A Toolbox for the Internet of Things – Easing the Setup of IoT Applications

Matheus Frigo[1,2], Pascal Hirmer[2],
Ana Cristina Franco da Silva[2], and Lucinéia Heloisa Thom[1]

[1] Institute of Informatics, Federal University of Rio Grande do Sul,
Bento Gonçalves Avenue, 9500, Porto Alegre, Brazil
lastname@inf.ufrgs.br
https://www.inf.ufrgs.br
[2] Institute for Parallel and Distributed Systems, University of Stuttgart,
Universitätsstraße 38, 70569 Stuttgart, Germany
lastname@informatik.uni-stuttgart.de
https://www.ipvs.uni-stuttgart.de/

Abstract. The Internet of Things (IoT) is a paradigm that aims at making people's life easier, e.g., through highly automated production processes in Smart Factories. In the IoT, heterogeneous interconnected devices communicate through standardized internet protocols to reach common goals. Well-known applications of the IoT are Smart Homes, Smart Factories, or Smart Cities. However, setting up new IoT environments can be cumbersome, since this oftentimes requires many complex manual steps, such as setting up devices, configuring sensors, or writing scripts. The heterogeneity and high distribution in the IoT makes this task even more complex and time-consuming. To ease the setup of IoT environments, we propose a new approach composed of (i) a toolbox with common building blocks of the IoT, and (ii) a business process based approach to orchestrate the setup of these building blocks. Through our approach, domain experts can select the building blocks they require for their IoT application and generate a step-by-step manual for their setup.

Keywords: Internet of Things · Toolbox · BPM · Domain Experts

1 Introduction

In the Internet of Things (IoT), interconnected devices communicate through standardized internet protocols to achieve common goals [17]. Usually, these devices are connected to sensors, to monitor the environment, and actuators to control it. The IoT enables creating new kinds of applications, such as Smart Homes [8], Smart Cities [5], or Smart Factories [15].

However, since devices in the IoT are very heterogeneous, building IoT applications can be very cumbersome [11]. For example, there are powerful devices, such as Raspberry Pis, able to provide an operating system and built-in communication technologies, such as WiFi or Bluetooth. In contrast, there are very

restricted devices, such as micro-controller boards, that do not provide any sophisticated operating system but rather a limited runtime environment to run small scripts, usually implemented in C programming language. Some of these restricted devices provide a WiFi connection, others require Bluetooth or need to be connected via cable.

Furthermore, in the IoT, not only the devices are very heterogeneous but also the communication protocols, e.g., CoAP [3], MQTT [9], or HTTP. This makes selection of the right hardware components and protocols for building IoT applications even more complex. Consequently, there is a demand for coping with this large heterogeneity and to support IoT application developers in making the right choices when it comes to hardware, technology, and protocol selection. This wide variety also introduces a challenge to create and use a uniform way of describing things in smart spaces in terms of what a particular thing is, what is does and how it communicates [10].

To cope with this issue, in this paper, we introduce a new approach for easing the setup of IoT environments and their applications. First, we introduce a toolbox, containing common building blocks that are oftentimes used in the creation of IoT environments. These building blocks can represent hardware components, network protocols, message brokers, gateways, IoT platforms, or any other software component. A building block consists of a high-level description to be understandable by domain experts as well as concrete implementations. These building blocks offer a uniform way of describing the software and hardware components of an IoT application. The building blocks are provided by experts in building IoT applications and are provided by a toolbox, which can be accessed by domain experts.

Second, we introduce a business process based approach to ease the set up of an IoT environment based on the suggested building blocks. Usually, setting up IoT environments requires many different steps that can be conducted either in parallel or sequentially in case of any dependencies. For example, before connecting a device to an IoT platform, a network connection needs to be set up. Hence, it makes sense to use business process management for orchestration of these steps. In this paper, we use the Business Process Model and Notation 2.0 (BPMN 2.0) [4] for modeling of the processes.

The remainder of this paper is structured as follows: in Section 2, we introduce our main contribution: a toolbox for the Internet of Things. Section 3 then describes a case study, which applies our approach to the Smart Factory domain. Section 5 discusses related work and, finally, Section 6 summarizes our paper and gives an outlook on future work.

