A Semantic Algorithm Repository and Workflow Designer Tool: Signal Processing Use Case

Sounak Dey(✉), Dibyanshu Jaiswal, Himadri Sekhar Paul, and Arijit Mukherjee

Innovation Lab, TCS, Kolkata, India
{sounak.d,dibyanshu.jaiswal,himadrisekhar.paul,mukherjee.arijit}@tcs.com

Abstract. Recently major emphasis is exerted on development of effective tools and techniques for enriching IoT development environment. Typically an IoT application, for example a health monitoring application, not only requires domain knowledge of a programmer, but also similar knowledge from a medical practitioner, a sensor manufacturer, an infrastructure manager, etc. Such involvement of several experts makes the development process complex, resulting in escalation of time and cost of the effort. Model Driven Development (MDD) has been proposed as a development technique where such problem can be mitigated. This paper presents a system based on the MDD paradigm. As a part of the system, we present a work-flow designer framework, a visual drag and drop interface, where a developer can stitch various functional models recommended from a well-organized, annotated and crowd-sourced semantic repository of algorithms (from various domains), named as Algopedia, to quickly build a semantic workflow and in turn an end to end IoT application.

Keywords: Algorithm ontology · Algorithm repository · Semantic workflow · Workflow design · Signal processing · Model driven development

1 Introduction

As we are progressing towards the age of Internet of Things (IoT), the number of deployed connected objects across the world are increasing tremendously and is predicted to reach a count of 4.9 billion today to about 25 billion by 2020 [7]. As a consequence, an avalanche of data is hitting our servers, gateways everyday, every moment. But mere capturing such data does not make much sense unless one can analyze and find useful insights from such data and use them in IoT based applications to solve different use cases in domains like healthcare, education, transportation etc. Compared to traditional IT systems, developing such analytical applications for IoT is a difficult process as dependencies exist on a very diverse set of skills like knowledge of sensor/things, signal processing on sensor observation, knowledge of algorithms for analyzing such data, finding semantics of data etc. This requires involvment of experts of sensor, algorithm, infrastructure, programming, knowledge modelling, domain etc. Clearly,

B. Mandler et al. (Eds.): IoT 360° 2015, Part II, LNICST 170, pp. 53–61, 2016.
DOI: 10.1007/978-3-319-47075-7_7

this process involves too many stakeholders and requires application developer to know each subject to some depth. Practically this slows down the process of development.

A Model-Driven-Development (MDD) paradigm for IoT application development is one interesting approach which advocates separation of concerns among different stakeholders by introducing re-usable metamodels for IoT system and then automating the development process by stitching required models relevant to a solution; thus augmenting capability of developer and minimizing development time. This also enhances reusability of models. Pal et al. in [15] proposed a concept for MDD by creating metamodels for sensors/things, algorithms, infrastructure, domain etc. Based on that cenceptual metamodels, we have created a framework for quick application development in IoT domain. The framework consists of 1. a tool for creating and editing semantic algorithm repository and 2. a recommendation based workflow generation and execution tool. The second tool helps stitching the recommended algorithms (created using first tool) for a paticular IoT use case. In this paper, these two key developments for MDD for IoT has been discussed in details using one example of sensor signal processing use case. The rationale behind choosing a sensor signal processing use case are: (i) most IoT applications are expected to process signals received from sensors, and (ii) usefulness of semantic repository and workflow generation tool can be best exhibited by this use case as it involves many substeps like sampling, signal extracting, feature selection etc.

Following section discusses other relevant works in this field. In Sects. 3 and 4, detailed architechture of the system and its working is explained respectively while in Sect. 5 we conclude with future works.

2 Relevant Works

There are many works related to algorithm repositories and workflow designer tools. We found that most of the algorithm repositories come with a workflow designer tool associated with them; but vice versa is not always true. Algorithmia [8] is a work that falls in first category. It comes with a tool for submission of new algorithms but it lacks depth in terms of variety and count. Also it does not recommend algorithms while creating a workflow. Caiman [16] is an online algorithm repository with very limited set of image processing algorithms focussed only for cancer research. Stony Brook Algorithm repository [17] is a very comprehensive algorithm repository but it does not facilitate creation of workflow. There is an algorithm ontology called OpenTox [2] targeted to create models for detecting chemical toxicity; this can be externally connected to our ontology to enrich it, but its structure is not exhaustive and generic enough to accomodate various types of algorithms used in IoT domain.

