Towards a UML profile for designing smart IoT data-centric applications

Houssam Bazza, Sandro Bimonte, Julian Eduardo Plazas, Laure Moiroux Arvis, Hassan Badir, Juan Carlos Corrales, Stefano Rizzi

Towards a UML Profile for Designing Smart IoT Data-Centric Applications

Houssam Bazza[1], Sandro Bimonte[2][0000−0003−1727−6954], Julian Eduardo Plazas[3][0000−0001−6621−0498], Laure Moiroux Arvis[2], Hassan Badir[1], Juan Carlos Corrales[3][0000−0002−5608−9097], and Stefano Rizzi[4][0000−0002−4617−217X]

[1] IDS, Abdelmalek Essaadi University, Tangier, Morocco
houssam.bazza@etu.uae.ac.ma, hassan.badir@uae.ac.ma
[2] TSCF, INRAE Clermont-Ferrand,9 avenue Blaise Pascal, Aubière, France
firstname.lastname@inrae.com
[3] GIT, Universidad del Cauca, Popayán, Cauca, Colombia
firstname.lastname@unicauca.ma
[4] University of Bologna, Bologna, Italy
stefano.rizzi@unibo.it

Abstract. The implementation of IoT (Internet of Things) systems is difficult since the data sent from the devices is complex, especially in agriculture and agroecology, where it is generated from heterogeneous hardware and software, and its applications involve different actors. In this scenario, conceptual design is mandatory to provide a formal and unambiguous representation allowing the different actors to set their requirements. The problem with the current representations is that they do not take into account neither the internal parameters nor the dynamic aspect of smart devices. To fill this gap we propose SmartSTS4IoT, an extension of the STS4IoT UML profile that models the different representations of internal/external data expressed from the same sensor and the logic used to adapt the sending/sensing policies to sudden environmental changes. The profile is illustrated with reference to a case study in the context of smart agriculture and validated theoretically.

Keywords: Internet of Things, UML profile, smart sensors

1 Introduction

Internet of Things (IoT) is a widely used technology. It is based on the convergence of wireless technologies, acquisition devices (sensors, smartphones, vehicles, etc.), and the internet to provide decision-makers with real-time data issued from different locations. Recent advances in the electronic and hardware components of IoT devices have led to a new, smarter kind of devices that goes beyond a basic data-sending functionality by incorporating some computation capabilities (ranging from simple rules to AI). So these smart devices can dynamically change their data sensing and sending behavior according to internal and/or external data. This leads to pushing intelligence down, in a real-time

and distributed way, at the devices level [17], thus opening perspectives for the development of many applications in several areas, such as health, traffic, smart building, and agriculture. Although IoT systems offer new important possibilities, their implementation is still difficult and time-consuming since (i) the data sent from smart devices are complex (real-time spatio-temporal stream data), and (ii) their implementation requires the interaction of various actors (decision-makers, sensors, data and network experts), who must be able to understand each other to clearly specify their functional and non-functional requirements, as well as the technological means [10].

Some recent works investigate the usage of software engineering methodologies for IoT [10]. In particular, conceptual design has been recognized as a mandatory activity for a successful implementation of complex systems. Several formalisms have been proposed to this end. Recently, the use of UML for the conceptual design and automatic implementation of IoT-based data-centric applications has been proposed [14]. The authors present a UML profile that allows to describe the IoT data used by any application by means of simple UML Class diagrams. The main idea is to provide IoT experts and decision-makers with an unambiguous formalism (UML) they can use to discuss and converge toward an implementation that satisfies the functional and non-functional requirements of the application. However, the profile proposed does not explicitly model the inner logic governing the behavior of smart devices.

To fill this gap, in this work we extend the profile proposed in [14] to allow different representations (including sensing and sending policies) of the data issued from the same IoT device. This allows defining explicitly the logic used to change the representation by means of OCL or other UML diagrams. Our proposal, named *SmartSTS4IoT*, represents a framework allowing the actors of IoT applications to discuss and converge towards shared and feasible requirements before the implementation step. Our claim is that representing smart devices at a conceptual level can guide and help IoT experts in the choice of the right technologies and their correct deployment.

The paper is structured as follows: Section 2 motivates our proposal by means of a case study in smart agriculture; Section 3 discusses the related work; Section 4 presents the SmartSTS4IoT profile; finally, Section 5 gives a theoretical evaluation of SmartSTS4IoT and draws the conclusions.