2 A Toolbox for the Internet of Things

In this section, we introduce a holistic method that describes the necessary steps domain experts need to undertake to set up IoT applications from scratch. This method is depicted in Figure 1. The foundation for this method is our toolbox, containing a predefined set of building blocks that can be combined to set up

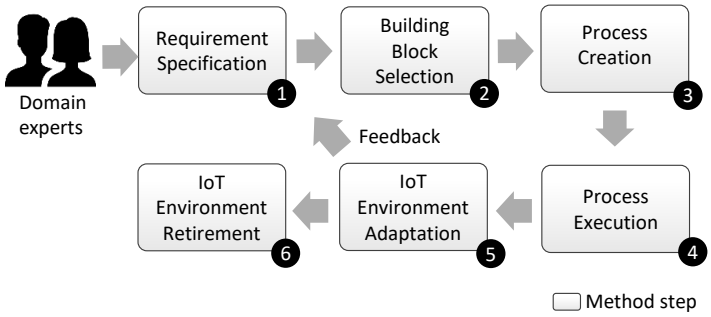

Fig. 1. Method for setting up IoT environments based on our toolbox

the desired IoT application. Even though this method is generic in terms of the kinds of applications that should be set up, applying it specifically to the IoT domain emphasizes its strengths, since in the IoT, usually many heterogeneous software and hardware components need to be set up, which can overwhelm application providers. Our method can help in setting up IoT environments with low effort and decreased necessary technical knowledge since technical details are abstracted through the building blocks.

In the first step of this method, domain experts and involved stakeholders discuss the characteristics of their IoT application and define a set of requirements for the application. The set of requirements is defined as $R \neq \emptyset$, whereas each $r \in R$ describes a specific application requirement. After that, in the second step, the toolbox recommends building blocks to be selected by domain experts. In step 3, a business process is created by experts based on the selected building blocks, defining the necessary steps that need to be undertaken to set them up. In step 4, this process is executed, guiding the domain experts in the process of creating the IoT environment for their application. In step 5, the IoT application is tested and, if there are no changes necessary, it goes into production. A feedback loop in this method ensures that new requirements can be considered also after the setup. In the final step 6, the IoT environment is retired, once an IoT application reaches its lifetime. In the following, the toolbox and the steps of this method are described in more detail.

2.1 Toolbox and Building Blocks

Our toolbox offers a set of building blocks (BB) that are divided into different categories. These categories include, for example, physical devices, network protocols, or software applications. In order to ensure understanding and applicability of our approach, we introduce a formalization of the toolbox. A BB is formalized as follows:

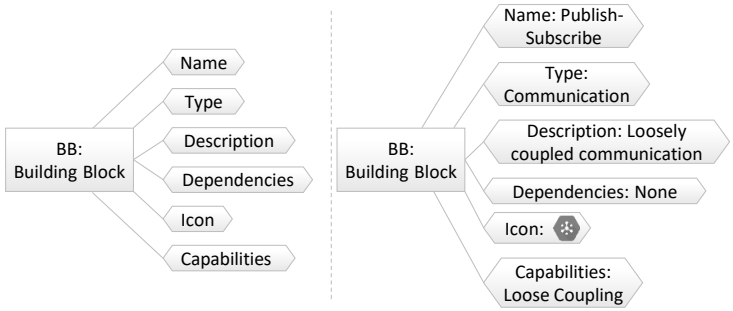

Fig. 2. Left: Remainder of a Building Block, right: Building Block example for publish-subscribe communication

Definition 1 (Building Block). Let $bb \in BB$ be a tuple $bb := (Name, Type, Description, Dependencies, Icon, Capabilities)$, whereas BB is the set of all available building blocks in the toolbox TB. A bb has the following properties:

- $Name \neq \emptyset$: unique name of the building block.
- $Type \neq \emptyset$: the type $t \in T$, where T is the set of available bb types.
- $Description$: optional description.
- $Dependencies$: set of $bb \in BB$ related to this building block.
- $Icon \neq \emptyset$: icon of the building block.
- $Capabilities$: set of capabilities of the building block, used to map user requirements to the BBs of the toolbox.

