Evaluating the Ease of Application Development for the Internet of Things

Pankesh Patel, Ajay Kattepur, Damien Cassou, Georgios Bouloukakis

Evaluating the Ease of Application Development for the Internet of Things

Pankesh Patel, Ajay Kattepur, Damien Cassou, and Georgios Bouloukakis
Inria Paris-Rocquencourt, France
firstname.lastname@inria.fr

February 14, 2013

Abstract

The Internet of Things (IoT) combines Wireless Sensor and Actuator Network (WSAN) (the challenge of *large scale of systems*), pervasive computing (the challenge of *heterogeneity* of nodes and the users own interaction with these nodes), and the elements of the traditional Internet such as web and database servers. An important challenge in such a diverse and multidisciplinary field is the *ease of application development* for the stakeholders, who are involved in the IoT application development process. Several application development approaches have been proposed to address this challenge in the related field of WSANs and pervasive computing. However, very few approaches for IoT applications have evaluated their programming framework on factors such as *expressiveness* and *development effort*. The former guides the stakeholders to identify a suitable approach for given application requirements at hand. The latter helps the stakeholders the lines of code that need to be written to develop the IoT application, which involves large number of heterogeneous devices.

In this paper, we evaluate our previously proposed model-driven approach, which supports the development of IoT applications, on two factors: *expressiveness* and *development effort*. The results of the expressiveness clearly indicate the subset of IoT application characteristics that can be suitably developed in our framework. Our evaluation results of the development effort show that (1) our approach drastically reduce development effort for developing IoT applications compared to node-level programming. (2) the reusability of specification and implementation across the same application domain, thus reducing development effort.

1 Introduction

The Internet of Things (IoT) [3] applications are being deployed in a wide variety of applications and domains [1]. An important challenge, which remains to be addressed in the IoT, is the *ease of application development* for the stakeholders, who are involved in the IoT application development process. Similar challenges have already been addressed in the closely related fields of Wireless Sensor and Actuator Networks (WSANs) and pervasive computing. While the main challenge in the former is the *large scale* of the systems (hundreds to thousands of largely similar nodes), the primary concern in the latter has been the *heterogeneity* of nodes and the major role that the user's own interaction with these nodes plays in these systems (cf. the classic "smart home" scenario where the user interacts with a smart display which works together with his refrigerator and toaster). The upcoming field of IoT includes both WSANs as well as smart appliances, in addition to the elements of the "traditional" Internet such as Web and database servers, exposing their functionalities as Web services etc.

There is a growing awareness about this problem in the research community [10], and several approaches [5, 7, 10, 14–16] have been proposed in the related field of WSANs and pervasive computing. On one hand the proposed approaches provide a diverse set of functionality and, on the other hand, IoT applications have widely different characteristics. Very few approaches have addressed the ease of application development challenge with the diverse nature of the IoT applications. Very few approaches for IoT applications have evaluated their approach on expressiveness and development effort factors. Consequently, it largely remains unclear for the stakeholders, as to which approach is suitable for a given application requirement at hand.

Selecting a suitable approach for given application requirements at hand requires a clear evaluation. We believe that such assessment not only be helpful for our research, but might also provide a conceptual basis for the selection of a programming framework that can be adopted to meet requirements at hand.

Therefore, in this paper, we address these concerns by evaluating our previously proposed model-driven development approach [12]. More precisely, we answer the following questions in this paper:

1. *What are the characteristics of IoT applications that can be modeled by our model-driven approach?* In order to answer this question, we first identify some common characteristics of IoT applications (described in section 2.1) based on the survey in our previous work [13]. In section 3.1, we classify the representative IoT applications (discussed in section 2) into identified characteristics using our approach. Further, we demonstrate design decisions of selected applications using our approach.

2. *How effective is the proposed model-driven approach to promote the reuse of software artifacts to deal with changing application requirements?* In order to answer this question, we consider a scenario where there is a requirement of adding new application into existing deployment (section 3.2, experiment 1). The primary reason of choosing this scenario is to show the ability of our approach to *deal with changing application requirements*. The results of this experiment conclude that there is a drastic reduction in development effort for subsequent application development.

