Model-Driven Development for Internet of Things: Towards Easing the Concerns of Application Developers

Arpan Pal, Arijit Mukherjee(✉), and Balamuralidhar P.

Innovation Labs, Tata Consultancy Services, Kolkata, Bangalore, India
{arpan.pal, mukherjee.arijit, balamurali.p}@tcs.com

Abstract. Internet-of-Things (IoT) is poised for a disruptive growth in near future with wide and easy deployments of sensor connected to Internet. Horizontal service platforms for IoT are increasingly gaining prominence for quick development and deployment of IoT applications. However, IoT application development needs diverse skill and knowledge from domain, analytics, infrastructure and programming, which is difficult to find in one application developer. In this paper we introduce a Model-driven-development (MDD) framework that tries to address the above issue by separating out the concern of different stakeholders through models and knowledgebases.

Keywords: Iot · MDD · Knowledgebase · Meta model · Service platform

1 Introduction

The Internet of Things (IoT) has already been recognized by researchers and analysts as one of the most disruptive technologies that will transform human lives and have major economic impact. People foresee lot of penetration of IoT technology in large-scale, complex applications such as Smart Cities, Smart Transportation, Smart Manufacturing, Smart Healthcare etc. Given the complexity of the system, there is increasing requirement for IoT development platforms providing different services.

In this paper we present the case for a model-driven framework to develop and deploy IoT applications and services on top of an IoT platform. In Sect. 2, we present the requirements of such a framework and provide a technology gap analysis. In Sect. 3, we present the proposed Model-driven-development (MDD) framework for IoT platforms. Finally we summarize and conclude in Sect. 4.

2 Need for Model-Driven-Development in IoT

IoT applications, traditionally, like all other embedded applications, are built bottom up as per vertical requirements. It starts with sensor integration, moving into sensor data collection using sensor networking, storing the collected data and finally analyzing the stored data to draw actionable insights [1, 2].

However, instead of taking the bottom-up vertical approach for application development, it is beneficial to have a horizontal, platform-driven approach [3].

R. Giaffreda et al. (Eds.): IoT360 2014, LNICST 150, pp. 339–346, 2015.
DOI: 10.1007/978-3-319-19656-5_46

A model-driven-development (MDD) approach that can abstract out the meta-model of the IoT system and automate much of the application development process allowing the application developer to focus only on domain specific concerns will be of high interest to the industry.

Many of the existing IoT service platforms, support features like user management, resource provisioning, application life cycle management, device management and configuration, connectivity service provisioning and management etc. [3]. Another such platform (TCS Connected Universe Platform - TCUP) [4] tries to address a few of the above concerns by providing an integrated application development platform covering device management, data storage and management, an API based application development framework and a distributed application deployment framework.

However, none of these actually addresses the issues related to ease of application development like code reusability, need for multiple skills in domain, analytics, sensors, programming etc., or visibility of data across application.. These can be addressed through the concept of separation of concerns among different stakeholders – this has been already prevalent in the area of MDD and recently it has been also applied in the context of IoT [5, 6] which provides a framework to specify the requirements at different levels. However, this framework does not address issues like analytics algorithm re-use, distributed execution of analytics, generation of analytics and reasoning workflows, a common ontology and semantics for sensor data etc. needed for a full-fledged MDD.

On MDD systems implementation side, OASIS [12] has brought out a new standard meta model for IT services "The Topology and Orchestration Specification for Cloud Applications" (TOSCA) [13] for improving portability of cloud applications to address the challenges related to heterogeneous application environments. In TOSCA, the structure of a service is defined by the Topology Template, a directed graph, which consists of Node Templates and Relationship Templates. Node Template is an instantiation of Node Type which defines the properties and operations of a component. TOSCA uses plans, a process model defining complex workflows that can be used for the management process of creating, deploying and terminating services. TOSCA specifies an XML based syntax for defining the entities described above. For the plans it relies on existing business process modeling languages such as BPEL [14] and BPMN [15]. Some initial exploration has been reported on using TOSCA as a meta-model for IoT services [11]. From our studies we observe that this approach has a potential for addressing multiple views of the IoT application development and is much suited for offering the services following a Platform-as-a-Service (PaaS) model.