Figure 2 depicts on the left the structure of a building block. Each block contains a name, a type, a description (suitable for domain experts), dependencies to other building blocks, an icon to ensure recognition, and a set of capabilities that are able to fulfill the users' requirements. On the right of Figure 2, an example building block is given for the communication paradigm publish-subscribe, enabling loosely coupled communication. In an initial phase, experts in the IoT domain need to agree upon a common list of building blocks, which has to be extensible, since new technologies appear frequently in the IoT. We are aware that this is an ambitious goal, which would require some sort of standardization IoT experts agree upon. In our vision, IoT technology providers create building blocks themselves in order to promote their new solutions and add them to the toolbox. With a strong community, the toolbox could grow to a comprehensive collection of all kinds of IoT components.

Each BB has one or more implementations, which are referred to as building block implementation (BBI) in the following. A BBI is a concrete implementation of a BB, which is formalized as follows:

Definition 2 (Building Block Implementation). Let $bbi \in BBI$ be a tuple $(Name, Artifact, Description, Icon, ImplementedBB, Dependencies)$, whereas BBI is the set of all available building block implementations in the toolbox TB. A bbi contains:

- $Name \neq \emptyset$: unique name of the building block implementation.
- $Artifact$: software artifacts.
- $Description$: optional description.
- $Icon \neq \emptyset$: icon of the building block implementation.
- $ImplementedBB \neq \emptyset$: a list of BB implemented by the building block implementation.
- $Dependencies$: list of $bbi \in BBI$ related to this building block implementation.

Furthermore, $bbiMapping : BB \rightarrow BBi$ corresponds to the mapping, which assigns a BB to one or more BBIs.

A BBI contains a software artifact, which can be of different types, e.g., a Docker container, a cloud service, or a binary application. Furthermore, BBIs can have a set of dependencies that could require installation of different BBIs. For example, some BBIs might require the installation of certain programming languages (Java, Python) or a specific operating system. In addition, each BBI requires a definition of its interfaces to enable an easy orchestration via business processes. In case of software artifacts, these interface descriptions should be defined based on standards, such as WSDL [18].

For our BB example in Figure 2, implementations include MQTT brokers, such as Mosquitto or RabbitMQ. Once domain experts decide to use the publish-subscribe paradigm, they will find the according building block in the toolbox and will be able to select one of the corresponding implementations.

The BBs and BBIs can be defined in a hierarchical structure, meaning that building blocks can inherit characteristics of other building blocks, enabling their specialization. The depth of the hierarchy is arbitrary and is decided by the designer of the toolbox. Figure 3 depicts an example of hierarchical building blocks for a connection between devices. In this example, as root, the generic building block "Connection" is defined, representing all possible kinds of connections. This *Connection* BB is then derived to the "Wireless Connection" building block, which represents all connection BBs that are wireless. The *Wireless Connection* BB has three different implementations, which can again be derived by different BBIs. Each BBI could have BB and/or BBI dependencies, and one BBI can implement one or more BBs, therefore the relationship between BBIs could be AND, OR or both. After the toolbox is filled with BBs and BBIs, it can serve as a foundation for our method to set up IoT environments, which is described in the following.

Step 1: Requirement Specification

In the first step of our method, IoT application developers, i.e., the domain experts and their stakeholders, need to define a set of requirements for their application.

A requirement is a condition or capability needed by the user to solve a problem or to achieve an objective [14]. These requirements in an IoT system's context can be functional (e.g., sensed variable, sending rate) or non-functional, also known as quality-of-service requirements (e.g., lifetime, reliability) [6].

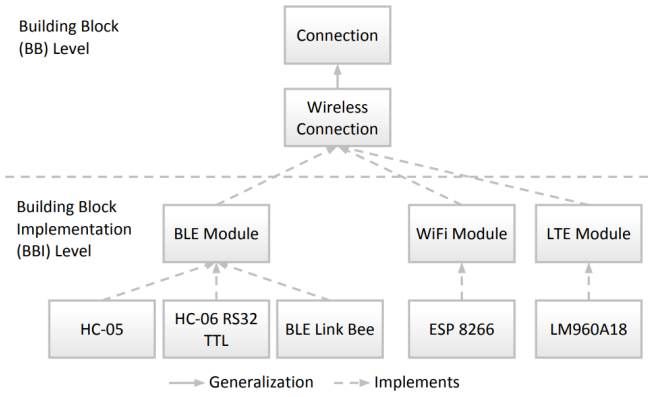

Fig. 3. Hierarchy of building blocks, example for a Wireless Connection building block