2 Requirements for designing smart IoT applications

In this section, we list the requirements that our conceptual model should support, using smart agriculture as a representative scenario [5].

With the advent of IoT, agriculture is moving towards a digital transformation and new tools are being developed to optimize the management of farms and harvests with a low level of human intervention. IoT is not limited to the deployment of simple measurement sensors for precision agriculture, but it comes with on-board intelligence that provides valuable assistance to decision-makers. For example, to optimize water consumption, sensors coupled with an intelligent

irrigation system are used to automatically plan the irrigation of a plot. Thus, the intelligence provided by IoT allows better knowledge and control of agricultural operations. In this paper we consider as running example the monitoring of fires in fields.

Internal parameters. The air temperature sensors are deployed in open fields, far from agricultural buildings, and they are powered by batteries. It is important to alert the farmers when the battery level is too low, in order to plan a human intervention in the field to promptly replace the batteries. Thus, first of all, a conceptual model for smart IoT should support the representation of *internal parameters* of the devices that must be communicated over the network (the battery level in this case).

State-dependent behavior. A sensor can operate in different states, and it will move from one state to another following the application of rules; depending on the state of the sensor, it can manage different data. As an example, consider the IoT application in charge of alerting farmers when a fire is burning. A temperature exceeding a given threshold (say 27 Celsius degrees) could mean that a fire is starting. However, before a fire prevention action is launched, a more accurate monitoring should be started to prevent false alarms. Thus, the temperature could be measured by end nodes every minute rather than every 30 minutes for some time and for all sensors in the field. Besides, the data sent from these end nodes should be compared inside a sink node to verify that they are coherent, i.e., that high temperature values are not erroneous. The logic used by nodes to change sensing and sending policies is crucial in these applications, so its formal representation is necessary to let the different actors involved agree on a solution. Sensor states can even be associated to different data acquisition policies. For example, in presence of high temperatures, air humidity should be monitored as well; thus, a device can manage different data at different times.

Data-centric representation. As stated in [14], the dynamic aspects of devices should be represented in the same diagram used for data, because they have a great impact on the semantics of the data collected and sent. This representation should also be well understandable, since the actors involved in the design and implementation of these systems have different skills (for example, farmers and experts in sensors, databases, and networks).

3 Related work

IoT technologies are mature enough to provide effective solutions to real life problems. However, their conceptual design has not been sufficiently studied by the academic and industrial communities. A conceptual representation for IoT has been investigated in several works, mostly focused on the design of physical component of the IoT system, i.e., on the *internal parameters* requirement. From the point of view of *state-dependent behavior*, few works have been proposed. Some works focus on the conceptual design of intelligence rules inside devices. For instance, in [9] a Model-Driven Architecture (MDA) approach is proposed to improve the interoperability among IoT devices so as to ensure

better data compatibility. In [15] the authors propose to combine mashup and model-based development, and represent the behavior of IoT components using UML Activity and State diagrams. A design architecture for representing the dynamic components for cyber-physical systems is introduced in [13], by distinguishing different types of smart devices. The possible components of a smart device are represented using classes and generalization within a UML Class diagram. A visual domain modeling for IoT based on UML is presented in [7]; it is understandable by users, and it represents logic rules as methods of UML classes. Noticeably, all the works described above do not explicitly associate intelligent rules neither to specific data collected, nor to sensing and sending operations; thus, the *state-dependent behavior* requirements is not met.

As to the *data-centric representation* requirement, the problem of having alternative data representations coexist has been framed as *multi-representation data*. In the context of spatial databases, multi-representation has been proposed [16, 4] to assign multiple attributes with different types to the same geographic element, according to the spatial scale used by the system. In the context of database design, multi-representation has been used to take into account temporal and thematic features. For instance, in [16] the MADS model is proposed, where each element of the database schema is annotated with a *stamp* defining a particular representation of the data by the end-user that is associated to some specific rules. This proposal comes with some limitations if used for smart IoT, mainly, methods describing sensing, sending, and transformations are not supported. The usage of UML for multiscale database schemata is proposed in [4], where different types are associated to one geometrical attribute of a class and a class operation is used to change the attribute type according to the scale in input. All these works provide good frameworks to design data with multiple representations but they are not suited for IoT data and their dynamic aspects.