3. *How effective is the proposed model-driven approach to develop IoT application, involving large number of heterogeneous devices, over node-level programming?* In order to answer this question, we simulate the application over increasing number of nodes using our approach and node-level programming (section 3.2, experiment 2). The primary reason of choosing this scenario is to show the development effort over *large number of heterogeneous devices*. At the end of this experiment, we conclude that our approach reduces development effort compared to node-level programming.

2 IoT applications and its characteristics

To illustrate the characteristics of IoT applications, we first take following two domains with two applications in each domain, which are widely discussed as IoT applications [1]. Then, we identify some common characteristics in section 2.1.

- **Building Automation.**
  - A *smart office [1]* application aims to regulate appropriate temperature for worker productivity and comfort, and provides general information such as average temperature of the building. The temperature in each room of the building is regulated by a sense-compute-actuate loop executing among the temperature sensors and heaters of the room to maintain an appropriate temperature of room. Additionally, average temperature values are computed at floor and building levels to be displayed on monitors at the building entrance and control station. When a user enters or leaves a room, a badge reader detects this event, and queries a central employee database for the user's preferences. Based on the response, the threshold used by the rooms devices is updated.
  - A *fire management [4]* application aims to detect fire in building. The fire is detected by analyzing data from smoke and temperature sensors. When a fire occurs, the application triggers sprinklers and unlocks doors to allow residents to evacuate the house. Additionally, residents of the building and of the whole neighborhood are informed through a set of alarms and warning lights.
- **Traffic Management.**
  - A *road traffic monitoring and control [9]* application aims to maximize the flow of vehicle on the road. This application contains four components: (1) devices with speed sensor installed on the highway lanes, which measures and reports the speed of vehicle periodically (2) devices with presence sensor installed on the highway ramps, which reports the presence of vehicles periodically (3) devices with Electronic Message Displays (EMD), which recommends driving speed to drivers (4) devices with traffic signals, which allow or disallow cars onto the highway. First, data is gathered from speed sensors and presence sensors. Second, average speed of vehicles on the road and average queue length at highway ramp are calculated. Finally, appropriate speed is recommended to drivers on EMD and traffic signal is controlled to allow or disallow vehicles on the highway for controlling traffic.

– A *parking guidance and information [19]* application aims to provide drivers with dynamic information on parking within controlled areas of city. Monitoring devices with presence sensor are installed at parking areas to establish the flow into and out of the vehicles parks in order to calculate the number of available spaces. Vehicle park count data are processed before being presented to the public via EMDs. It shows information such as "available spaces" or "full".

2.1 Characteristics of the IoT applications

In this section, we identify some common characteristics of IoT applications

- **Goal**: An application has either sense-only or sense-compute-actuate goal.
  - Sense-Only (SO): An application with this goal collects data and stores it for later, off-line analysis. A classic example is self-logging object that stores data about itself and its environment in great quantities. An example is the *smart office* application, which records information about building such as average temperature, and stores it for later purpose. This class of applications is found in domains such as food supply chain [6, 8], patient monitoring [11], etc.
  - Sense-Compute-Actuate (SCA): An application with this goal gathers data, then processes it, and triggers appropriate action. A simple instance of this is *fire management* application, which analyzes data from smoke detctor and temperature sensors and alerts residences by triggering alarms. This class of applications is found in domains such as optimizing power consumption costs [2], work place safety[1], etc.

Classes: Sense-only or Sense-Compute-Actuate or both

- **Topology**: It indicates whether an application is characterized by static or dynamic topology.
  - In static topology, devices involved in the IoT application do not move once they are deployed. For instance, in *road traffic monitoring and control* application, device with speed sensor and presence sensor are installed on the highway.
  - In dynamic topology, devices involved in the IoT application move autonomously. The involved devices may be wearable device (e.g., smart phone, badge) to monitor or track moving objects such as person or animals. For instance, in *smart office* application, a badge is attached with the user to set his preferred temperature in a place wherever he moves.