The toolbox provides a list of predefined requirements for the most common IoT application scenarios where the stakeholders can find the requirements that fit their needs. The list should be kept up-to-date to the current state-of-the-art of the IoT field. These requirements need to consider different aspects, such as network capabilities, used communication paradigms, costs, efficiency, security, privacy, available computing resources, and so on. Defining such requirements might involve many different stakeholders and even a requirements engineer, i.e., a person who understands the domain of the desirable application and also possesses enough IoT knowledge to be aware of the suitable features and resources for the development. E

To enhance the reusability of the toolbox in multiple projects, a systematic collection of requirements and the respective chosen building blocks are documented to provide an overview of possible relations, assisting in requirements engineering for future applications.

Step 2: Building Block Selection

In the second step of our method, involved stakeholders select the building blocks they need based on the set of requirements created in step 1. As mentioned before, the toolbox contains a collection of best practices in IoT application development and, hence, gives an overview regarding which technologies and approaches are available without the need for domain experts to acquire this knowledge themselves.

The selection of BBs is made based on recommendations by the toolbox, which are generated by matching the requirements of the IoT application to the capabilities of the building blocks. The final selection is conducted by the involved stakeholders. We define the mapping that assigns a requirement to one or more building blocks as $reqMapping : R \rightarrow BB$.

To make this selection process more clear, let us assume that a company aims at developing a smart factory application in which self-driving vehicles transport goods on the shop floor. The goal of this application is that these vehicles dynamically determine the best routes and also consider other vehicles as well as people and utilities of the shop floor. To achieve this, the vehicles need to be equipped with sensors, monitoring the environment, and need to communicate with other vehicles and participants on the shop floor through a wireless network. Setting up such a complex scenario, requires the expertise of many different stakeholders that need to agree upon functional and non-functional requirements. Regarding functionality, for example, an indoor localization system is essential to monitor the position of vehicles, utilities and people on the shop floor. In addition, a broad-band wireless network (e.g., 5G) needs to be established to cope with the high data load. Furthermore, non-functional requirements are of high importance in such safety-critical environments. Security, safety, robustness, accuracy and real-time capabilities are only some of the essential requirements of this scenario. In workshops, the stakeholders need to assess and discuss the requirements.

Based on the functional requirements "indoor localization" and "wireless communication", we show in Table 1, which BBs and BBIs could be recommended by the toolbox:

Table 1. Example of BB and BBI selection.

Building Block	**Matching Requirement**	**Implemented by BBI**
UWB Localization	indoor localization	RTLS localization modules
BLE Localization	indoor localization	BLE modules
WiFi	wireless communication	WiFi network
LTE	wireless communication	4G network
5G	wireless communication	5G network

If we now consider the non-functional requirements as well, for example high-efficiency, robustness, and accuracy, the toolbox would filter the resulting BB list accordingly. For example, localization using BLE and triangulation has issues regarding accuracy and also efficiency. Hence, the BB "BLE localization" would not be recommended by the toolbox, but rather a solution based on UWB (ultra-wideband). Similiar issues occur for WiFi connections, since they tend to be unstable. The toolbox could suggest using LTE or 5G networks. The company can then choose depending on costs or already available infrastructure on the shop floor.

Once the BBs are selected, a list of available BBIs is generated by the toolbox. The selection of the BBIs is then conducted by the stakeholders based on available expertise or costs.

Step 3: Process Creation

The third step deals with the business process modeling [7] to guide domain ex-

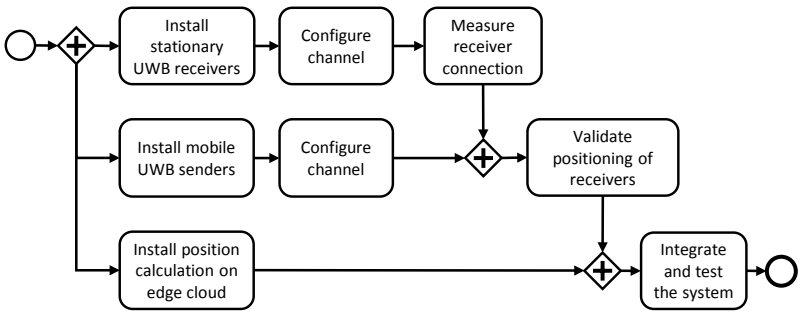

Fig. 4. Exemplary simplified BPMN 2.0 process to set up the indoor localization system

