# Demonstration Abstract: IoTSuite: A Framework to Design, Implement, and Deploy IoT Applications

Saurabh Chauhan, Pankesh Patel, Ashish Sureka
ABB Corporate Research (India)
{saurabh.chauhan, pankesh.patel, ashish.sureka}@in.abb.com

Flávia C. Delicato
Federal University (Brazil)
fdelicato@gmail.com

Sanjay Chaudhary
Ahmedabad University
sanjay.chaudhary@ahduni.edu.in

***Abstract*—Internet of Things (IoT) application development is technically challenging and complex due to several factors such as the presence of different IoT protocols, interoperability issues, lack of industry standards and dealing with the heterogeneity that exists both in Physical and Internet worlds. Furthermore, developers involved in the IoT application development have to address issues pertaining to different life-cycles ranging from design, implementation to deployment (deployment complexities due to the possibilities of wide range of devices). Manual effort involved in all above three phases for heterogeneous devices is a time-consuming and error-prone process. We demonstrate IoTSuite which is an application development platform consisting of a suite of tools motivated by the need to simplify IoT application development. The IoTSuite supports several features such as automatic code generation and integration with a set of modeling languages. It takes high-level specification as input that abstract heterogeneity related complexity. It integrates compiler and deployment module to provide automation at different phases of application development process.**

***Index Terms*—Cyber-Physical System (CPS), Development Framework, Internet of Things (IoT)**

## I. Motivation and Aim

Internet of Things (IoT) application development is technically challenging because of some of the following key factors [1][2]

*Heterogeneity* IoT applications typically execute on a network consisting of wide range heterogeneous devices. The device characteristics vary in terms of their types (such as sensors, RFID tags, actuators, user interfaces, storage, elements of the traditional Internet such as Web and database servers), interaction modes (periodic, event-driven, request-response, notify) as well as different platforms (eAndroid, Java SE on laptops, micro-controller with no OS).

*Different life-cycle phases* Application developers have to address issues that are attributed to different life-cycles, including *design*, *implementation*, and *deployment*. During the design phase, the application logic needs to be analyzed and separated into a set of distributed tasks for the underlying network consisting of various heterogeneous entities. The tasks then have to be implemented for a specific platform of a device. At the deployment phase, the application logic has to be deployed onto a network of devices.

The above phases incurs lot of manual effort from the development team to support heterogeneous devices. The overall process is time-consuming and error-prone. To address these technical challenges, we have built upon our existing framework [1][2] and evolved IoTSuite[1] with substantial additions and enhancements suitably for modern days of IoT applications.

### A. Demo Application: Smart Home

To illustrate the characteristics of IoT applications, we take a smart home application (refer dataflow in Figure 1). A home consists of several rooms, each one is instrumented with several heterogeneous entities for providing residents' comfort, safety, and optimizing resources. To accommodate a resident's preference in a room, a database (`ProfileDB`) is used to keep the profile of each resident, including his/her preferred temperature level. An RFID reader (`BadgeReader`) in the room detects the resident's entry and queries the database service. Based on this, the thresholds used by the room devices (`Heater`) are updated. To ensure the safety of residents, a fire detection application is installed. It aims to detect fire by analyzing data from smoke (`SmokeSensor`) and temperature sensor (`TemperatureSensor`). When fire occurs, residences are notified on their smart phones (`End-User Application`) by an installed application. Additionally, residents and their neighbors are informed through a set of alarms (`Alarm`). Moreover, the system generates the current environment status on dashboard (`DashBoard`) (e.g., humidity, temperature, outside temperature by interacting with external web services `Yahoo Weather Service`) for the situation awareness.

## II. Application development using IoTSuite

Developers carry out the following steps in order to develop IoT applications using IoTSuite (refer [2] for more details).

*Specifying high-level specification* This step includes the specification of four high-level specifications (Step 1 in Figure 2): (1) *Domain specification* includes specification of resources, which includes tags (e.g., `BadgeReader`), sensors (e.g., `TemperatureSensor`), actuators (e.g., `Alarm`), and storage (e.g., `ProfileDB`) and third party web services (e.g., `Yahoo Weather Service`). (2) *Architecture specification* consists of specification of computational services (e.g., `Proximity`) and interaction among them. (3) *User interaction specification* specifies data exchange between

[1]An open source version, targeting on Android, Node.js and JavaSE - enabled devices and MQTT runtime, is available at https://github.com/pankeshlinux/IoTSuite.