Classes: Static topology or Dynamic topology or Hybrid

- **Scale.** It indicates that whether a sensing task of an application requires cooperation of thousands or small number of devices. For instance, in *smart office* application, calculating average temperature of room task requires very few temperature sensors while calculating daily temperature of large building task requires the use of thousands of devices with temperature sensors.

Classes: Large scale or Small scale

- **Heterogeneity**: It indicates whether an application runs on network with homogeneous devices or heterogeneous devices.
  - A network of homogeneous devices consists of mostly identical devices from hardware and software point of view.
  - A network of heterogeneous devices consists of different type of sensor and actuator. For instance, *fire managament* application consists of different kind of sensors such as temperature sensor, smoke detector and different kind of actuator such as light screen, alarm etc. These devices may be different in terms of their implementations. For instance, one temperature device produces data in Celsius and other in Fahrenheit.

Classes: Homogeneous or Heterogeneous

[1] `http://www.sensei-project.eu/`

- **Interaction modes**: It indicates how devices interact with each other. The interaction mode could be *continuous*, *event-driven*, *request-response*, or *command*.
  - Continuous. The sources communicate their data continuously at a prespecified rate.
  - Event-driven. The sources report information only if an event of interest occurs.
  - On-demand. The sources report their result only if they are requested.
  - Command. The source triggers an action of other entity with or with no argument.

  Classes: Continuous or Event-driven or On-demand or Command

- **Entity type**: It indicates whether an application consists of sensing or actuating entity of type virtual, physical or both. For instance, in *smart office* application, when a user enters or leaves a room, storage service (i.e., virtual entity) is queried for the user's preference. Based on the user's preference, heater (i.e., physical entity) is triggered.

  Classes: Virtual entity or Physical entity or both

- **End-user's role**: It indicates what role an end-user plays. A user could play role following role: *originator*, *recipient*, or *intermediary*.
  - Originator. The application could take action with human *originator*. The originator may trigger an *event* or *query* to the application. For instance, *smart office* application, when a user enters a room with its badge. A badge reader detects this event and queries a central employee database for the user's preferences. Based on these, the thresold used by the room's devices is updated.
  - Recipient. The user could play role as a *recipient*, who is notified final results by an application. For example, the *fire management* application notifies residents of the building and of the whole neighborhood through a set of alarms and warning lights.
  - Intermediary. The user could play role as *intermediary*, who is prompted for input as required.

  Classes: Originator (event/query) or Recipient or Intermediary or No role

- **Application logic**: It indicates whether an application contains simple logic or composition logic.
  - Composition logic. An application synthesizes several different measurements from heterogeneous sources. We define such an intelligence as composition logic.
  - Simple logic. An application combines largely same type of measurements from homogeneous sources. For instance, the *calculating average temperature* application aggregates sensor measurement from temperature sensors.

  Classes : Simple Logic or Composition Logic or Hybrid

3 Evaluation

This section evaluates factors of our approach such as the *expresiveness*, *required software development effort* to develop an application, which are essential for the success of any programming approach [18]. These factors are described below:

- **Expresiveness**: It indicates the scope of our approach. We have measured it by developing wide range of applications in multiple application domains such as building automation, health-care, and transportation [5] and classified these applications into identified characteristics of section 2.1. Section 3.1 shows the scope of our apporach with the reference of applications described in section 2.
- **Development effort**: It indicates the number of handwritten lines of code (LoC) required to develop an application [5]. The more hand-written code there is, the more development effort [17]. We have measured it through the number of hand-written LoC in section 3.2 with the reference of applications described in section 2.

3.1 Expresiveness

In this section, we have evalauted the scope of our approach. More precisely, we answer the following question: *What are the characteristics of IoT applications that can be modeled by our model-driven approach?*

To conduct our evaluation, we have decomposed the requirements for applications (described in section 2) of a domain into specific goals. The main advanatage of this decomposition process is that it results into elementary goals. For example, the "smart office" application of Building Automation domain is decomposed into two applications: (1) "regulating temperature" (2) "calculating average Temperature of building". Table 1 classifies sample applications according to characteristics defined in section 2.1.