perts and involved stakeholders through the process of setting up their IoT environment. The tasks of the process set up the previously selected BBIs, whereas usually several process tasks are required for each BBI. The business process creation needs to be conducted manually by experts, which can become a cumbersome and time-consuming task.

Hence, setting up a given BBI in most of the cases will need common steps to be taken even in different scenarios and the toolbox categorizes these typical steps as *best practices* that can be used for the process creation. Assume, for example, that a BBI sets up an indoor localization system on the shop floor, the corresponding steps that need to be undertaken are mostly the same for different companies with only minor differences based on the layout of the shop floor.

An example of a process is provided in Figure 4 in BPMN 2.0 syntax. This process shows a simplified version of the different steps to set up an indoor localization system using UWB technology, as discussed above. In parallel, the receiver and sender hardware needs to be installed on the shop floor. Furthermore, software for calculation of the position needs to be installed in an edge cloud environment, which will later communicate with the receivers. After configuration and measurements of the hardware, the system can be integrated, tested, and can then go into production.

Note that the configuration of different BBIs can have a different number of tasks in the process due its complexity, as shown in our example process. For manual creation, the above mentioned best practices can help in selecting the tasks for the creator.

Step 4: Process Execution

In step 4, the process is then executed in order to realize the setup of the IoT environment. Depending on the chosen modeling language, a suitable BPMN 2.0 Business Process Management System (BPMS) needs to be provided. For BPMN 2.0, for example, the established BPMS Camunda could be used. In each step of the process, domain experts get a notification about the tasks they need to

conduct, for example, plugging in sensors, configuring a WiFi connection, or installing software on IoT devices. Documentation of the different technologies that are set up, can be found in the BBIs. In simple scenarios, setting up the IoT environments, should also be possible for non-expert users. However, usually this requires domain-specific expertise, as shown in our smart factory example. After task completion, the process moves to the next task. Parallelism is usually supported by such BPMS as well.

Once the process has successfully reached its end, the IoT environment is set up. This process can then be reused in similar scenarios, sometimes only with minor necessary adaptations. In case of issues in the processes creation or execution, e.g., due to unforeseen errors, experts need to be available to cope with occurring difficulties and to fix the problems. All occurred problems should then be documented inside the BBI's description so that this knowledge is preserved.

Step 5: IoT Environment Adaptation

It is not unusual that adaptations are necessary over time after an IoT environment was set up. For instance, due to changes in the application and, hence, changes in the requirements, or due to failing devices that need to be replaced.

If adaptations are necessary, the feedback loop in our method allows to return to the first step, redefining the requirements of the application, adding or removing requirements. After that, in the second step, the already chosen and running BBIs need to be considered, i.e., only the new BBs and BBIs should be chosen from the toolbox.

The process creation step then creates a so-called *adaptation plan*, which includes removing unnecessary BBIs and only setting up and integrating the newly added ones. After execution of the adaptation plan, the IoT environment is set up again, considering the new requirements as well.

Step 6: IoT Environment Retirement

The final step is the retirement of the IoT environment, which is executed once an IoT application reaches its lifetime. In this case, the creation of a *termination process* is required, reversing the steps of the original process setting up the environment. Using the process for creation as a foundation eases the creation of the termination process. After this process was executed, the IoT environment is retired. Note that the creation and termination processes should be stored since they can be useful to re-setup IoT applications after some time.

3 Case Study

This section presents a case study that applies our approach to a smart factory scenario. We worked on this scenario in cooperation with a large German producer of transport vehicles on shop floors. Our experiences with this scenario helped us with the conception and improvement of our toolbox.