4 A UML profile for smart IoT

UML provides a formal extension mechanism through *profiles*, i.e., meta-models that define specialized semantics and rules while respecting the standard UML structure and constraints [11]. The new semantics introduced by a profile allows UML to be customized for specific domains or platforms by extending its meta-classes (e.g., Class, Property, and Package) through Stereotypes, Tagged Values, and OCL constraints.

The STS4IoT UML profile [14] represents IoT applications involving multiple devices which execute different transformations. It can model raw or transformed data according to the application requirements, providing three kinds of operations (aggregation, conversion, and filter) which can be executed at different levels of the IoT network depending on the requirements expressed on data and device capacity. STS4IoT follows the MDA principles by providing an abstract data design (Platform Independent Model - PIM), an implementation view of the data for a specific IoT device (Platform Specific Model - PSM), and an automatic implementation for the selected device (Model-to-Code); thus, it supports

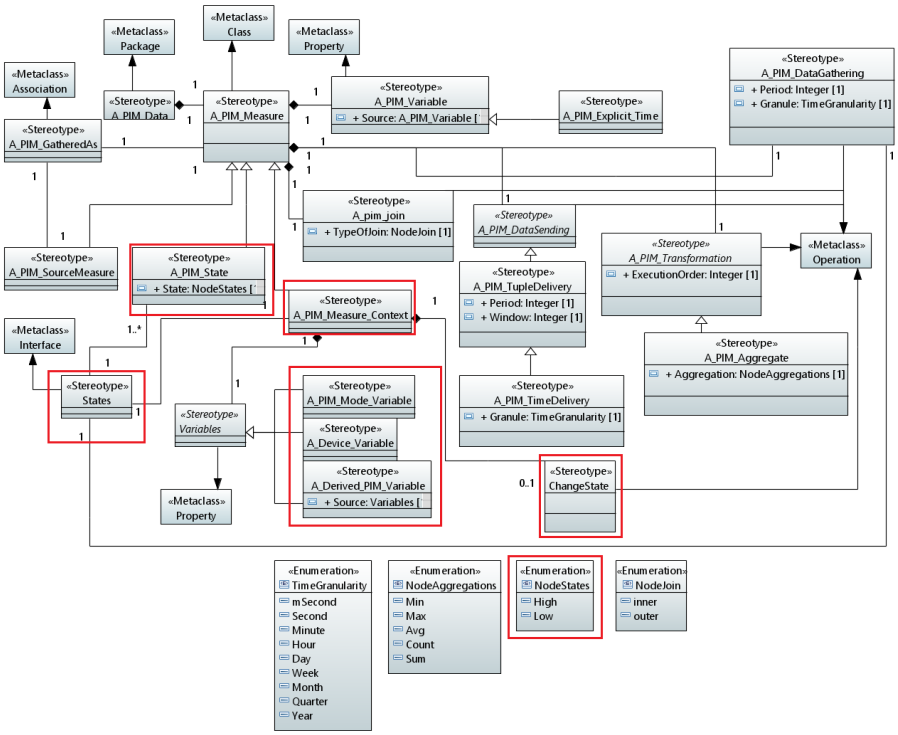

Fig. 1. The SmartSTS4IoT PIM Profile

fast prototyping. The STS4IoT PIM is centered around the A_PIM_Measure class, which represents the data collected by the IoT device. Its attributes (A_PIM_Variable) are the sensed and sent values. A_PIM_Measure comes with two methods representing the sensing and the sending policies: A_PIM_SensorGathering and A_PIM_DataSending, respectively. The original meta-model is shown in Figure 1; despite its advantages, the STS4IoT approach only supports static data, so it does not take the capabilities of smart IoT into account.

The new elements introduced in our extension of the STS4IoT PIM meta-model, called SmartSTS4IoT, are presented inside red squares in Figure 1 (the PSM and the device model are not discussed for space reasons). To cope with the requirements related to state-dependent behavior, we adopt the well-known *state pattern*, a behavioral software design pattern that allows an object to change its behavior when its internal state changes [8]. Its UML implementation is based on the Class element; its goal is to make the implementation of the dynamical behavior of class objects transparent and improve maintainability, so that new states can be easily added. It is structured as follows: (i) a Context class is introduced to represent all state-independent data; (ii) Context is associated to a State interface that provides the operations for handling dynamics; (iii)