Domain	**Applications**	**Goal**	**Topology**	**Scale**	**Hetero.**	**Entity type**	**Interact. patterns among devices**	**App. Logic**	**End-user's role**
Building Automation	Regulating Temperature	SCA	Static	Small	Hetero.	Physical & Virtual	Event-driven, Command, On-demand	Composition	Originator
	Calculating Average Temperature	SCA	Static	Large	Homo.	Physical	Continous & Command	Simple	Recipient
	Detecting Fire	SCA	Static	Large	Hetero.	Physical	Event-driven & Command	Composition	Recipient
Traffic Management	Road Traffic Monitoring and Control	SCA	Static	Large	Hetero.	Physical	Continous & Command	Composition	Recipient
	Parking Guidance and Info.	SCA	Static	Large	Hetero.	Physical	Continous & Command	Composition	Recipient

Table 1: Classification of IoT Applications in our Approach

Table 2 shows above discussed applications adopted in the case-studies *using our approach*. We only focus on design decisions concerning the IoT applications. Therefore, we ignore functions that are provided by middleware, device specific code, and the deployment of the physical node. When developing applications, we show two aspects of an IoT application in table 2:

1. *Can we describe basic building blocks of an IoT application ?*
2. *Can we describe components that describe application logic of an IoT application?*

The "vocabulary" column of table 2 shows the basic building blocks of an IoT application: sensing entity, actuating entity, storage, and partition definition of a system. Sensing and storage entity act as data sources and actuating entity performs an action. The column "architecture" lists the computational and controller component of an application. A computational component is fueled by sources defined in the vocabulary. It holds application logic of an applications, thus computes data from sources and then passes it to controller, which takes an action.

3.2 Development effort

This section shows how our approach eases application development. The qualitative attribute we are interested in is *development effort*. We measure this attribute through the number of hand-written Lines of Code (LoC). The following two experiments along with the applications of Building Automation domain is used to assess development effort.

- **Experiment 1:** Developing a new application in a same domain and same deployment scenario.
- **Experiment 2:** Developing an application involving large number of nodes.

Experiment 1: Developing a new application in a same domain and same deployment scenario. The primary aim of this experiment is to answer the question: *How effective is the proposed*

Domain	Application	Vocabulary				Architecture	
		Region	Sensing Entity	Actuating Entity	Storage	Computational Component	Controller
Building Automation	Regulating Temperature	`Building`, `Floor`, `Room`	`Temperature Sensor`, `Badge Reader`, `Smoke Detector`	`Heater`, `Monitor`, `Door`, `Alarm`, `Sprinkler System`, `Light`	`ProfileDB`	`Proximity`, `Calculate RoomAvgTemp`	`RegulateTemp Manager`
	Calculating Average Temperature					`Calculate RoomAvgTemp`, `Calculate FloorAvgTemp`, `Calculate BuildingAvgTemp`	`DisplayManager`
	Detecting Fire					`Calculate RoomAvgTemp`, `Calculate FireState`	`FireHandling Manager`
Traffic Management	Road Traffic Monitoring and Control	`Zone`, `Area`	`Speed Sensor`, `Presence Sensor`	`Electronic Message Display`, `Traffic Signal`	-	`Calculate AvgSpeed`, `CalculateAvg QueueLength`	`SpeedLimit Manager`, `TrafficSignal Manager`
	Parking Guidance and Information					`Calculate ParkingSpace`	`DisplayParking SpaceManager`

Table 2: List of components of application with its domain

model-driven approach to promote the reuse of software artifacts to deal with changing application requirements?

For this experiment, we consider a scenario where there is a requirement of adding a new application into existing deployment. To simulate this scenario, we have choosen three applications of Building Automation domain. As a first step, we developed "Regulating Temperature" application, then second "calculating avg. temperature" application, finally "detecting fire" application. We deployed all these applications on a set of simulated nodes running on top of a simple middleware dedicated to evaluation. Intially, when a first application was developed using our approach, we have written vocabulary spec., network spec., device drivers, and application-specific architecture specification and its application logic from scratch. The first row of table 3 shows required development effort (i.e., total LoC) for first "Regulating Temperature" application. However, the reusability of voc spec., network spec., and device drivers become apparent when we have developed subsequent applications. To develop subsequent applications, we only need to specify architecture specification and application logic. The second and third row of table 3 show that there is a drastic reduction in development effort for next two applications.

