Sustainable Interoperability and Data Integration for the IoT-Based Information Systems

Ambreen Hussain
Faculty of Arts and Technology
Staffordshire University, Stoke-on-Trent
ST4 2DE, United Kingdom
ambreen.hussain@research.staffs.ac.uk

Wenyan Wu
School of Engineering and Built Environment
Birmingham City University, Curzon Street
Birmingham, B4 7XG, United Kingdom
wenyan.wu@bcu.ac.uk

Abstract—Devices used in Internet of Things (IoT) technology are interconnected through internet and continuously send data for storage in disparate databases and files. This poses heterogeneity in data syntax and semantics. In the past few years, researches have been carried out to handle these problems in various information systems such as water and healthcare by applying semantic and syntactic uniformity. This research presents an Information and Communication Technology (ICT) framework to sustain the interoperability between stakeholders within an information system by applying Model Driven Software Development (MDD) paradigm and ontology. The major contribution of this research is building software model which is annotated with semantic model to maintain the comprehension between the modelers and the programmers allowing sustainability by adapting to the changing environment. Moreover, the framework offers two types of semantic validation i.e. model and data of the documents to integrate sensor data which we claim as a novelty in this study. The validated IoT data is integrated through RESTful web service interface of SensorThings API. The aim of this paper is to apply the methodology of the framework, on IoT based healthcare system, that has been successful in water information system.

Keywords—Model Driven Software Development; Ontology; Healthcare systems; Semantic Validation; OGC SensorThings

I. INTRODUCTION

Internet of things (IoT) is an infrastructure which allows interconnection of uniquely identified devices referred as “Things” through internet. The devices of different manufacturers have introduced the heterogeneity in the communication protocols and data format in IoT-based information systems [13]. These devices send real-time data continuously which makes it difficult to integrate and access these data [10]. The focus of this research is to deal with heterogeneity issues, allowing sustainable interoperability and integrate data form IoT-based information systems such as healthcare. In healthcare systems records of doctors, patients, nurses and medical equipments are stored in databases. Ontology based emergency medical services can be used to access, integrate and interoperate data from IoT devices [23].

Sustainable interoperability is to realize the needs of the present without compromising the ability of future changes, meeting new system requirements, and performing adequate adaptation and suitable management of the transitory elements. Semantics have not been given proper attention using model driven strategies. It is challenging for a modeler to understand a model developed by another modeler even while apprehending the same domain knowledge. Thus, it is needed to enrich the model with conceptual and structural descriptions by supplying it necessary semantic annotations to make it self-explanatory [1].

Under this research an interoperability framework was implemented for water distribution systems in Water [3] project by overcoming systems and data heterogeneity that enabled integration of sensor data from disparate resources at one large database such as Water Data Warehouse (WDW) for analysis to generate proper decisions that can support to match water supply with water demand while minimizing the energy consumption. The framework successfully gathers sensor data from clients, processes it, transforms it into SensorML1, WaterML 2.02, and O&M3 documents and insert sensor and observations using OGC SOS4 web service [15]. The framework was built using Java programming language in Eclipse platform. The question arose that, with the clients’ future requirements in terms of more functionality or different execution platform, how well these requirements will be accommodated by the current methodology of the interoperability framework. Would the framework be able to work for the new execution platform while sustaining the interoperability? To answer these questions it is envisaged that, Model Driven Software Development (MDD) with Ontology can improve our interoperability framework by first sustaining interoperability and secondly by making it generic so that it could generate desired output application of either desktop or web for more than one execution platform for a client.

The main challenge is to sustain interoperability and resolve issues at many levels which has not been given proper attention elsewhere such as interoperability between the modelers and the programmers separately. To bring the interoperability between programmers the framework will be generated in different programming languages for their executable platforms by using Model Driven Development (MDD). This means same concepts will be generated in different programming languages.

