in global and distributed environments. However, IoT is at an initial phase and demands research for the development and standardization of safe and reliable technologies for efficient decision-making. Those authors investigate the development of UML4IoT, which integrate CPS and the IoT. It describes a framework for orienting the challenges introduced by the usage of the IoT in the process of products development.

UML4IoT presents two ways of modelling the interface of simple intelligent objects: (a) Using the UML class diagram and extensions for specifying a part of the system and (b) using source code in Java, if high level projects are not sufficiently represented by UML resources. The UML4IoT is Object-Oriented (OO) and use the class diagrams with extensions, forming a profile for a particular domain.

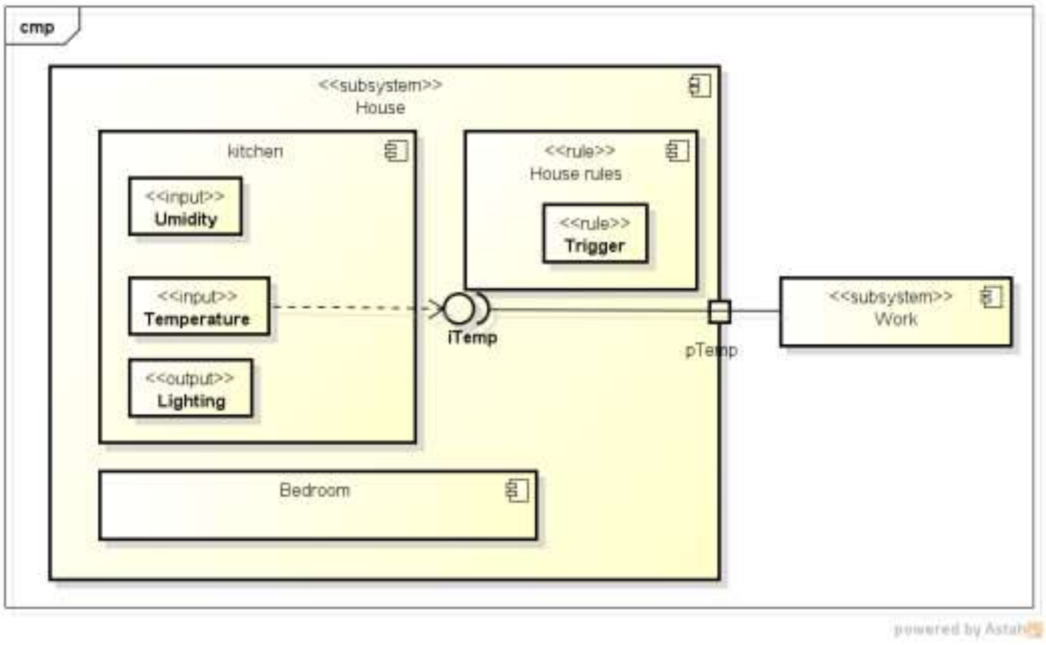

Fig. 1: UML elements used in an IoT model (Eterovic *et al*., 2015)

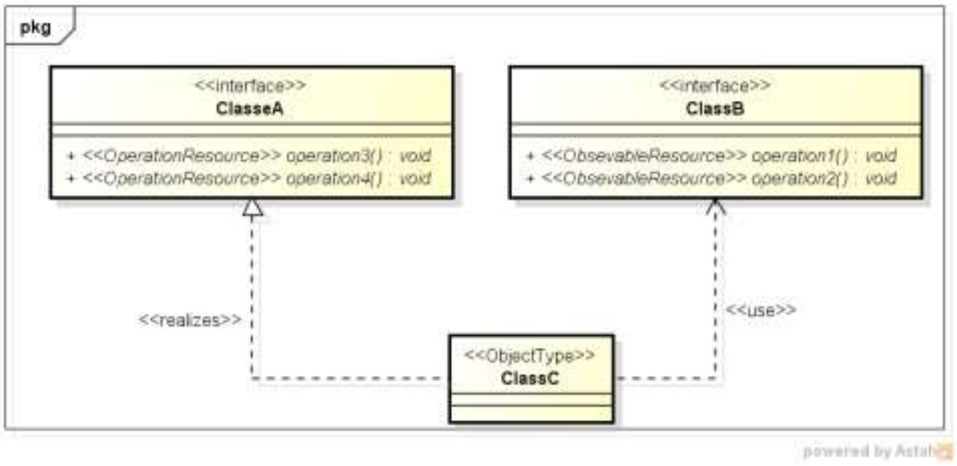
pkg
<<interface>>
ClasseA
+ <<OperationResource>> operation3() : void
+ <<OperationResource>> operation4() : void
<<interface>>
ClassB
+ <<ObsevableResource>> operation1() : void
+ <<ObsevableResource>> operation2() : void
<<realizes>>
<<use>>
<<ObjectType>>
ClassC
powered by Astah

Fig. 2: UML extensions used in the UML4IoT proposal. Adapted from (Thramboulidis and Christoulakis, 2016)

According to (Thramboulidis and Christoulakis, 2016), in an IoT system there are components which request services, represented in the model by the UML stereotype <<*realizes*>>. The stereotype <<*interface*>> represents the interfaces that enable the communications between the systems components.

In the Fig. 2, the ClassA which provides services for the ClassC has its methods stereotyped as <<*OperationResource*>>, whereas ClassB, which uses ClassC's services has its methods stereotyped as <<*ObservableResource*>>. Such extensions and other defined by the profile UML4IoT enable the transformation of a UML OO traditional approach to a Representional State Transfer (REST) architecture (Thramboulidis and Christoulakis, 2016).

Reggio (2018) uses the UML for specification of the functional and non-functional requirements of IoT systems, in the Service-Oriented Architecture (SOA) paradigm. The method proposed is the IoTReq. According to the author, an IoT system present peculiar characteristics which need specific methods for representing their requirements, implementing a hardware and software intersection. The IoTReq method proposes the domain modeling, passing to an extended domain modeling, definition of the strategic objectives, specification of operational objectives (functional requirements) and the definition of the technological objectives (non-functional objectives), as shown in Fig. 3.