In this paper, we propose an integrated MDD framework as part of an IoT Service Platform.

3 Proposed Model-Driven Framework

A typical IoT platform caters to the needs of four different types of users or stakeholders – Applications Developers, Sensor Providers, End Users and Platform Administrators. Out of these four types of stakeholders, the focus of the proposed MDD framework is the Application Developer and the Sensor Provider. We further

sub-divide sensor providers into sensor manufacturers and sensor service providers and application developers into domain experts, application programmers and algorithm experts. In subsequent paragraphs we show how these stakeholders interact with the MDD framework.

3.1 Support Knowledgebase for MDD

Sensor Knowledgebase – The sensor knowledgebase is contributed by multiple stakeholders. The sensor device manufacturer can register the sensor in the system providing information on sensor make/model, features, operating conditions (information that is available in datasheets of sensors) along with details on sensor communication interfaces/protocols, device drivers, sensor data models etc. The sensors are normally instantiated and provisioned in the system by the sensor service provider where they add additional metadata like deployment time/location, user details etc. This knowledge base then can be used by the application developer to query specific sensors via the Sensor Explorer Interface [7].

Analytics Algorithm Knowledgebase – The analytics algorithm knowledgebase is contributed mainly by the algorithm writers and experts. It not only contains the archive of algorithm executables in form of libraries, it also contains metadata about algorithms detailing their application areas, performance parameters, accuracy, CPU complexity, memory load etc. This knowledge base can be used by the application developers to query and look for specific algorithms suitable for their application [8].

Domain-Specific Knowledgebase – Finally the application developers are primarily concerned about solving a specific set of domain problems. They are expected to be having good programming skill but limited domain knowledge, sensor knowledge and algorithm analytics knowledge. The domain-specific knowledgebase that is populated by domain experts intend to bridge this gap by providing knowledge like mapping between physical phenomenon and sensor observation, mapping between sensor application and sensor technology etc.

Infrastructure Knowledgebase – Application developers, after development of applications, need to deploy them – typically part of the applications run on edge or gateway devices collecting sensor data and part of the applications run on the cloud [9, 10]. The infrastructure knowledgebase needs to collect information about the compute/memory/communication capabilities of the available gateway devices, available gateway-to-cloud communication channels, storage capacity of the cloud and detail of available compute hardware in infrastructure of the cloud. All this knowledgebase is typically contributed by the System Administrator.

3.2 Proposed Model Driven Framework

Here we propose a model driven framework for easy development and life-cycle management of IoT applications. The motivation is to use the skills of existing IT

workforce to easily understand and develop the IoT applications using an abstract layer hiding the heterogeneity and complexity of underlying diverse technologies.

An IoT application in totality has multiple views and respective design concerns (Fig. 3). Following views can be considered to capture major design dimensions (as illustrated in Fig. 1 with their respective dependencies on knowledgebase):