1 http://www.opengeospatial.org/standards/sensorml
2 http://www.opengeospatial.org/standards/waterml
3 http://www.opengeospatial.org/standards/om
4 http://www.opengeospatial.org/standards/sos
A programmer of one language will know those concepts and can make changes in another programming language or could communicate well with the programmer of another language because they share the conceptual understanding. The methodology brings interoperability between modelers by employing Ontology to provide semantic annotation to the model to enable modelers to understand it in the right way and provide the means to semantically validate the model and documents sent for data integration.

The achievement of sustainable interoperability is three-fold: 1) Semantic interoperability between the systems by use of ontology, 2) software interoperability between programmers by use of MDD, and 3) interoperability between modelers by use of semantic annotation.

The term annotation means to attach some data with some other piece of data [8]. Unlike classic text annotations for reader’s reference, semantic annotations are used by machines [9]. There are three types of annotations formal, informal, and ontological. Informal annotations are not machine-readable as these do not use formal languages. Formal annotations are machine readable but do not use ontological terms. In ontological annotations, the terminology has a commonly understood meaning that corresponds to a shared conceptualization called ontology [8]. The model annotation is of three types: 1) Annotation by Inheritance which is described by Is_a relationship, 2) Annotation by Partial Inheritance which is defined by Is_case_of relationship, and Annotation by Association which enables the connection of ontological entities with the model entities by association [6].

OGC SensorThings API provides an open and unified way to interconnect the internet of things (IoT) devices, data, and applications over the Web. It builds on open standards such as the web protocols, the OGC Sensor Web Enablement (SWE) standards and the Observation and Measurement (O&M). The main difference between OGC SensorThings API and OGC SOS is that the SensorThings API is designed for resource-constrained IoT devices. As a result, it follows the REST principles, JSON encoding and URL conventions [2]. The RESTful web service interface provides the typical Create, Read, Update, and Delete (CRUD) actions on uniquely-identifiable resources. The OGC SensorThings API is specifically designed for the IoT but inspired by the OASIS Open Data Protocol (OData), which defines a general-purpose RESTful service interface [4][5].

Our major contributions are sustaining interoperability between systems by bridging gap between understanding of modelers and programmers through semantic annotation. Moreover, the framework offers two types of semantic validation i.e. model and data of the documents that sent through SensorThings API to integrate IoT data which we claim as the novelty in this study.

Model driven approaches in IoT based infrastructures and Interoperability by building architectures and frameworks in different domains are described in next section. Methodology to sustain interoperability and integrate data based on MDSD is defined in section 3 and implemented on healthcare case study in section 4.

II. RELATED WORK

Object Management Group’s (OMG) framework Model Driven Architecture (MDA) [7] is widely used for MDSD. Its core idea is to abstract the core PIM (Platform Independent Model), which can completely describe the business function and have nothing to do with implementation techniques. Then multiple transformation rules are made according to different implementation techniques. PIM is transformed into Platform Specific Model (PSM) by these conservation rules and assistant tools. PSM have parameters specifying what should transformed into what technology with specific implementation techniques. Finally, the enriched PSM is transformed into the code. The purpose of MDA is to separate business modelling from underlying platform techniques by PIM and PSM and to protect the modelling result that cannot be affected by technical change. The PIM defines the conceptual model based on visual diagrams, use-case diagrams and metadata by using the standards such as unified modelling language (UML5). There are other standards defined by OMG such as object constraint language (OCL6), XML metadata interchange (XMI7), meta object facility (MOF8), Query View Transformation (QVT9) and common warehouse meta-model (CWM10). Thus, the PIM defines an application protocol in its full scope of functionality, without platform dependencies and constraints.

An architecture is developed to separate domain model from the implementation of IoT-technology linked by a tool developed using Model Driven Development [11]. Another model-driven based approach is proposed which allows IoT-based data consumers to express their vision of Data Quality (DQ) using a graphical model editor which is then transformed to generate DQ management system [12]. The issue of heterogeneity in communication protocol of IoT devices is addressed. The authors proposed an extended-SensorThings API which focuses on its tasking profile which has not been included by OGC yet [13]. A collaborative virtual environment that allows the integration of knowledge from Vehicular ad hoc networks is designed by using MDD that generates a groupware application to promote collaborative work [14]. A resource-based data access method is used to process IoT data. The authors presented an IoT-based emergency medical services to demonstrate the method [10].