According to the SOA paradigm followed by the IoTReq method, an IoT system has participants which use and provide services through an architecture and as such it is modelled. For static vision of the model the participant objects are stereotyped as <<*participant*>>. Such objects provide and use services through ports that are stereotyped as <<*service*>> for services provided and as a tilde ("~") before the service's name for services used, as shown in Fig. 4. In Fig. 4, ClassA provides the services serv1 and serv2, whereas ClassB uses such services. Both classes are stereotyped as <<*participant*>>, since both are parts of the IoT system.

Patnaik and Snigdh (2019) report the main concepts and abstractions related to the IoT paradigm which can be represented with UML resources. The use case diagram, for example, designates functional requirements of the system, actors as well as objectives of the application. The class diagram represents a static model with the systems objects and their relationships. Sequence, Collaboration, Activity and state diagrams model the interactions and the component and deployment diagrams are suggested when necessary. The authors refer to (Zambonelli, 2016) as a basis for their proposal.

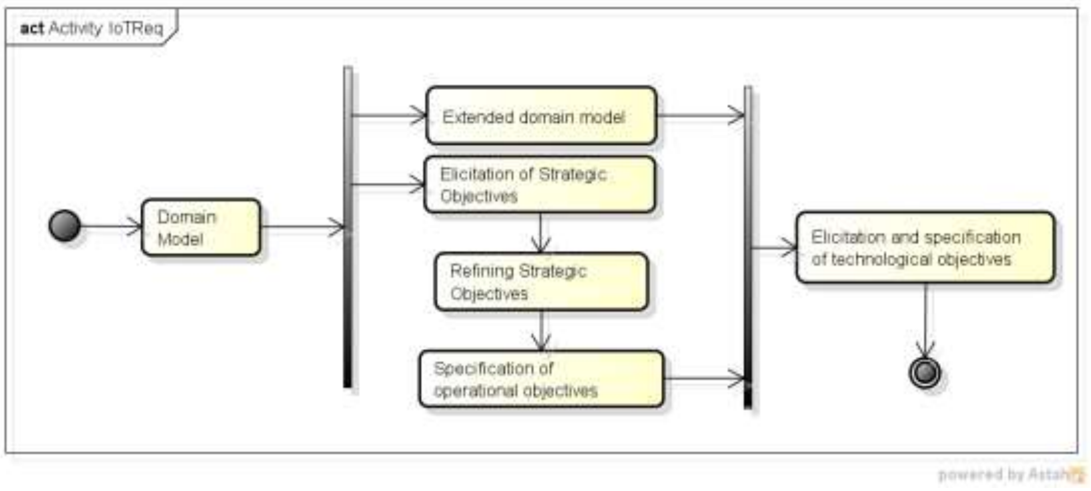
act Activity IoTReq
Domain Model
Extended domain model
Elicitation of Strategic Objectives
Refining Strategic Objectives
Specification of operational objectives
Elicitation and specification of technological objectives
powered by Astah

Fig. 3: Overview of the IoTReq method (Reggio, 2018)

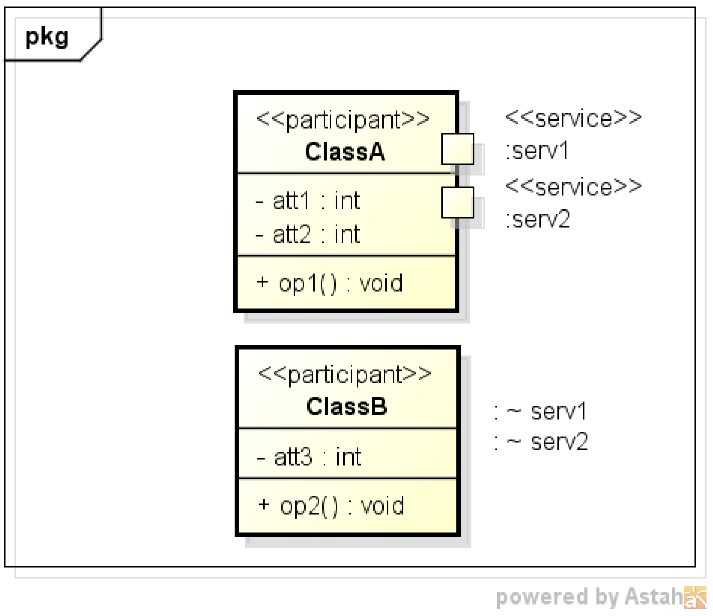

Fig. 4: UML stereotypes for the static view of an IoT system. Adapted from (Reggio, 2018)

Ciccozzi and Spalazzese (2016) propose the method MDE4IoT and use the resources of the base subset for executable UML models – Foundational UML (fUML) and its action language ALF described in (OMG, 2018; 2017) for describing the system's functionalities. Software allocations for the hardware and hardware components are represented through the component diagram. The consistency between the models is enabled by MDE4IoT from several standpoints since the UML is used for creating specific domain profiles having a unique metamodel as a basis (Ciccozzi and Spalazzese, 2016).

Prehofer and Chiarabini (2013) propose the combination of two approaches for modeling IoT systems: A model-based approach together with mashup tools. Such proposal is shown in Fig. 5 with the integration of the component diagram and the Paraimpu tool (Pintus *et al*., 2012).

For the model-based approach, (Prehofer and Chiarabini, 2013) use the class diagram as well as the component diagram and map the physical entities in a deployment diagram. The system behavior is modeled with the sequence diagram, state machine diagram and activity diagram as originally suggested by the UML.

Robles-Ramirez *et al*. (2017) present the IoTsec, which uses UML extensions for security encapsulated in UML nomenclature and stereotypes for modeling common actors. The objective is to facilitate the representation of security issues with a visual notation, even if the developers are not completely familiar to Internet security concepts.

The ThingML approach (Morin *et al*., 2017) includes a set of tools and a methodology directed to IoT applications, besides modeling with UML resources. Editors, exports to UML and multiplatform code generation are among those tools.

A Reference Architecture Model (RAM) of an IoT system is represented with UML resources by (Bauer *et al*., 2013), which discuss architecture issues in detail. An extension of the language SysML (OMG Friedenthal *et al*., 2006; Roudier and Apvrille, 2015) extended the notation originating the SysML-sec version for capturing security and protection issues. The authors define model-driven environments with the Model-Driven Engineering (MDE) for supporting the system development and use a tool for automate the verification and formal simulation of models, providing online feedback for UML diagrams.