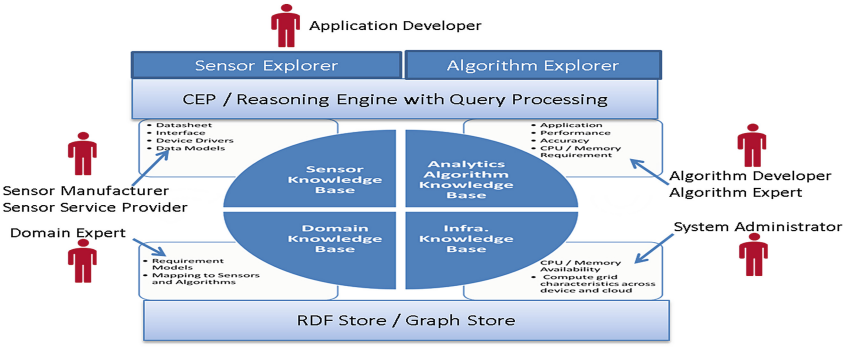

Fig. 1. Support knowledgebase structure for MDD

Information Flow and Evolution – An IoT application can be viewed as a set of data flows from sensor to sinks (actuator, database, reports, visualizer) traversing many computing operations that transforms the data to various information elements. This information flow can be modeled as a directed graph with nodes as computing modules that computes the designated information element. The edges indicate the input/output dependency relation. This also serves as a semantic model for the IoT application. It can also map to domain ontology (Sect. 3.1) through a suitable semantic mapping. Figure 2 depicts the flow graph corresponding to an example vehicle telematics service.

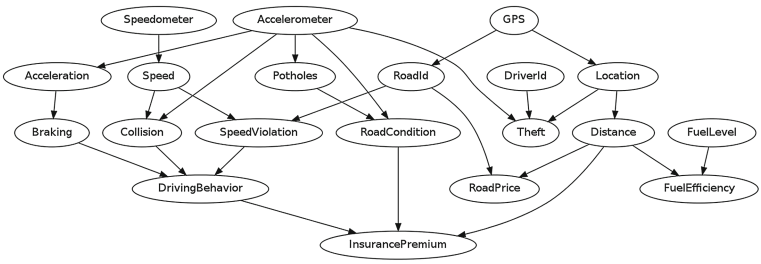

Fig. 2. Information flow graph corresponding to a vehicle telematics application - the raw sensor data is generated at the root nodes (Speedometer, accelerometer, GPS, fuel level)

Each of these nodes requires a specification of the computing model or algorithm to be used for computing the designated information element(s). There could be standard re-usable computing blocks that can be used across applications or there can be

application specific algorithms. Some of the computing operations may involve a rule engine or a complex event processing. But in general they are highly domain specific and need to be developed with the help of domain experts. The developer may seek the help of an algorithm explorer tool (introduced in Sect. 3.1) to pick the most suitable algorithm to compute the desired information element and specify it as a node property.

Node Binding to Devices – In a typical IoT application, sensing and computing may span multiple devices/platforms. For example in healthcare application a smartphone may be used to sense the vital parameters and the preprocessing, aggregation and diagnostics may be partitioned across smart phone, and cloud platform. Here we view the partitioning of the computing flow graph into sub-graphs and binding suitable computing devices to execute them. A sub-graph may be assigned to run on a Linux box, or a mobile phone. Further it may use a CEP engine or a JavaME environment to run. Specific device with its detailed specifications can be selected from the Infrastructure Knowledgebase and set the related node property. If it is computing node then the designated algorithm should have compatibility with the device chosen. If it is a sensor node then the selected device should have the required support. The respective knowledgebase (KB) described in Sect. 3.1 will be helpful in ensuring this compatibility.

Communication of Software Modules Within and Across Devices – When the computing operations are modularized, there is need for specifying the communication mechanism between these modules. The connected modules within same computing environment can use standard parameter passing mechanisms, messaging or inter-process communication primitives. Communication across devices will require external interfaces such as USB, Bluetooth, Zigbee, Wi-Fi 2G/3G etc. Further the nature of data exchange may follow models of REST, Pub-Sub, Proxy etc. The edges in the flow graph model can be used to capture the communication interface details. As in the case of device binding, the infrastructure knowledge base will be used to check the compatibility and consistency.

Security Bindings to Devices, Software and Information – The security schemes for data while communicating, storage and transformation are captured in this view. The specification is applied to a path in the sub graph spanning two end-points. Identity, credentials and security keys are also to be specified.

Deployment and Orchestration – This view captures the information to build the executable software modules, test and deploy them to the target devices. Also the specification of operational behaviors of the entire system that need to orchestrate during the operation is also specified in this view.