On the second category of works, Galaxy [10] is a genome data based workflow creation and execution system but it has its proprietary execution platform thus restricting variety in workflow designing. Wotkit [9] is another online tool which helps user add sensor, capture and visualize data and create custom IoT

application based on widgets exposed as REST APIs, but does not allow any features like processing and analysis of data. There are some cloud based IoT application development frameworks like Axeda [3], BlueMix [1], ThingWorx [5] etc. which have features like device management and configuration, cloud application development, connectivity service provisioning and management; but they lack in standard algorithm repository and semantic search and recommendation during workflow designing. In the next section, overall architechture of the system is presented.

3 System Architechture

As shown in Fig. 1, the system has three main components: 1. Algopedia: an annotated algorithm repository. 2. Web Based UI for creating/modifying Algopedia. 3. A web based workflow designing and execution tool.

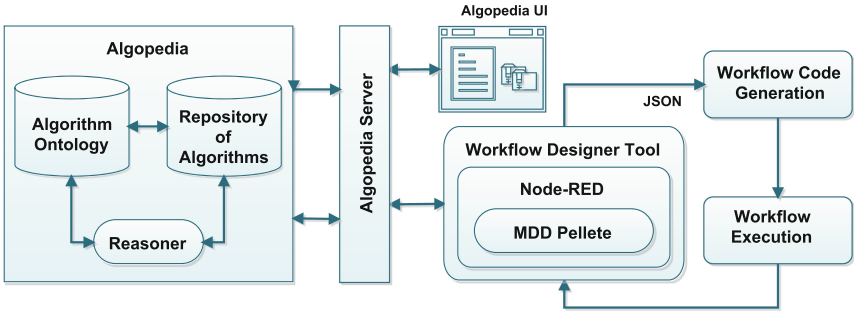


Fig. 1. Architecture diagram

3.1 Algopedia

Algopedia consists of three distinct parts: an algorithm ontology, a repository and a reasoner module. The core ontology structure has been created using Protege 4.0 [4]. Basic algorithm classes like Machine Learning, Statistical Analysis, Filtering etc. and their subclasses are captured here. Other related entities like *Features, SignalType* etc. are also defined here along with associated rules, restrictions and annotations (refer Fig. 2). This ontology contains pseudocodes associated with algorithm classes. The structure and elements of this ontology provides the basic model for algorithms. This core ontology can be enriched with more algorithm classes, relations and annotations by mining relevant information from web and by crowd sourcing from algorithm domain experts.

Based on the algorithm models in this ontology and following the pseudocodes therein, a repository of implemented working code is created. Any coder who

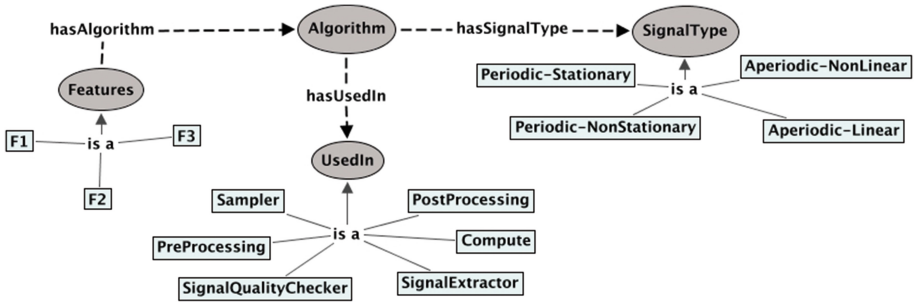


Fig. 2. Relations and entities around algorithm class

have implemented a code instance following an algorithm pseudocode can submit his/her work in this repository with proper annotations. To elaborate, algorithm class *Filter* and its subclass *BandPassFilter* are created by algorithm domain expert and is stored in Algopedia ontology along with their respective pseudocodes. If somebody implements a *bandpassfilter.c* based on *BandPassFilter* pseudocode then that particular code implementation goes to algorithm repository and stays as an individual (also called instance) of *BandPassFilter* class. Same *BandPassFilter* pseudocode can be implemented in different languages (like C, R, Java etc.) by different coder with different code complexities; the repository can accomodate each of them and all such instances will be associated with the class of *BandPassFilter*. This instance repository can again be enriched by web mining, crowd-sourcing and by feedbacks from users of the instances.