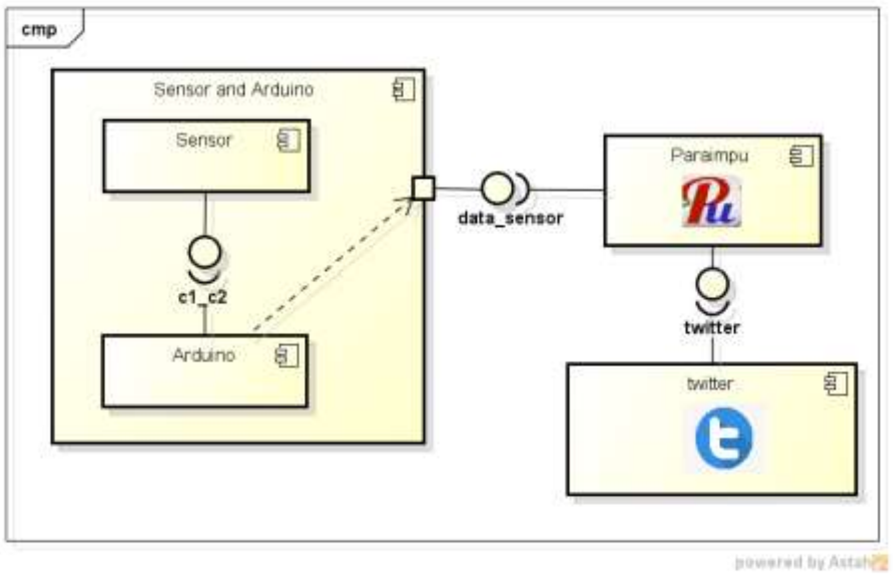

Fig. 5: Component diagram integrated with mashups tools (Prehofer and Chiarabini, 2013)

Table 1: Methods that use UML and its extensions (Robles-Ramirez *et al*., 2017)

UML for IoT	1	2	3	4
UMLsec (OMG Friedenthal *et al*., 2006)		▨	▨	▨
IoT-A (Bauer *et al*., 2013)	▨		▨	
SysML (OMG Friedenthal *et al*., 2006)	▨		▨	
SysMLsec (Roudier and Apvrille, 2015)		▨	▨	▨
UML4IoT (Thramboulidis and Christoulakis, 2016)	▨		▨	
ThingML (Morin *et al*., 2017)	▨			
IoTsec (Robles-Ramirez *et al*., 2017)	▨	▨	▨	▨
IoTReq (Reggio, 2018)			▨	

Thus, several methods for representing IoT systems use existing resources. Table 1 summarizes those methods, resources and approaches. The columns are the characteristic presented by each method: (1) Specific extensions for IoT; (2) reference and security models of the systems; (3) UML extensions or visual representation; and (4) security requirements model.

Case Study: Power Consumption Monitoring and Prediction System – EnergySaver

In this section, a case study with the EnergySaver system is presented. EnergySaver is a system for monitoring and predicting power consumption with AI, developed by the Laboratory of Computational Intelligence of the Federal University of Western Pará, in Brazil. The components of the system are represented in Fig. 6.

The EnergySaver system monitors electronic devices, for example with a current sensor connected to a water cooler at the laboratory, representing an edge device (leaf node) of the IoT system. The sensor collects data for the Arduino, which retransmits them to a Raspberry Pi. The Raspberry Pi is responsible for transmitting data to the server using the Message Queue Telemetry Transport (MQTT; http://www.mqtt.org) protocol. Data are stored in a MongoDB database (www.mongodb.com) and sent to a webpage in real-time.

A Long-Short Term Memory (LSTM; Hochreiter and Schmidhuber, 1997), a type of recurrent neural network, was implemented, tested and deployed in the prediction module for forecasting univariate time series (e.g., power consumption data of a university building, or power consumption data of an edge device).

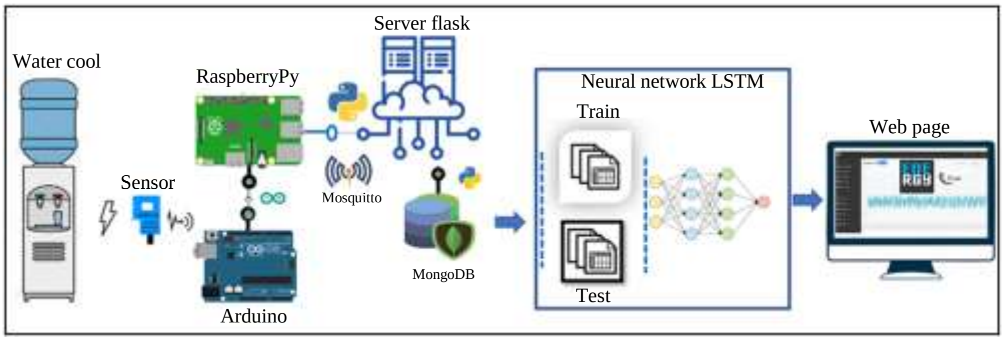

Fig. 6: Layout of the power consumption monitoring and predicting system – EnergySaver.