Interoperability has been approached in different systems by implementing architectures and frameworks such as IDEAS [16], Athena [17], EIF [18] in enterprise systems and GridWise Interoperability Context-setting Framework [19] in electric system community.

III. METHODOLOGY

This sections presents the methodology which consists of two parts. First part defines the framework methodology and

---

5 http://www.uml.org/
6 http://www.omg.org/spec/OCL/2.4/
7 http://www.omg.org/spec/XMI/
8 http://www.omg.org/spec/MOF/
9 http://www.omg.org/spec/QVT/
10 http://www.omg.org/spec/CWM/
second part defines software design methodology. The methodology of the framework consists of following components:

- Data Access
- Data Mapping
- Data Export

The data access component of the framework collects data from clients’ data repositories using factory design pattern. It can support Web Service, databases and Excel files. The data mapping component use XML schema files and XSLT style sheets to map the data. XML is used because it is widely accepted standard. The data export component sends JSON documents to SensorThings API Service which saves these data on associated PostGreSQL database [15] [24].

The software design methodology consists of following steps which are repeated for each part of framework methodology mentioned above. This paper focuses on Data Export component only.

- Software Model
- Semantic Model
- Semantic Annotation
- Semantic Validation

A. Software Model

This step involves building models i.e. PIM and PSM for each framework component. The Metamodel of the framework is shown in Fig. 1. There are three transformation stages involved in this implementation of MDA although it supports four types of model-to-model transformations mainly: 1) PIM-to-PIM transformations involve transformations related to platform independent model refinement and are applied when PIMs are enhanced, filtered or specialized; PIM-to-PSM transformations are used when PIM is sufficiently refined to be projected to the execution infrastructure; PSM-to-PIM transformations relate to platform dependent model refinement; PSM-to-PIM transformations abstract models of existing implementations into platform independent models [20] [21].

B. Semantic Model

Semantic model is represented by building knowledgebase for the IoT based information system. This knowledgebase is built in the form of Ontology by using software such as Protégé61.

The Ontology consists of entities and relations between those entities. Ontology also consists of the rules and axioms for the better understanding of the information system.

C. Semantic Annotation

Ontology is used to annotate software model helps modelers to understand it in the right way and get the desired output. The annotation helps programmer to include the given semantics used in transformation process to get the desired output. At this stage of research, we are using, through annotations, domain information and knowledge directly in the design model. The model annotation is allowing mapping of UML class’s definition and attributes with domain ontology concepts. Our future approach for semantic annotation of the model is based on an approach [6] where authors strengthened design models by references to domain ontologies by building an annotated model. The annotated model links design model with domain ontology. Their methodology includes the verification step which is limited to constraints attached to properties and concepts in ontology that ensures if the properties defined in design model are still valid after annotation. We are further extending the verification process to validate the semantics of the resulting documents by validating the OWL/SWRL rules defined in the ontology.

D. Semantic Validation

The semantic validation of JSON documents in this research is inspired by an approach [22] to validate Water Data Transfer Format (WTDF) documents. The authors used a query reduction technique to allow the existed reasoners to validate the data. The difference is they validate after the documents are produced. Our approach validates first and then produce the documents to integrate data. This last step assures if the output JSON documents produced by Export component are semantically right and in line with the requirements. There are certain constraints need to be checked for the validation process. We consider two types of semantic validation 1) Data semantics validation of the documents, and 2) Semantic validation of the model. Data Semantic validation of the documents produced will assure if the data is semantically correct. All the OWL DL rules defined in ontology will fall into this type. Semantic validation of the model will assure if the model, transformations and code are semantically correct per requirements and align with the ontology to produce correct documents. All the ontological equivalence and restrictions fall into this type. These validations will be performed by using constraints defined in OCL to check against the model and documents during the validation process.