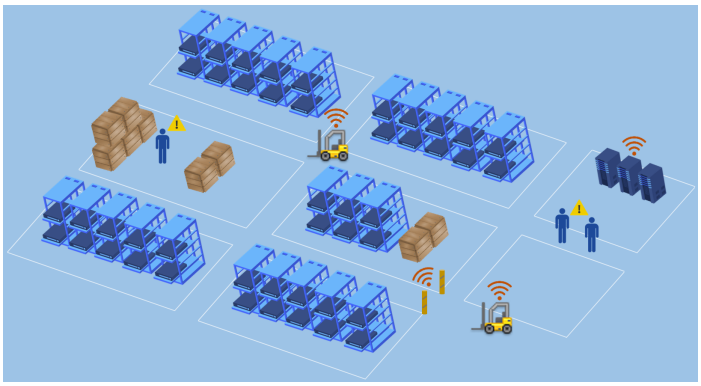

Fig. 5. Case study in the smart factory domain

In this scenario, depicted in Fig. 5, indoor localization of self-driving vehicles should be realized. These vehicles are able to pick up goods in a warehouse and transport them to a different location. In order to realize this scenario, so it is robust enough for real-world use, it must be known where the vehicles are at all times. For this, each vehicle needs a localization tag, which communicates with multiple stationary tags throughout the factory to determine the current location. Hence, many assets, sensors, and actuators need to be set up. In the following, the steps of our method are described specifically for this scenario.

Step 1: Requirement Specification. The requirement specification for this use case requires many different stakeholders: Shop floor workers that have experience with the environment the scenario is implemented in, network experts, setting up wireless connectivity and the edge cloud environment, indoor localization experts, setting up the localization system, electricians making sure that the power supply for the localization is guaranteed, business experts, bringing in knowledge about the processes, and so on.

Step 2: Building Block Selection. Once the domain experts have created the list of requirements, the toolbox makes suggestions regarding the building blocks that can be applied to this scenario. For example, one requirement is the need for an indoor localization system. The toolbox can propose using Bluetooth Low Energy localization through triangulation or more exact systems, using a real-time localization system, e.g., based on ultra-wideband (UWB). Since the requirement "high accuracy" is also given, the recommendation would suggest the more accurate UWB based system. Furthermore, wireless communication might be required. Due to the high efficiency requirements, a 5G network might be suggested instead of WiFi or LTE. The suggestions of the toolbox are an important part to decrease the time necessary to analyze and select the right systems for the scenario, which usually can take months.

Step 3: Process Creation. The process creation is conducted by an IoT expert and is, for now, designed manually. In this case, the set of BBs and BBIs above was given to a specialist and the process was created as shown in Figure 4 only for the localization system. For such a complex scenario, multiple processes are required to set up the specific parts (e.g., localization, door control, networks, etc.). Best practices based on previous scenarios can help in process creation.

Step 4: Process Execution. In order to set up the desired scenario, the processes in Figure 4 are executed. By doing so, the corresponding experts install the hardware, e.g., the localization system, by using the steps given by the process. After all processes have been executed, the scenario is set up. In general, during process execution, there might be dependencies between the processes. For example, it is necessary to set up the wireless communication first before the localization system can be set up. This needs to be modeled in the processes.

Step 5: IoT Environment Adaptation. In step 5, the IoT environment could be adapted by the stakeholders, for example, to extend the area the self-driving vehicles can move into or adding more vehicles. The overall setup should stay the same. Only necessary adaptations should be made in the IoT environment. In this case scenario, an adaptation process includes, e.g., setting up more UWB senders and receivers and an additional configuration and integration step. The rest of the setup should stay the same.

Step 6: IoT Environment Retirement. Possible cases when such a scenario is retired are moving the factory to another location or using other kinds of devices that fundamentally change the setup. In these cases, a termination process reverses all steps of the setup, for example by removing the UWB senders, powering down the edge cloud, etc.

Prototype. The toolbox is under development as a web-based prototype that will serve as a proof-of-concept for our approach. This prototype is available on Github (separated in backend: https://github.com/mtfrigo/IoT-Toolbox-Backend and frontend: https://github.com/mtfrigo/IoT-Toolbox-Frontend). A screenshot of the frontend is depicted in Fig. 6.

4 Discussion

The recommendation of Building Blocks is highly dependent on the applications' requirements. Their discovery is a complex task for the involved stakeholders. Hence, for the method to work well, it is essential that the stakeholders are supported in the requirement selection process by experts of different domains. Missing requirements may lead to a non-satisfactory result.