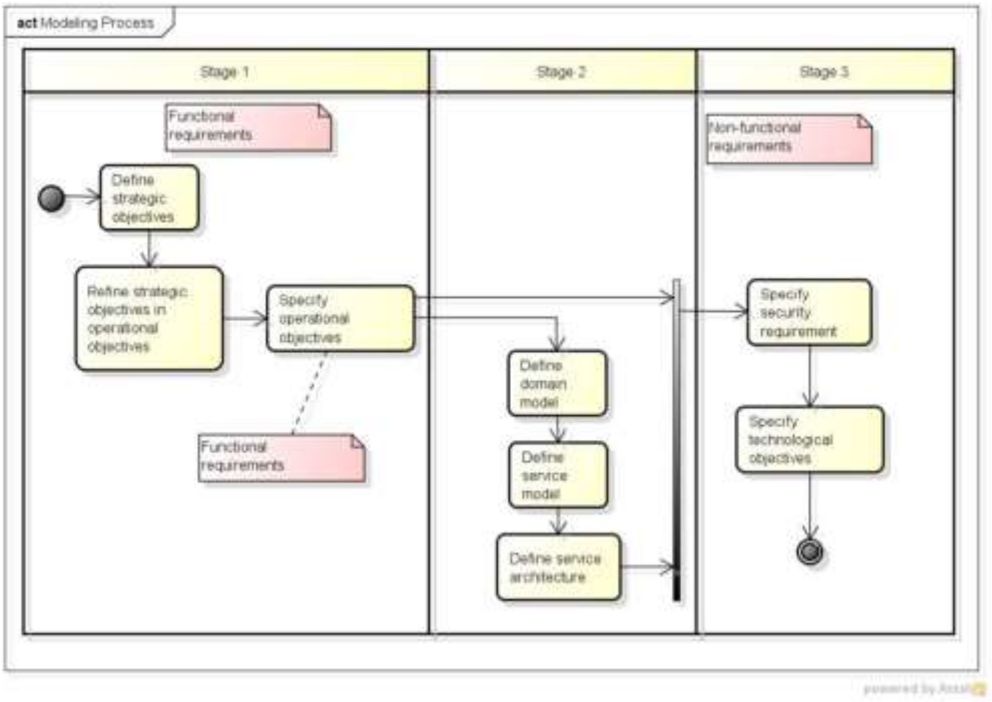
act Modeling Process
Stage 1
Stage 2
Stage 3
Functional requirements
Define strategic objectives
Refine strategic objectives in operational objectives
Specify operational objectives
Functional requirements
Define domain model
Define service model
Define service architecture
Non-functional requirements
Specify security requirement
Specify technological objectives
powered by Astah

Fig. 7: Activity diagram for modeling an IoT system

Methodology

Our system modeling is based on Service-Oriented Architecture (SOA), with the UML representation of the domain model, service model, service architecture and the security model. The modeling process was based on (Thramboulidis and Christoulakis, 2016; Reggio, 2018; Robles-Ramirez *et al*., 2017) and is shown in Fig. 7, detailing the activities developed in three stages. The first stage defines strategic objectives that are refined into operational objectives, giving rise to functional requirements. The second step models the system components with the domain model, the services offered and their architecture. The third stage of modeling specifies security requirements and defines the technology to be used through technological objectives.

Use case diagram without actors was used to define functional and non-functional requirements. Following Reggio's model (2018), thick lines were used to specify the strategic objective (Fig. 8). The strategic objective is to monitor and predict energy consumption and is subdivided into sensor monitoring, viewing a web page and making predictions. The dotted arrows show the connections between them. Actors are not represented in this model.

Figure 9 to 11 show the EnergySaver system's operational objectives (requirements). Thin lines are used for operational objectives that translate into system requirements.

The second stage of the modeling shows the static view of the system through the domain model with the class diagram (Fig. 12). The purpose of this model is to show the components of the system, which are represented by stereotyped classes. The main classes are identified with the <<*participant*>> stereotype, which in the EnergySaver model are: Water cooler, sensor, web page and LSTM neural network.

The service model for the use case "Update data set" is represented by the sequence diagram (Fig. 13), where the participating objects exchange messages by making available and using services.

Classes participating in the service are stereotyped with <<*service*>>. The services use two types of in/out interface to specify the use or offer of a service respectively.

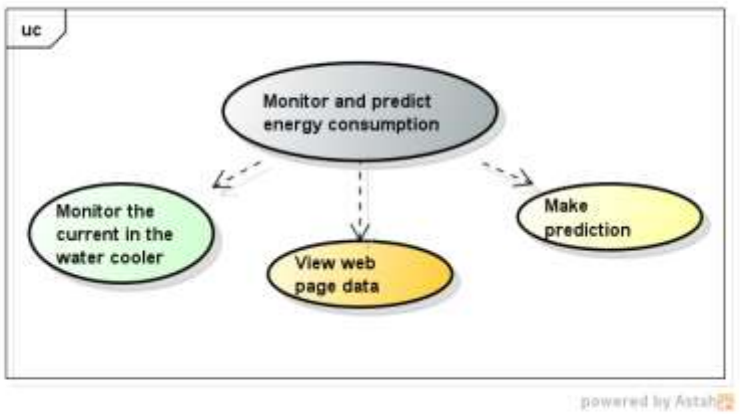
uc
Monitor and predict energy consumption
Monitor the current in the water cooler
View web page data
Make prediction
powered by Astah

Fig. 8: Strategic objectives of the IoT system

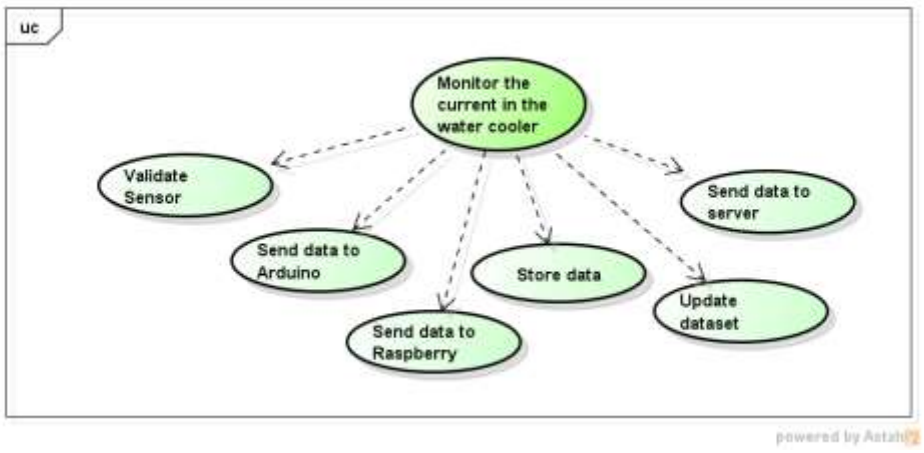

Fig. 9: IoT system operational objectives (requirements)