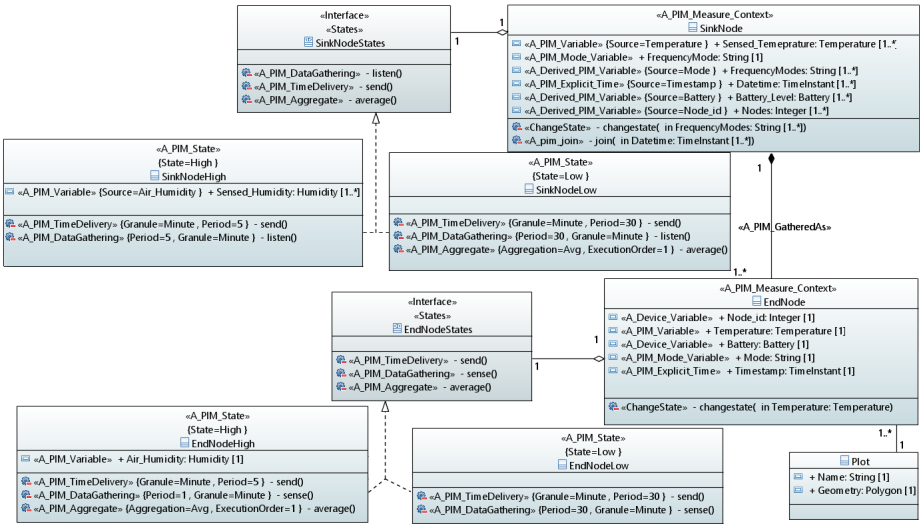

Fig. 2. PIM for our running example

multiple ConcreteState classes, corresponding to the different states of Context, implement this interface.

According to the state pattern, we have extended STS4IoT by introducing class A_PIM_Context, which extends A_PIM_Measure and represents the IoT device data independently from their states and behaviors. A_PIM_Context includes a new operation, ChangeState, in charge of changing states. The device can also send other types of information beside the gathered data, such as internal parameters (represented with the A_Device_Variable stereotype) or data calculated by the device (with the A_Derived_PIM_Variable stereotype).

An example is shown in Figure 2 to describe the case study of Section 2. The EndNode class represents temperature sensors (A_PIM_Variable), which also send a timestamp and the battery level (A_Device_Variable). This class includes the ChangeState operation, which takes in input the temperature value and changes the acquisition and sending policies accordingly. The interface of the state pattern, represented with the States stereotype, is associated to one A_PIM_DataGathering operation and one A_PIM_DataSending operation. Each state is represented with the A_PIM_State class stereotype, which provides a State tagged value to specify the representation currently adopted by the device. Its values (e.g., High and Low) are taken from the NodeStates enumeration.

In our example, the EndNode context class is associated via the EndNodeStates interface to two classes stereotyped A_PIM_State: EndNodeLow and EndNodeHigh. The former represents the behavior of the sensor in the context of a low temperature; in this situation, the sensor senses and sends data every 30 minutes. Class EndNodeHigh represents the sensor operation when the temperature is high. In this case the sensor is required to sense data every minute, and to

send their average every 5 minutes. Moreover, when in this mode, the device also gathers air humidity data. All EndNodes send their data to SinkNode, which receives the gathered temperature from all the sensors, their battery levels, and their current States. SinkNode changes its state, represented with the stereotype A_Derived_PIM_Variable, according to those of the Endnodes, yielding two possible behaviors: SinkNodeHigh (when all Endnodes are in High mode) and SinkNodeLow (when at least one EndNode is in Low mode). State switching is ruled by ChangeState(), which takes in input the states sent from the EndNodes.

The example described above shows how our UML profile supports the requirements listed in Section 2, namely, *internal parameters* and *state-dependent behavior*, by means of a single Class diagram, as requested by the *data-centric representation* requirement.

5 Evaluation and conclusion

IoT technologies are mature enough to be applied in real-life applications; however, the design and implementation of smart IoT data-centric applications is still time-consuming. In this paper, we have focused on the conceptual design of smart IoT data. We have extended the STS4IoT UML profile for IoT by adapting the state pattern design pattern to allow representing internal parameters as well as dynamic and multi-representation data, which characterize smart IoT. Our SmartSTS4IoT profile comes with all the advantages offered by the state pattern for embedded applications [6], since it separates the different data sensing/sending policies in different classes, thus enabling end-users to focus on one system state at a time, eventually making the UML diagram better readable.