The reasoner module in Algopedia works both on algorithm ontology and repository. This module 1. validate correctness of ontology and repository after a new entry is made or after a round of enrichment via web/experts/coders and 2. can recommend a set of ranked algorithms most suitable for given criteria like signal type, feature type etc.

3.2 Web Based User Interface (UI)

Purpose of this web based user interface is to allow algorithm code writers to submit their instance of an algorithm implementation in Algopedia repository along with relevant annotation and metadata like author name, library dependencies, compiler version, input/output parameter details etc. This user interface can also be used to edit/modify aforesaid details of instances those are already there in the repository. Figure 3 shows a snapshot of this UI. The UI submits and fetches data from algorithm repository using REST APIs. This user interface is dynamically bound with the underlying Algopedia ontology. This means: the right panel in (Fig. 3) is created dynamically as per properties defined in ontology. If a property of an Algorithm class changed or a new property is added, then the changes automatically reflects back in UI as a text box or combo box (depending on the

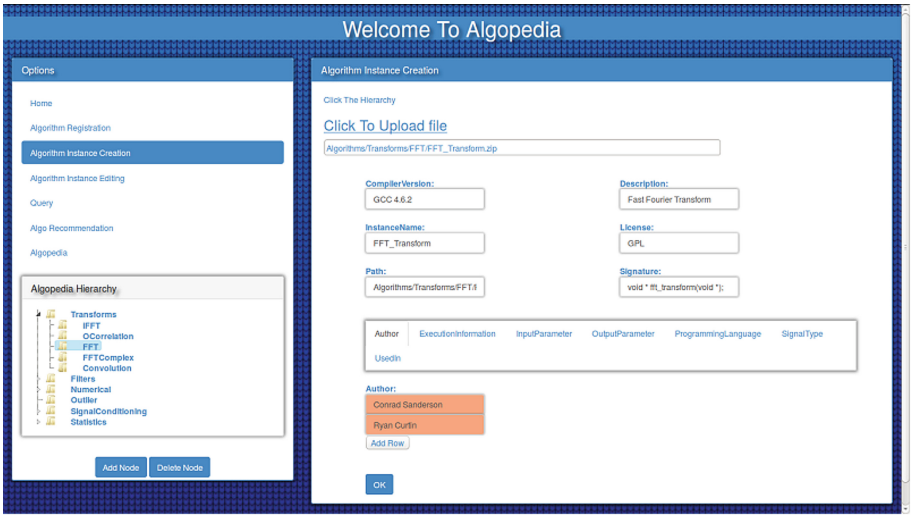


Fig. 3. Algopedia user interface

nature of the property). This is achieved without any re-coding done at user interface end. To and fro of data from repository to UI is carried in the form of JSON and the structure of JSON along with some annotations dictates how the user interface will looks.

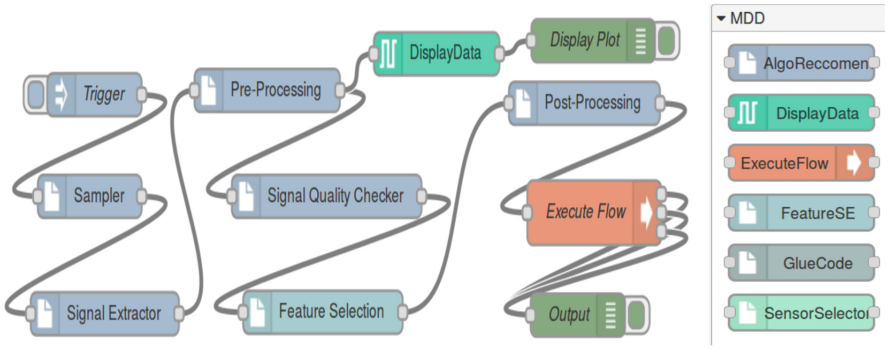
3.3 Workflow Designing and Execution Tool

To use algorithm instances for practical problems, a customised workflow generation tool based on Node-RED [6] is used. This has a web based user interface where developers can drag and drop nodes, select their choice of algorithm instances (based on recommendation) and can stitch them to complete a workflow. Once completed and deployed the workflow is converted into JSON structure, using which "Workflow Code Generation" block creates an executable code. This can be executed in a local execution platform when supplied with proper data.

In following section, a detail description of how the system works is explained using a signal processing domain use case.