Our method treats the IoT application as a combination of software and hardware. Thus, it proposes specific software and hardware components, which

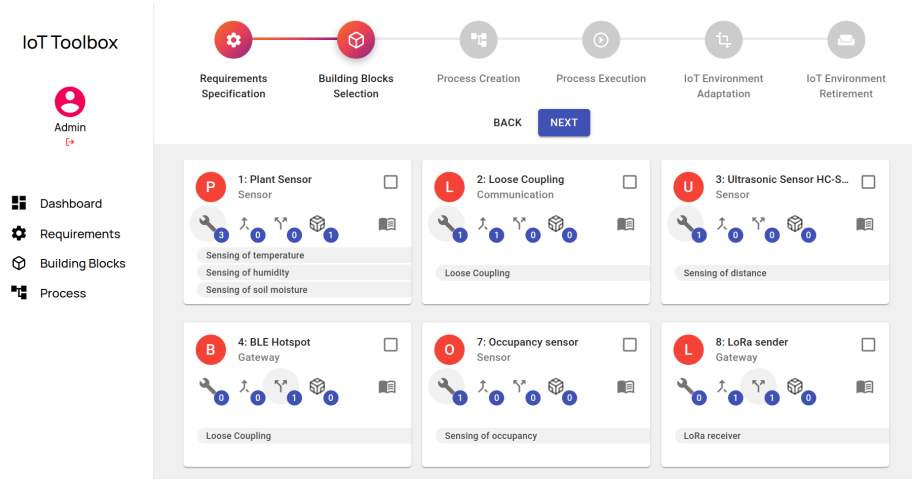

Fig. 6. Toolbox

should be used to implement the application and also provide artifacts for the implementation. However, there are limitations regarding the artifacts, as they are particular to the application and not to the domain, causing that the developer might have to add additional code manually to make the artifacts work with the respective environment. Hence, programmers are still necessary, even though their effort is decreased by our recommendations.

Furthermore, there are some limitations, especially regarding process creation and adaptation. Currently, the processes need to be created by experts in the corresponding modeling language, e.g., BPMN 2.0. Automated techniques, such as process mining, for the process creating and adaptation should be added.

As result of the provided case study, we have also identified that given a complex scenario, the task to gather suitable devices and technologies can take several years and creating an optimal solution is even harder. This time can be significantly reduced using the toolbox and choosing the recommended building blocks. Secondly, such a complex scenario, as the case study, can be hard and time-consuming to model even for an expert, and the toolbox doesn't provide any support for this yet. We will work on this in the future.

5 Related Work

Franco da Silva et al. [16] present Internet of Things Out-of-the-Box, an approach using the OASIS standard TOSCA [12] to set up IoT environments. In their work, the goals are similar to ours, setting up IoT environments with as less effort as possible. Instead of a toolbox, they are using a TOSCA Type Repository, however, the steps of defining requirements and selecting the most suitable TOSCA types are not described. The processes they use for setting up the IoT

environments do not support human tasks. Thus, hardware device setup and other manual steps should already be done in their approach.

Yelamarthi et al. [19] introduce a modular IoT architecture. Their work is similar to our approach since they use modular building blocks to create an IoT architecture. Their work shows that IoT architectures work best when they are built in a modular manner since replacing parts of the architecture becomes more easy. We built on the idea of such a modular architecture and extend it by providing a means to set it up through a holistic lifecycle method based on our toolbox containing a variety of building blocks.

In the past, many IoT environment models have been developed. These models contain information about the devices of the IoT environment, their properties (e.g., location or computing resources), the attached sensors and actuators, and their interconnection. However, these models highly differ in their content, the formats being used, and the domain they are applied to. Famous examples for such models are SensorML [13] or IoT-Lite [2]. Some of these models are developed and maintained by large organizations, others have been created in research projects and are maintained by a small group of people. Furthermore, some of them are even standardized. These models can be used to provide a standardized mean to describe the building blocks we aim for. Since many models are built based on ontologies (e.g., using the Web Ontology Language OWL [1]), important aspects, such as hierarchies and inheritance, dependencies, and attributes, are already provided. When defining the specifics of the building blocks in the future, we will consider these models.