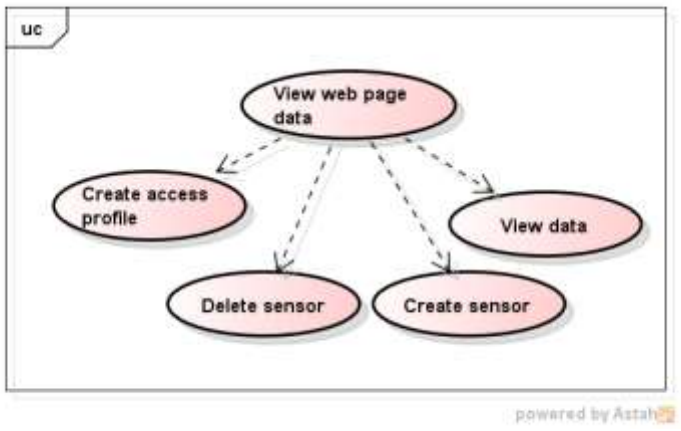

Fig. 10: IoT system operational objectives (requirements)

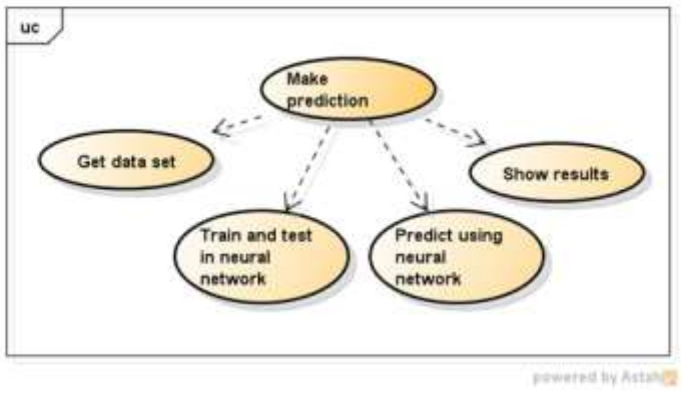
uc
Make prediction
Get data set
Show results
Train and test in neural network
Predict using neural network
powered by Astah

Fig. 11: IoT system operational objectives (requirements)

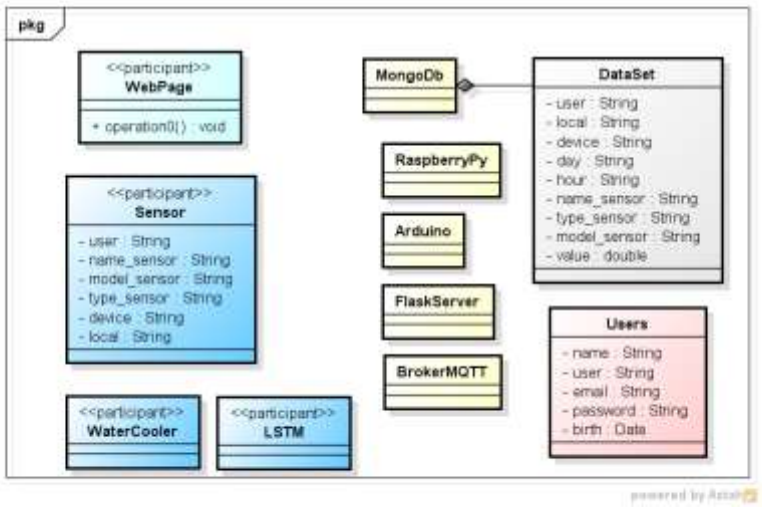

Fig. 12: Domain model (classes) of the IoT system

For the IoT system modeled, the update service is represented by the service class "Update" with its interfaces (UpdateIn and UpdateOut) in the sequence diagram. The LSTM network is responsible for predicting information based on data stored in the database (see the Service Model "make prediction" in

Fig. 14). The objects participating in this service are: The data set, the LSTM neural network and the web page. The sequence of exchanging messages between these objects includes details of the network's functions with their input and forget gates, as well as training and testing the network.

The services architecture was modeled in the third stage of the process, using the class diagram, with the representation of the publisher/subscriber communication interfaces. The system classes use ports, represented by small rectangles for the interfaces to exchange services, as in Fig. 15. As an example, the services exchanged between the Arduino and the Raspberry Pi. The Arduino device provides the reading of the data and calculates the average and makes it available to the Raspberry Pi.