We conclude that the primary reason of drastic reduction of development effort in next two applications using our approach is *seperation of concerns*. By seperating the problem of developing an application into well-defined concerns, stakeholders can deal with them individually with changing requirements and achieve high reusability, thus reducing development effort.

Experiment 2: Developing an application involving large number of nodes. The primary aim of this experiment is to answer the question: *How effective is the proposed model-driven approach to develop IoT application, involving large number of heterogeneous devices, over node-level programming?*

To simulate this experiment, we have developed the "detecting fire" application using *our approach* and *node-level programming* (described below) and deployed it on a set of simulated nodes running on top of the middleware dedicated to evaluation.

- **Developing "Detecting fire" application using our approach.** In first deployment scenario, we have developed the application on 34 nodes instrumented with heterogeneous sensor, actuator,

Domain	Application	Lines of code					
		Vocabulary Spec.	Architecture Spec.	Deployment Spec.	Application Logic	Device-specific code	Total
Building Automation	Regulating Temperature	46	24	197	48	219	534
	Calculating Avg. Temperature	0	19	0	80	0	99
	Detecting Fire	0	35	0	58	0	93

Table 3: Required development effort to write applications in Building Automation domain

and storage service deployed in 1 building with 2 floors, each consist of 2 rooms. We measured total LoC required to develop this application. The total LoC is a summation of LoC of vocabulary spec., architecture spec., application logic, device drivers , and *deployment specification for 34 nodes*. Similarly, we have developed this application over increasing number of nodes to measure development effort. The figure 1 shows the LoC (i.e., development effort) required to develop the application over different number of nodes. In the figure 1, we have observed that lines of code increase as number of nodes increase. The reason of this increase is because of network specification. As number of nodes increase, stakeholders have to write more LoC in network specification that add into final LoC.

- **Developing "Detecting fire" application using node-level programming.** To measure development effort using node-level programming, we have calculated total LoC that need to be written for individual node, including mapping code for each software component. Similarly, we have developed the application over increasing number of nodes to measure development effort. The figure 1 shows the LoC (i.e., development effort) required to develop the application over different number of nodes. In the figure 1, we have observed that lines of code increase as number of node increase drastically because as number of nodes increase, stakeholders have to write more lines of code for individual nodes as well as more mapping code.

The results in the figure 1 shows that there is a drastic reduction in the development effort using our approach compared to node-level programming. We conclude following reasons for this achievement:

- Our approach provides abstractions to specify high-level description of IoT application at *system level*. For instance, in the table 2, one entity description for potentially many implementations and many instances. Thus, development effort does not depend on the number of nodes, which reduces development effort.
- Our approach is supported by a mapping process, which analyses the application definition and automatically generates device-specific code to result in a distributed software system collaboratively hosted by individual devices.

References

[1] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54:2787–2805, 2010.

[2] C. Buckl, S. Sommer, A. Scholz, A. Knoll, A. Kemper, J. Heuer, and A. Schmitt. Services to the field: An approach for resource constrained sensor/actor networks. In *2009 International Conference on Advanced Information Networking and Applications Workshops*, pages 476–481. IEEE, 2009.

[3] CASAGRAS EU project final report. `http://www.rfidglobal.eu/userfiles/documents/FinalReport.pdf`, 2009.

[4] D. Cassou, B. Bertran, N. Loriant, C. Consel, et al. A generative programming approach to developing pervasive computing systems. 2009.