6 Summary and Future Work

In this paper, we present an approach to guide users through the whole process of setting up IoT environments and corresponding custom IoT applications. We introduce a holistic method that covers the whole lifecycle of an IoT application. First, the requirements of the environment and its applications need to be defined by domain experts and involved stakeholders. After that, a toolbox is provided, containing a large variety of building blocks domain experts can select from. These building blocks cover different aspects of IoT environments. By plugging them together, IoT environments can be set up easily. The actual set up is then provided by processes, defining the order of the necessary steps to be conducted to set up the IoT environment based on the building blocks. After process execution, the IoT environment is successfully set up and can be adapted any time through the feedback loop in our method. Once the IoT environment is retired, the method reaches its end.

Acknowledgments

This work is financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and the research project IC4F funded by the German Ministry for Economics and Energy (01MA17008G).

References

1. Antoniou, G., van Harmelen, F.: Web Ontology Language: OWL, pp. 67–92. Springer Berlin Heidelberg (2004)
2. Bermudez-Edo, M., Elsaleh, T., Barnaghi, P., Taylor, K.: IoT-Lite: a lightweight semantic model for the internet of things and its use with dynamic semantics. Personal and Ubiquitous Computing **21**(3), 475–487 (Jun 2017)
3. Bormann, C., Castellani, A.P., Shelby, Z.: CoAP: An Application Protocol for Billions of Tiny Internet Nodes. IEEE Internet Computing **16**(2), 62–67 (2012)
4. Chinosi, M., Trombetta, A.: BPMN: An introduction to the standard. Computer Standards & Interfaces **34**(1), 124–134 (2012)
5. Chourabi, H., Nam, T., Walker, S., Gil-Garcia, J.R., Mellouli, S., Nahon, K., Pardo, T.A., Scholl, H.J.: Understanding smart cities: An integrative framework. In: 2012 45th Hawaii International Conference on System Sciences. pp. 2289–2297 (2012)
6. Costa, B., Pires, P., Delicato, F.: Specifying functional requirements and qos parameters for iot systems. In: Proceedings of the 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing. pp. 407–414 (11 2017). https://doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2017.83
7. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, Second Edition. Springer (2018)
8. Harper, R.: Inside the Smart Home. Springer Science & Business Media (2006)
9. Hunkeler, U., Truong, H.L., Stanford-Clark, A.: MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks. In: 2008 3rd Int Conf on Comm Systems Software and Middleware and Workshops (COMSWARE '08). pp. 791–798 (2008)
10. Khaled, A.E., Helal, A., Lindquist, W., Lee, C.: Iot-ddl–device description language for the "t" in iot. IEEE Access **6**, 24048–24063 (2018)
11. McEwen, A., Cassimally, H.: Designing the Internet of Things. Wiley Publishing (2013)
12. OASIS: Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0 (2013)
13. OGC: Sensor Model Language (SensorML). online, http://www.opengeospatial.org/standards/sensorml
14. Pohl, K.: Requirements Engineering: Fundamentals, Principles, and Techniques. Springer Publishing Company, Incorporated, 1st edn. (2010)
15. Shrouf, F., Ordieres, J., Miragliotta, G.: Smart factories in industry 4.0: A review of the concept and of energy management approached in production based on the internet of things paradigm. In: 2014 IEEE Int Conf on Industrial Engineering and Engineering Management. pp. 697–701 (2014)
16. Franco da Silva, A.C., Breitenbücher, U., Hirmer, P., Képes, K., Kopp, O., Leymann, F., Mitschang, B., Steinke, R.: Internet of Things Out of the Box: Using TOSCA for Automating the Deployment of IoT Environments. In: Proceedings of the 7th Int Conf on Cloud Computing and Services Science (CLOSER). pp. 358–367. SciTePress Digital Library (2017)
17. Vermesan, O., Friess, P.: Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems. River Publishers (2013)
18. Weerawarana, S., Curbera, F., Leymann, F., Storey, T., Ferguson, D.F.: Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More. Prentice Hall PTR, USA (2005)
19. Yelamarthi, K., Aman, M.S., Abdelgawad, A.: An application-driven modular IoT architecture. Wireless Communications and Mobile Computing (2017)