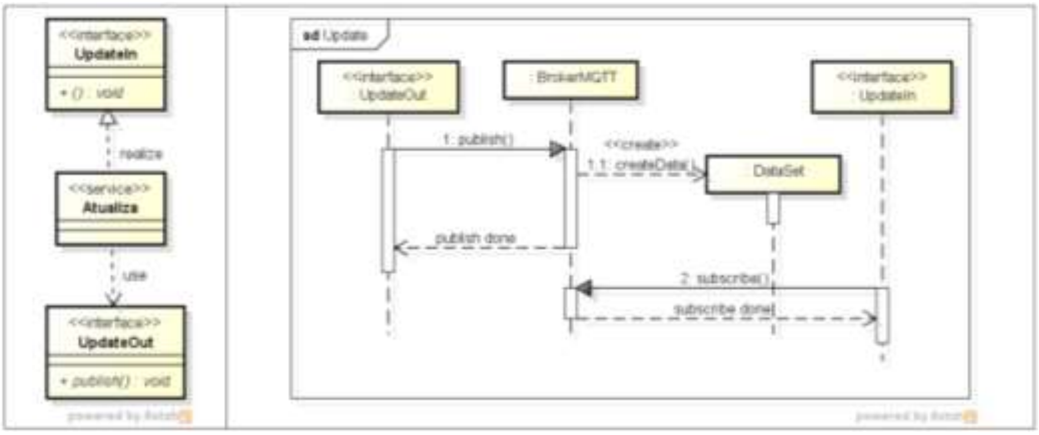

Fig. 13: Service model of the IoT System

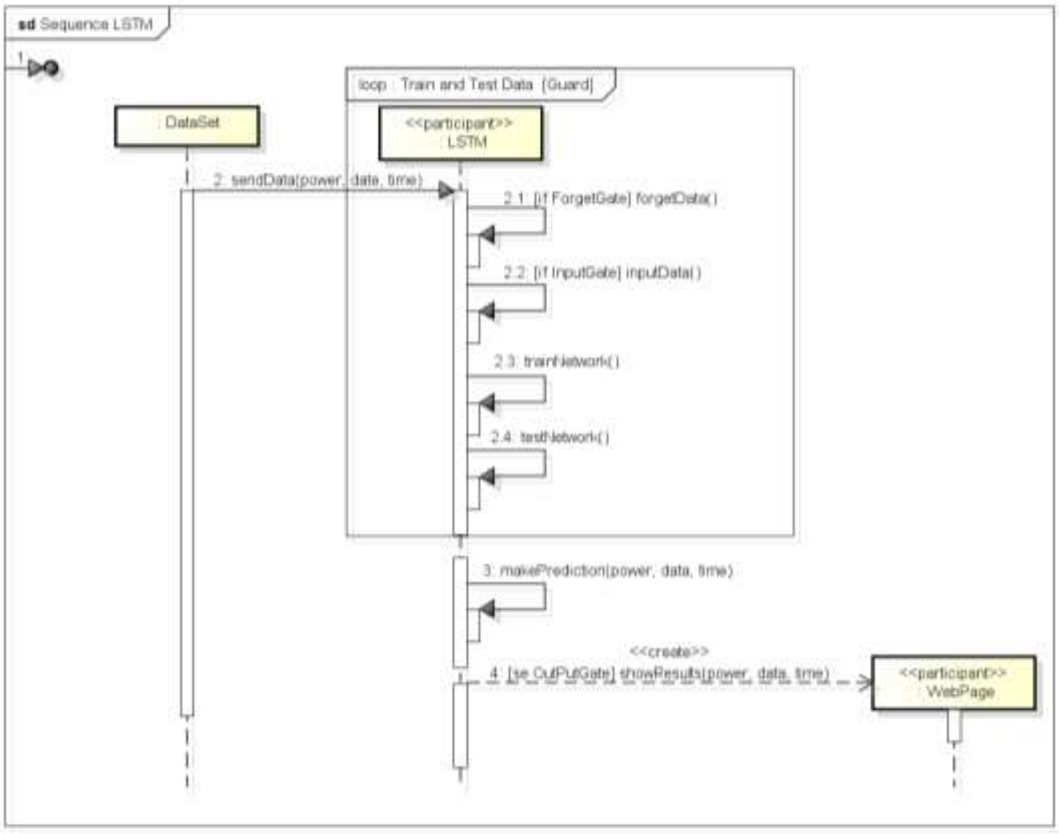
sd Sequence LSTM
loop : Train and Test Data [Guard]
: DataSet
<<participant>> : LSTM
2: sendData(power, date, time)
2.1: [if ForgetGate] forgetData()
2.2: [if InputGate] inputData()
2.3: trainNetwork()
2.4: testNetwork()
3: makePrediction(power, data, time)
<<create>>
4: [se OutPutGate] showResults(power, data, time)
<<participant>> : WebPage

Fig. 14: Service model of the use case “make prediction”