We need to have an underlying meta-model to represent all of the above dimensions of an IoT service and a user friendly environment to build and deploy the application from its specifications. Recently there is a development framework proposed in [6, 7] which uses Srijan language specifications as a meta-model. In our opinion they have some shortcomings in representing some of the specific contexts of IoT applications including communications, temporal behavior of the system and business process orchestrations. Probably that language can be extended to support

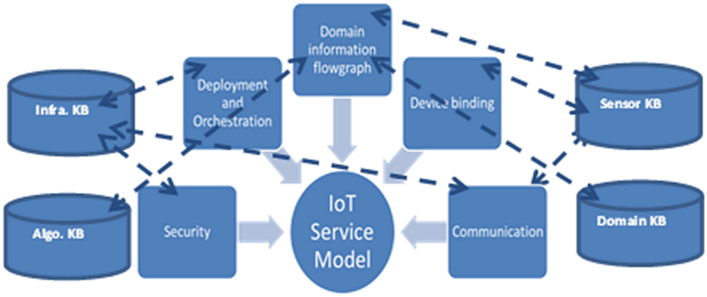

Fig. 3. Different views of an IoT application

these aspects. Here we explore the use of another meta-model specified for cloud applications.

4 Initial Prototype and Experimental Results

We have built an initial prototype which covers some of the aspects mentioned in the paper. The prototype is built and tested on a set of use-cases primarily linked with the healthcare domain. In this section, we explain the principles using one of the use-cases where a mobile phone camera is used as an optical sensor to capture the video of a person's index finger using which the heart-rate is calculated based on the concepts of PPG (photoplethysmogram) [17]. In this method, the video signal from the camera is converted to a PPG signal which is then processed following the steps of a typical signal processing workflow (as shown in Fig. 4).

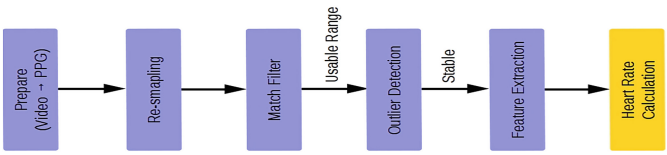

Fig. 4. Workflow showing steps to calculate heart-rate using PPG

To assist the application developer at each stage, several components have been built, although each of these is at a very nascent stage, but are capable of sufficiently underlying the concepts proposed in this paper. As a starting point, an application developer must discover sensors capable of generating a video signal which can be used for PPG. Figure 5 shows a snapshot of the SensorExplorer component (the sensor knowledgebase) which semantically links the sensors and the metadata from which the application developer can discover sensors using temporal/spatial/spatio-temporal queries [18]. We have used standard technologies such as SensorML, RDF to create ontologies for all involved entities such as sensors, algorithms etc. These graph based models enable the development of a visual programming interface where the developer

can assemble a flow graph using nodes and relations from a library. With suitable pre-built command they can test and validate the compiled application and deploy over a real infrastructure. The open source tool "Node_RED" [16] is an example of such visual tool for workflow development which we plan to use in our implementation.

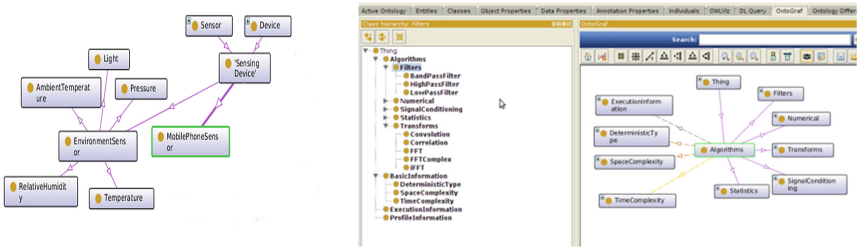

Fig. 5. Semantic linking of sensors and algorithms

Once a sensor and the corresponding data stream is discovered, the application developer creates the signal processing workflow by selecting different algorithms for each workflow step. At this stage, another component, namely the Algopedia (the algorithm knowledgebase), which is an annotated repository of algorithms (shown in Fig. 5), assists the developer to select the algorithms based on the data type, sensor type, and other algorithmic requirements which are available as metadata in the algorithm ontology. The workflow created at this step is then tested and deployed using information from the infrastructure knowledgebase and it has been found that the results are comparable to the output of similar workflows created after numerous experiments and considerable development effort by developers who are experienced in the domain. The sensor and algorithm knowledgebase semantically link information of sensors used to capture information in the healthcare domain and algorithms used to process the signals from such sensors. As of now, the domain knowledgebase is rudimentary in nature, but we plan to build it using information retrieved from several sources available over the Web.