4 Detail Description of Working of System

We have already mentioned that algorithm developers from various domains should contribute to the Algopedia repository via the Algopedia UI. With the Algopedia repository at hand, one can easily develop a workflow by making use of some custom nodes provided in the workflow designing tool. From the purview of an application developer in the context of a signal processing domain, an end-to-end working of the system is discussed here. A standard signal processing



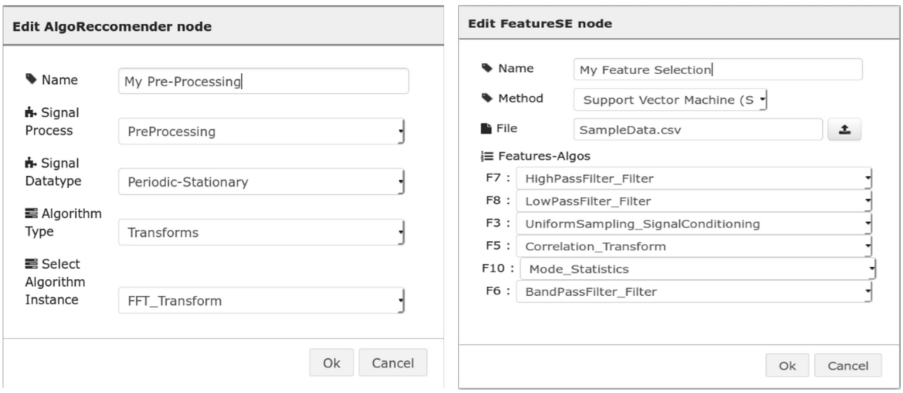
(a) Sample workflow for signal processing application

(b) MDD Custom Nodes

Fig. 4. Workflow designer tool

solution follows stages like: *Sampling* → *Signal Extractor* → *Preprocessing* → *Signal Quality Checker* → *Feature Selection and Modelling* → *Post Processing* → *Output*. Corresponding workflow may look like one shown in Fig. 4a. The above sequence of stages can be considered as a template workflow for a signal processing domain, which can be further modified by adding or deleting different nodes. Newly formed workflows can again be stored as template workflows and can be recommended to other users. In order to create this workflow, developer needs to use some (one or more) custom-nodes (Fig. 4b) like *SensorSelector, AlgorithmRecommender, FeatureSE, DisplayData* etc. from the list of custom-nodes provided in a separate palette called *MDD*, on the left hand pane of the designer tool. In ***SensorSelector*** node, user can semantically discover a sensor from a list of live sensors (capable of sending data) attached to the system and then collect data to pass on to next node in the workflow. A sensor ontology has been defined [12] at the backend to support the functionality of this node. ***AlgorithmRecommender*** node (refer Fig. 5a) enables developer to select a stage (like pre-processing, post-processing, signal quality checker etc.) of execution, the data type of the incoming signal (like periodic-stationary, aperiodic-nonstationary etc.) and the type of algorithm (from a list populated from ontology). On the basis of these three parameters, the node recommends a list of ranked algorithm instances from the Algopedia repository. Designer can select one such algorithm from this list which will be executed on the incoming data when the flow is triggered. Designer can also provide a name (like "My Pre-Processing") to identify this customised node.

The MDD palette provides another node named ***FeatureSE*** to apply feature extraction and selection on the data wherever required in the workflow (refer Fig. 5b). Feature selection can be done via a variety of methods such as SVM, PCA, MIC etc. Developers can make a choice among these options to decide how they want to train the model for their application. Each such machine learning technique requires a training data that can be uploaded by the developer to



(a) AlgorithmRecommender Node (b) FeatureSE Node

Fig. 5. Functionality of nodes

train and obtain a model corresponding to the feature selection method chosen. On the server end, the training data is used to find plausible features from the dataset and hence train the model in turn. Once the training is complete, the server returns a set of features extracted from the training dataset, along with the trained model exposed as a webservice to be used later during the execution of workflow. The set of features returned during this stage are listed along with a set of algorithms recommended by Algopedia for further processing. A combination of such *AlgorithmRecommender* and *FeatureSE* nodes are used to create the full workflow of algorithms to be executed. Along with these, ***Gluecode*** node can be used for tweaking the output of one node before entering another node as an input, or for incorporating some more computational code specific to user's need and for other similar purposes. ***DisplayData*** node is another useful node in MDD palette. This is used for visualizing data at any intermediate stage during workflow designing. This node supports plotting of data from various sources like URL, data file and previous node.