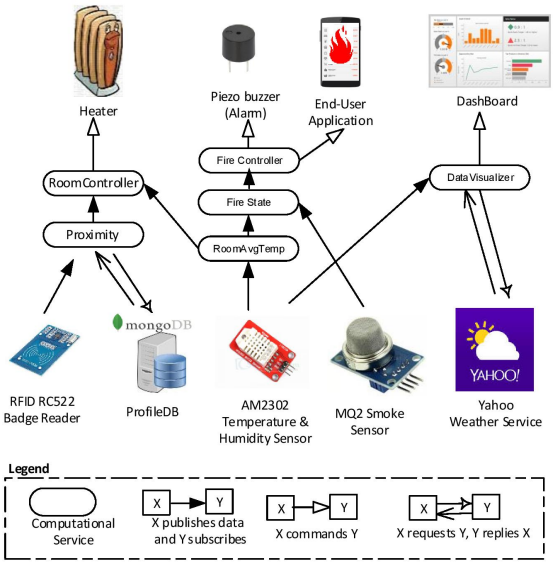


Fig. 1. DataFlow of Smart Home Application

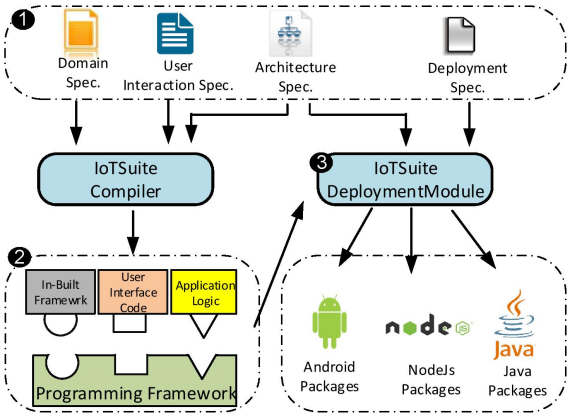


Fig. 2. Application Development using IoTSuite

an application and a user (e.g., `End-user application`, `DashBoard`). (4) *Deployment specification* describes a device and its properties in a target deployment.

*Compiling high-level specification* The compilation of high-level specifications generates a programming framework (Step 2 in Figure 2). It contains abstract classes, corresponding to each entities defined in the high-level specifications. The abstract classes contain *concrete methods*, *abstract methods*, and *interfaces*. The concrete methods hide interaction details with other entities. The *abstract methods* are implemented by the developers to write the application logic[2]. To implement *user interfaces*, the developers implement *interfaces* that connect appropriate UI elements to concrete methods of the generated programming framework.

We have integrated existing open-source sensing framework[3] for Android devices. Moreover, we have implemented sensing and actuating framework for Raspberry Pi and storage framework for MongoDB, MySQL, and Microsoft AzureDB. So, the developers do not have to implement platform-specific code.

*Generating deployment packages* This process consists of two steps (Step 3 in Figure 2). First, it takes a set of devices defined in the deployment specification and a set of computation components defined in the architecture specification and maps computational service to a device. Second, it combines the mapping output, the code generated at Step 2 and creates packages that can be deployed on devices. This stage supports the application deployment phase by producing device-specific code to result in a distributed software system collaboratively hosted by individual devices, thus providing automation at the deployment phase.

## III. Demonstration

The goal of our demonstration is to show each step of IoT application development process using IoTSuite. During the demo, we will design and implement the smart home application (discussed in Section 1) and deploy them on real devices. We have exposed IoTSuite as Eclipse plug-in (available at https://github.com/chauhansaurabhb/IoTSuite-Eclipse-Plugin) to provide end-to-end support for IoT application development.

*Specifying high-level specification using Editor* The first step of our demo is – how developers specify *domain*, *architecture*, *deployment*, and *user interaction* specifications. To write these specifications, we present editor support with features such as syntax coloring, error checking, auto completion, rename re-factoring, outline view, and code folding.

*Compiling high-level specifications and implementing an application* The second step of our demo is – how developers can compile high-level specification using IoTSuite compiler and implement the application logic and user interface code on top of the generated programming framework. On top of the generated framework, we implement the application logic and user interface code using Eclipse IDE.

*Generating deployment packages* The third step of our demo is – how developers can generate packages that can be deployed on devices. We demonstrate the deployment of these packages on real devices (shown in Figure 1) and show the execution of smart home application in the real environment.

## References

[1] S. Chauhan, P. Patel, F. Delicato, and S. Chaudhary, "A development framework for programming cyber-physical systems," *Workshop on Software Engineering for Smart Cyber-Physical Systems*, 2016.

[2] P. Patel and D. Cassou, "Enabling high-level application development for the internet of things," *Journal of Systems and Software*, vol. 103, pp. 62 – 84, 2015.

[2]We use the term application logic to refer a functionality of a software component. An example of the application logic is to open a window when the average temperature value of a room is greater than $30°C$.

[3]http://www.funf.org/