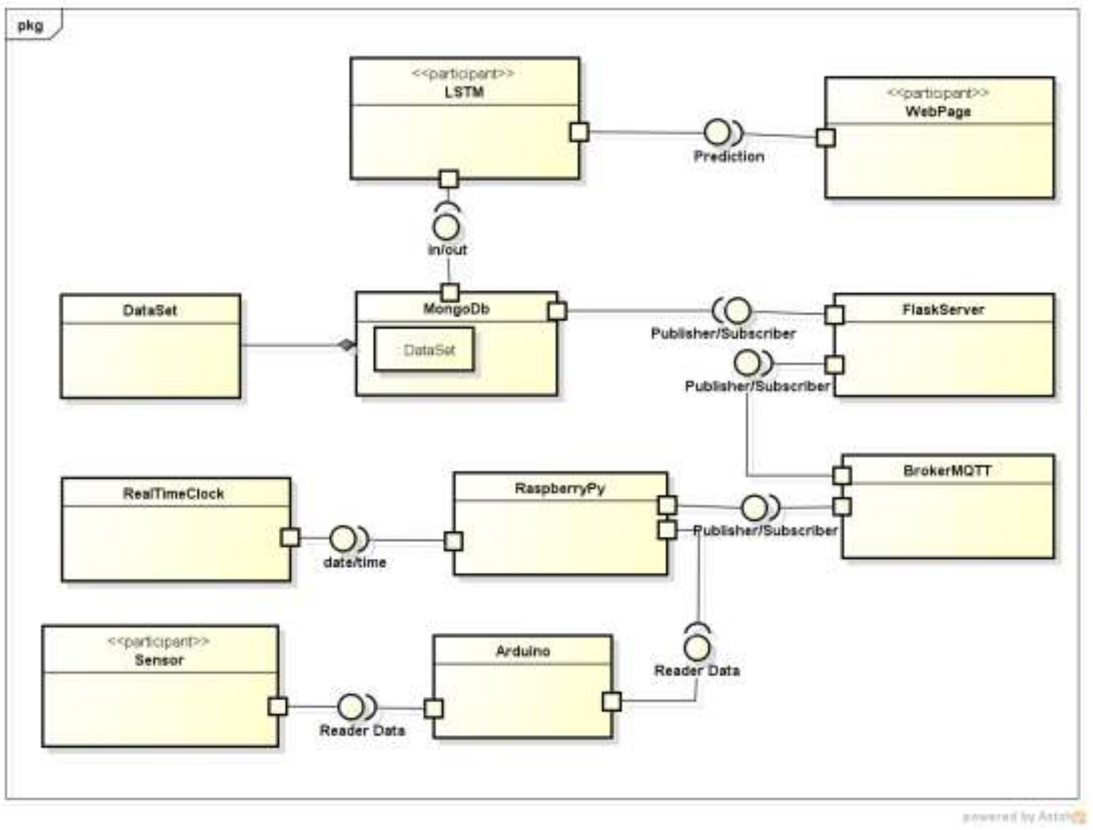

Fig. 15: Classes interfaces with the exchange of services

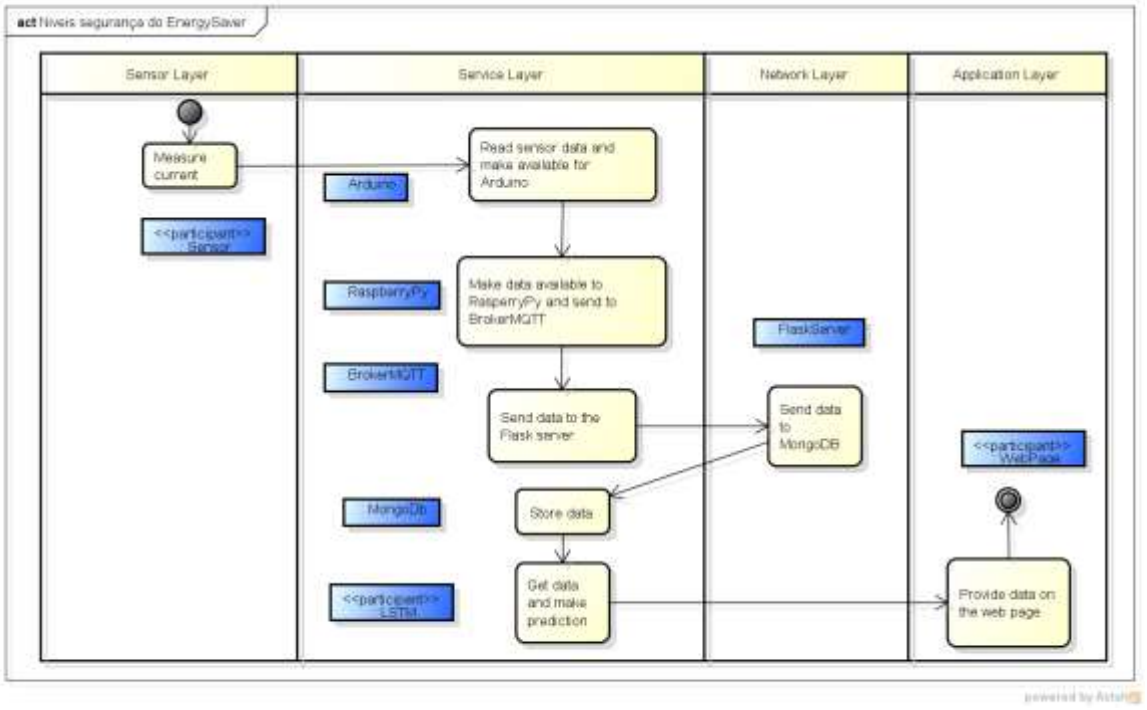
act Niveis segurança do EnergySaver
Sensor Layer
Service Layer
Network Layer
Application Layer
Measure current
<<participant>> Sensor
Arduino
Read sensor data and make available for Arduino
RaspberryPy
Make data available to RasperryPy and send to BrokerMQTT
BrokerMQTT
FlaskServer
Send data to the Flask server
Send data to MongoDB
MongoDb
Store data
<<participant>> LSTM
Get data and make prediction
<<participant>> WebPage
Provide data on the web page

Fig. 16: Lane model to specify the different levels of the system

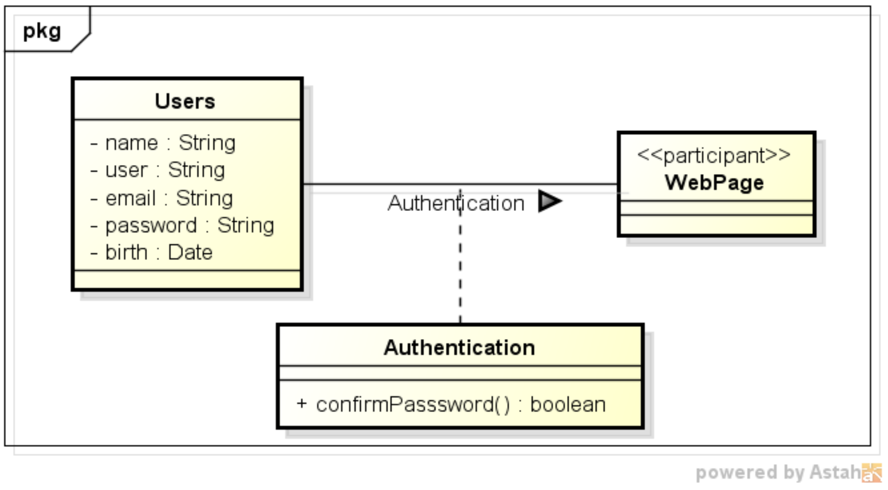

Fig. 17: Web page access authentication requirement

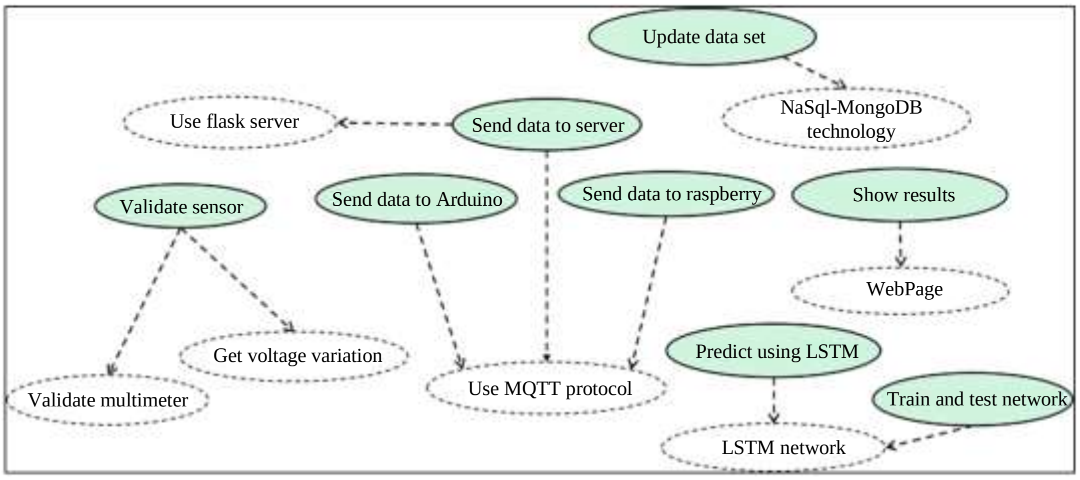

Fig. 18: EnergySaver technological goals

The security model is based on the work of (Robles-Ramirez *et al*., 2017). Activity diagram organizes in lanes the different levels of the application showing the activities and components that are part of each level. Thus, it is possible to see the need to implement a security protocol for each level, as shown in Fig. 16.