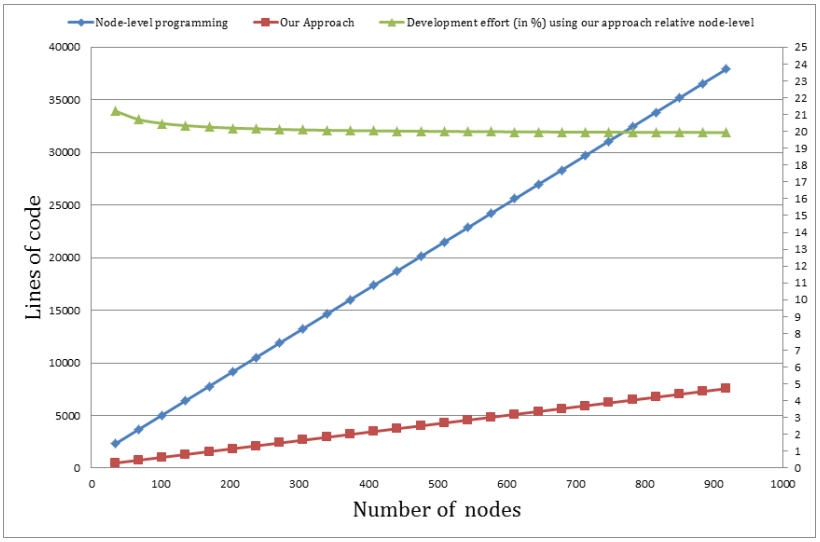

Figure 1: Developement effort in Fire detect application

[5] D. Cassou, J. Bruneau, C. Consel, and E. Balland. Towards a tool-based development methodology for pervasive computing applications. *Software Engineering, IEEE Transactions on*, (99):1–1, 2011.

[6] A. Dada and F. Thiesse. Sensor applications in the supply chain: The example of quality-based issuing of perishables. In *Proceedings of the 1st international conference on The internet of things*, pages 140–154. Springer-Verlag, 2008.

[7] A. Dey, G. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2-4):97–166, 2001.

[8] A. Ilic, T. Staake, and E. Fleisch. Using Sensor Information to Reduce the Carbon Footprint of Perishable Goods. *Pervasive Computing, IEEE*, 8(1):22 –29, jan.-march 2009.

[9] L. Mottola, A. Pathak, A. Bakshi, V. Prasanna, and G. Picco. Enabling scope-based interactions in sensor network macroprogramming. In *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, pages 1–9. IEEE, 2007.

[10] L. Mottola and G. Picco. Programming wireless sensor networks: Fundamental concepts and state of the art. *ACM Computing Surveys (CSUR)*, 43(3):19, 2011.

[11] D. Niyato, E. Hossain, and S. Camorlinga. Remote patient monitoring service using heterogeneous wireless access networks: architecture and optimization. *Selected Areas in Communications, IEEE Journal on*, 27(4):412 –423, may 2009.

[12] P. Patel. Enabling High-Level Application Development in the Internet of Things. Techreport, http://hal.inria.fr/hal-00732094, July 2012.

[13] P. Patel, A. Pathak, T. Teixeira, and V. Issarny. Towards application development for the internet of things. In *Proceedings of the 8th Middleware Doctoral Symposium*, page 5. ACM, 2011.

[14] A. Pathak and V. K. Prasanna. High-Level Application Development for Sensor Networks: Data-Driven Approach. In S. Nikoletseas and J. D. Rolim, editors, *Theoretical Aspects of Distributed Computing in Sensor Networks*, Monographs in Theoretical Computer Science. An EATCS Series, pages 865–891. Springer Berlin Heidelberg, Jan. 2011.

[15] A. Ranganathan, S. Chetan, J. Al-Muhtadi, R. Campbell, and M. Mickunas. Olympus: A high-level programming model for pervasive computing environments. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 7–16. IEEE, 2005.

[16] E. Serral, P. Valderas, and V. Pelechano. Towards the model driven development of context-aware pervasive systems. *Pervasive and Mobile Computing*, 6(2):254–280, 2010.

[17] R. Sugihara and R. Gupta. Programming models for sensor networks: A survey. *ACM Transactions on Sensor Networks (TOSN)*, 4(2):8, 2008.

[18] A. Taherkordi, F. Loiret, A. Abdolrazaghi, R. Rouvoy, Q. Le-Trung, and F. Eliassen. Programming sensor networks using remora component model. *Distributed Computing in Sensor Systems*, pages 45–62, 2010.

[19] T. A. Unit. Traffic advisory leaflet, parking guildance and information. Technical report, Department of Transport, March 2003.