To give a theoretical validation of SmartSTS4IoT we adopted the framework proposed in [3, 12], which evaluates a given meta-model against 2500 other meta-models in the literature using five quality metrics, namely, *reusability*, *understandability*, *functionality*, *extendibility*, and *well-structuredness*. These metrics are computed by counting some types of elements (e.g., abstract classes, associations, etc.) in the meta-model. It turned out that SmartSTS4IoT has the same quality than STS4IoT in terms of reusability, understandability, functionality, and well-structuredness, while leading to a 163% increase in terms of extendibility (0.29 against 0.11), meaning that new elements can easily be added to our meta-model. We also made a proof-of-concept implementation of our proposal, not described here for lack of space, through a laboratory implementation using the RIOT sensors available in the FIT IoT-LAB testbed [2, 1].

We are currently working on the automatic implementation of code for the RIOT platform, aimed at providing a set of large scale experiments to evaluate the conceptual and implementation gains ensured by the SmartSTS4IoT profile.

6 Acknowledgement

This work has been supported by the French National Research Agency under the IDEX-ISITE Project, initiative 16-IDEX-0001 (CAP 20-25) and by the Eu-

ropean Union Next-GenerationEU (PNRR – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.4 – D.D. 1032 17/06/2022, CN00000022). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

References

1. Adjih, C., et al.: FIT IoT-LAB: A large scale open experimental IoT testbed. In: Proceedings WF-IoT. pp. 459–464. Milan, Italy (2015)
2. Baccelli, E., et al.: RIOT: an open source operating system for low-end embedded devices in the IoT. IEEE Internet Things J. **5**(6), 4428–4440 (2018)
3. Basciani, F., Di Rocco, J., Di Ruscio, D., Iovino, L., Pierantonio, A.: A tool-supported approach for assessing the quality of modeling artifacts. Journal of Computer Languages **51**, 173–192 (2019)
4. Bédard, Y., Larrivée, S.: Spatial database modeling with pictrogrammic languages. In: Encyclopedia of GIS, pp. 716–725. Springer-Verlag (2008)
5. Bhatnagar, V., Singh, G., Kumar, G., Gupta, R.: Internet of things in smart agriculture: Applications and open challenges. International Journal of Students' Research in Technology & Management **8**(1), 11–17 (2020)
6. Douglass, B.P.: Design patterns for embedded systems in C: an embedded software engineering toolkit. Elsevier (2010)
7. Eterovic, T., Kaljic, E., Donko, D., Salihbegovic, A., Ribic, S.: An internet of things visual domain specific modeling language based on UML. In: Proceedings ICAT. pp. 1–5 (2015)
8. Gamma, E., Helm, R., Johnson, R.E., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series, Addison-Wesley, Reading, MA (1995)
9. Kaur, K., Sharma, A.: Interoperability among internet of things (IoT) components using model-driven architecture approach. In: Proceedings ICTCS. pp. 519–534 (2019)
10. Larrucea, X., Combelles, A., Favaro, J., Taneja, K.: Software engineering for the internet of things. IEEE Software **34**(1), 24–28 (2017)
11. Luján-Mora, S., Trujillo, J., Song, I.: A UML profile for multidimensional modeling in data warehouses. Data Knowl. Eng. **59**(3), 725–769 (2006)
12. Ma, Z., He, X., Liu, C.: Assessing the quality of metamodels. Frontiers of Computer Science **7**(4), 558–570 (2013)
13. Patnaik, K.S., Snigdh, I.: Architectural modelling of cyber physical systems using UML. International Journal of Cyber-Physical Systems **1**(2), 1–19 (2019)
14. Plazas, J.E., Bimonte, S., Schneider, M., de Vaulx, C., Battistoni, P., Sebillo, M., Corrales, J.C.: Sense, transform & send for the internet of things (STS4IoT): UML profile for data-centric IoT applications. Data Knowl. Eng. **139**, 1–29 (2022)
15. Prehofer, C., Chiarabini, L.: From internet of things mashups to model-based development. In: Proceedings COMPSAC. pp. 499–504. Taichung, Taiwan (2015)
16. Spaccapietra, S., Parent, C., Zimányi, E.: Spatio-temporal and multi-representation modeling: A contribution to active conceptual modeling. In: Proceedings ACM-L Workshop. pp. 194–205. Tucson, Arizona (2006)
17. Sriram, R.D., Sheth, A.: Internet of things perspectives. IT Professional **17**(3), 60–63 (2015)