Figure 17 exemplifies the representation of a security requirement for accessing the web page. In this way, the requirements for all levels can be represented.

The technological objectives of the system will be the basis for non-functional requirements. They can be represented, according to (Reggio, 2018), with use cases with dotted lines, as shown in Fig. 18. Use case model specifies the technologies used for each operational objective. Thus, to validate the sensor (operational objective) it is necessary a Variac voltage regulator and and a multimeter. MongoDB and NoSql technologies are used to store data. The MQTT protocol is used to send data to Arduino and Raspberry Pi. In order to make the predictions, an LSTM neural network is deployed. Flask is the server used for the system and the results are displayed on a web page.

Conclusion

With UML resources, it is possible to represent a small IoT system, following the proposals of several authors (Thramboulidis and Christoulakis, 2016; Robles-Ramirez *et al*., 2017; Reggio, 2018). Different stages of the system implementation produced very consistent models, including functional and non-functional requirements that were represented by the UML use cases. The domain model was specified with the classes and their relationships. The class diagram and the sequence diagram with its communication interfaces modeled the services of the system, including AI tools, which enables pattern recognition and decision-making, making sense of the massive amount of data in IoT systems. Classes with ports and interfaces were used to design the services architecture. The activity diagram showed a security model for the different levels of the IoT system.

The challenge of modeling IoT systems lies in their heterogeneity, due to its physical and virtual components that are integrated, forming a complex system. UML resources are able to represent the different views of an IoT application (static, behavioral, security, etc.) using its diagrams and extensions.

The need to visually represent IoT systems was the motivation for researchers to use the UML resources, giving rise to several proposals. Just as UML emerged to standardize the representation of OO systems in 1999, the current effort is to standardize it to a consistent UML for IoT.

Acknowledgment

A.A.M.M. acknowledges Banco da Amazônia S.A. (BASA) for supporting this research (Project Energy Monitoring System using Internet of Things, Big Data and Machine Learning).

Author’s Contributions

Marla Teresinha Barbosa Geller: Research, methodology, visualization, writing-original draft.

Anderson Alvarenga de Moura Meneses: Conceptualization, supervision, writing-review and editing.

Ethics

This article is original and contains unpublished material. The authors have read and approved this manuscript and no ethical issues are involved.

References

Bacquet, J., Riemenschneider, R., & Jensen, P. W. (2018). Future Trends in IoT. In: Next Generation in Internet of Things, Vermesan, O., & Bacquet, J., (Eds.), River Publishers Series in Communications, pp: 9-17. ISBN: 9788770220071.

Bauer, M., Boussard, M., Bui, N., Carrez, F., & Francois, C. (2013). Internet of Things – Architecture IoT-A, Deliverable D1.5 – Final architectural reference model for the IoT v3.0. Internet of Things - Architecture (IoT-A).

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). The Unified Modeling Language User Guide Addison-Wesley. Reading.

Ciccozzi, F., & Spalazzese, R. (2016, October). Me4iot: supporting the internet of things with model-driven engineering. In International Symposium on Intelligent and Distributed Computing (pp. 67-76). Springer, Cham.

Eterovic, T., Kaljic, E., Donko, D., Salihbegovic, A., & Ribic, S. (2015, October). An Internet of Things visual domain specific modeling language based on UML. In 2015 XXV International Conference on Information, Communication and Automation Technologies (ICAT) (pp. 1-5). IEEE.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Morin, B., Harrand, N., & Fleurey, F. (2017). Model-based software engineering to tame the iot jungle. IEEE Software, 34(1), 30-36.

OMG Friedenthal, S., Moore, A., & Steiner, R. (2006, July). OMG systems modeling language (OMG SysML) tutorial. In INCOSE Intl. Symp (Vol. 9, pp. 65-67).

OMG. (2017). Action Language for Foundational UML (Alf); Concrete Syntax for a UML Action Language; Version 1.1. https://www.omg.org/spec/ALF/1.1/PDF

OMG. (2018). Semantics of a Foundational Subset for Executable UML Models (fUML). Version 1.4. OMG Document Number: formal/2018-12-01 https://www.omg.org/spec/FUML/1.4/PDF. (Retrieved Jan 10, 2020).

Patnaik, K. S., & Snigdh, I. (2019). Modelling and Designing of IoT Systems Using UML Diagrams: An Introduction. In Integrating the Internet of Things Into Software Engineering Practices (pp. 36-61). IGI Global.

Pintus, A., Carboni, D., & Piras, A. (2012, April). Paraimpu: a platform for a social web of things. In Proceedings of the 21st International Conference on World Wide Web (pp. 401-404).

Prehofer, C., & Chiarabini, L. (2013). From IoT mashups to model-based IoT. In W3C Workshop on the Web of Things.

Pressman, R. S. (2006). Engenharia de Software, McGrawHill, 6a.

Reggio, G. (2018, May). A UML-based proposal for IoT system requirements specification. In Proceedings of the 10th International Workshop on Modelling in Software Engineering (pp. 9-16).

Robles-Ramirez, D. A., Escamilla-Ambrosio, P. J., & Tryfonas, T. (2017, November). IoTsec: UML extension for Internet of things systems security modelling. In 2017 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE) (pp. 151-156). IEEE.

Roudier, Y., & Apvrille, L. (2015, February). SysML-Sec: A model driven approach for designing safe and secure systems. In 2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD) (pp. 655-664). IEEE.

Serpanos, D., & Wolf, M. (2017). Internet-of-things (IoT) systems: architectures, algorithms, methodologies. Springer.

Thramboulidis, K., & Christoulakis, F. (2016). UML4IoT—A UML-based approach to exploit IoT in cyber-physical manufacturing systems. Computers in Industry, 82, 259-272.

Zambonelli, F. (2016). Towards a general software engineering methodology for the Internet of Things. arXiv preprint arXiv:1601.05569.