Each configured node in the workflow bundles the set of metadata information (like algorithm name and signature, dependent libraries, path of executable file etc.) in a form of JSON object and passes on to the next node in sequence. At the end, the whole JSON containing the sequence and metadata is forwarded to the ***ExecuteFlow*** node which parses it and executes them in sequence to obtain the final results. If the results seems to be satisfactory to the developer, they can register their workflow back into the algopedia repository, or else can reconfigure the algorithms and other nodes for better results. The job of the developer is to drag and drop the nodes of his/her choice to the worksheet, stitch them together in desired sequence, configure them and deploy and trigger the workflow.

5 Future Works

Our algorithm repository has good set of algorithms for machine learning, filtering, signal processing and a basic set of math libraries; but it can be enriched with other class of algorithms catering broader set of problems. Automated completion of a semantically correct and contextual workflow [13] is another aspect which could be a desired feature for our workflow designer tool. The framework described here is focussed at IoT problem domain, but it can be reused in different domains like banking, genetics, biochemistry etc. given that the domain knowledge can be modelled and incorporated into the system as envisaged in [15]. At present, execution of the workflow is done locally in a PC. Based on an algorithm requirement for computational resources the whole workflow can be partitioned into smaller units to be executed in a distributed manner across cloud, PC, handheld devices, gateways etc [14]. Works like Wings [11] which enables execution of semantic workflow in a condor based platform can guide us in this respect.

References

1. IBM Bluemix. www.ibm.com/software/bluemix/welcome/solutions2.html. Accessed 15 June 2015
2. OpenTox. http://www.opentox.org/dev/apis/api-1.1/Algorithms. Accessed 01 Sept 2015
3. PTC Axeda IoT Platform. http://www.ptc.com/axeda/product/iot-platform. Accessed 01 Sept 2015
4. Protege 4.0 (2006). http://protege.stanford.edu/. Accessed 01 Sept 2015
5. ThingWorx IoT Platform (2009). http://www.thingworx.com/. Accessed 01 Sept 2015
6. NodeRED (2013). http://nodered.org/. Accessed 01 Sept 2015
7. Gartner Says 4.9 Billion Connected Things Will Be in Use in 2015 (2014). http://www.gartner.com/newsroom/id/2905717. Accessed 01 Sept 2015
8. Algorithmia (2015). https://algorithmia.com/. Accessed 01 Sept 2015
9. Blackstock, M., Lea, R.: IoT mashups with the WoTKit. In: 2012 3rd International Conference on Internet of Things (IOT), pp. 159–166. IEEE (2012)
10. Blankenberg, D., Kuster, G.V., Coraor, N., Ananda, G., Lazarus, R., Mangan, M., Nekrutenko, A., Taylor, J.: Galaxy: a web-based genome analysis tool for experimentalists. Curr. Protoc. Mol. Biol. 19–10 (2010)
11. Deelman, E., Moody, J., Kim, J., Ratnakar, V., Gil, Y., González-Calero, P.A., Groth, P.: Wings: intelligent workflow-based design of computational experiments. IEEE Intell. Syst. **1**, 62–72 (2011)
12. Dey, S., Jaiswal, D., Dasgupta, R., Misra, A.: A semantic sensor network (SSN) ontology based tool for semantic exploration of sensor. Semant. Web Chall. Compet. ISWC (2014)
13. Grambow, G., Oberhauser, R., Reichert, M.: Semantically-driven workflow generation using declarative modeling for processes in software engineering. In: 2011 15th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW), pp. 164–173. IEEE (2011)

14. Mukherjee, A., Paul, H., Dey, S., Banerjee, A.: Angels for distributed analytics in IoT. In: 2014 IEEE World Forum on Internet of Things (WF-IoT), pp. 565–570, March 2014
15. Pal, A., Mukherjee, A., Balamuralidhar, P.: Model driven development for internet of things: towards easing the concerns of application developers. In: International Conference on IoT as a Service, IoT360 Summit, Rome (2014)
16. Reyes-Aldasoro, C.C., Griffiths, M.K., Savas, D., Tozer, G.M.: Caiman: an online algorithm repository for cancer image analysis. Comput. Methods Prog. Biomed. **103**(2), 97–103 (2011)
17. Skiena, S.: Who is interested in algorithms and why? Lessons from the Stony Brook algorithms repository. ACM SIGACT News **30**(3), 65–74 (1999)