5 Conclusion

In this paper we have introduced a MDD based framework for application development for IoT platforms. The concept of MDD specific to IoT is a recent one. Our proposed framework tries to separate and abstract out concerns of different stakeholders in IoT application development through use of models and knowledgebase, thereby improving the ease of application development. In this paper we present the concept and possible approaches to implement it. As future work, we intend to develop specific applications using the proposed framework and measure the ease of application development experienced by the developer community.

References

1. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): a vision, architectural elements, and future directions. Elsevier J. Future Gener. Comput. Syst. **29**, 1645–1660 (2013)
2. Balamurali, P., Misra, P., Pal, A.: Software platforms for Internet of Things and M2M. J. Indian Inst. Sci. Multi. Rev. J. **93**(3), 1–12 (2013). ISSN: 0970-4140 Coden-JIISAD
3. Köhler, M., Wörner, D., Wortmann, F.: Platforms for the Internet of Things – An Analysis of Existing Solutions. http://cocoa.ethz.ch/downloads/2014/02/1682_20140212%20-%20Bocse.pdf
4. Misra, P., et al.: A computing platform for development and deployment of sensor data based applications and services. Patent No. WO2013072925 A2
5. Patel, P., Morin, B., Chaudhary, S.: A model-driven development framework for developing sense-compute-control applications. In: MoSEMInA 2014, 31 May 2014
6. Patel, P., Pathak, A., Cassou, D., Issarny, V.: Enabling high-level application development in the Internet of Things. In: Zuniga, M., Dini, G. (eds.) S-Cube. LNICST, vol. 122, pp. 111–126. Springer, Heidelberg (2013)
7. Dasgupta, R.; Dey, S.; A comprehensive sensor taxonomy and semantic knowledge representation: energy meter use case. In: 7th International Conference on Sensing Technology (2013)
8. Maiti, S., et al.: Repository and Recommendation System for Computer Implemented Functions. Indian Patent Application No: 918/MUM/2014
9. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the Internet of Things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC 2012, New York, NY, USA, pp. 13–16. ACM (2012)
10. Mukherjee, A., Paul, H.S., Dey, S., Banerjee, A.: Angels for distributed analytics in IoT. In: 2014 IEEE World Forum on Internet of Things (WF-IoT), pp. 565–570. IEEE (2014)
11. Li, F., Vögler, M., Claeßens, M., Dustdar, S.: Towards automated IoT application deployment by a cloud-based approach. In: IEEE 6th International Conference on Service-Oriented Computing and Applications (SOCA) (2013)
12. Organization for the Advancement of Structured Information Standards (OASIS) https://www.oasis-open.org/
13. Topology and Orchestration Specification for Cloud Applications, V1.0. November 2013. OASIS. http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html
14. Business Process Execution Language (BPEL). https://www.oasis-open.org/committees/wsbpel
15. Business Process Model and Notation (BPMN). http://www.bpmn.org/
16. Node-Red, A visual tool for wiring the Internet of Things. http://nodered.org/
17. Pal, A., Sinha, A., Choudhury, A.D., Chattopadhyay, T., Viswanathan, A.: A robust heart-rate detection using smartphone video. In: 3rd ACM MobiHoc Workshop on Pervasive Wireless Healthcare (2013)
18. Dasgupta, R., Dey, S.: A comprehensive sensor taxonomy and semantic knowledge representation: energy meter usecase. In: 7th International Conference on Sensing Technology (ICST), pp 791–799 (2013)