IV. IMPLEMENTATION OF HEALTHCARE CASE STUDY

The methodology described in previous section is applied on a healthcare case study in this section. We don’t have healthcare case study in hand to apply the methodology. We are borrowing a healthcare case study [10] for the demonstration purpose only.

---

61 http://protege.stanford.edu/
A. Software Model

In healthcare applications, sometimes patient’s data is stored in different hospital databases because a patient might go to different hospitals to see different doctors. On the cloud platform databases from different hospitals should be isolated because of security issues. But some databases should have shared data definition for data accessing [10]. This is perfect scenario for our methodology which is designed to access data from disparate resources through a single component. The data sources for emergence medical services is shown in Fig. 2 which is input to our methodology in Fig. 3. The layers of modelling data export Component are shown in Fig. 4. The layer 3 is the abstract model which is inherited by UML Meta model on the layer 2. The PIM of Export component consist of three important classes i.e. Export, Sensor, and Observation. The figure shows only Export and Sensor for demonstration. Sensor class consists of all the information related to sensor while observation class contains observational data information. While Export class collects the necessary information from these classes, convert it into JSON files and send these files to restful web service through Httpclient.

B. Semantic Model

The data model for SensorThings API is shown in Fig. 5.

The ontology shown in Fig. 6 defines entities of the case study and the relation between them through object properties.
There are physical entities in emergency medical services such as ambulance, stretcher, oxygen tank, breathing machine, hospital, room, etc. These entities are termed as Entity Oriented Resources (EoR). Some resources are combination of other physical entities such as ambulances can have stretcher, oxygen tank and breathing machine. These are defined as compositedEoR (cEoR) [10]. This can make a restriction constraint in ontology shown in Fig. 7.

C. Semantic Annotation

The semantic annotation of the PIM described above is performed by using semantic model given in previous section is shown in Fig. 8. The annotation is green and ontology is pink colored. The axioms from semantic model are explained in the annotation to help modeler and programmer to understand the model e.g. Phenomenon is the term used by sensor web community which is used as observedProperty in SensorThings API. Phenomenon is part of sensor table in database but it should be part of Observation. Similarly, it explains Ambulance is a thing which consists of a stretcher, breathing table and oxygen tank those are also things with a sensor, location and observations. A sensor should have a URI with Feature of Interest (FOI) and observations.

D. Semantic Validation

All the validation will be performed by using OCL. As described two kinds of validation in the previous section. The first kind of validation can be performed by defining an OWL rule in EMS ontology that if Observations has results and phenomenon is Temperature then the unit of this phenomenon should be in Degree Celsius (°C) shown in Fig. 9. This rule can be state in OWL DL shown in Fig. 10. Similarly, the validation rule can be performed by writing OCL statement against the model in Fig 11. The second type of validation will assure correct JSON documents are produced after the model transformations. For example, in SensorThings API the name always starts with the “at” sign (@), followed by the namespace iot, followed by a dot (.), followed by the name of the term (e.g., “@iot.id”:1) [2] showed in Fig. 12.

Figure 9 SWRL Rule in EMS Ontology

Observations

\[ \exists \text{hasPhenomenon. \{} \text{Temperature} \}\]
\[ \exists \text{hasObservationResults. Results} \]
\[ \equiv \exists \text{hasObservationResults.} \text{(Results} \]
\[ \exists \text{hasUnit. \{} \text{°C} \} \]

Figure 10 Observations Rule in OWL DL

context Observation inv:

self. Phenomenon = ‘Temperature’ implies

self. Results.Unit = “°C”

Figure 11. Observations Rule in OCL

Figure 12. Example of JSON document to create an Observation
CONCLUSION

This paper proposes an ICT framework that employs MDSD and ontology technologies to sustain interoperability by maintaining understanding between software modelers and software programmers through semantic annotations. The framework offers integration of sensor data from IoT-based information system using RESTful web service interface of SensorThings API. It also provides the means of validating the semantics of the annotated model and the JSON documents for IoT data integration. The framework is successful with the water information system and exemplified in this paper by its application on EMS in healthcare system.

FUTURE WORK

Future work includes transformation of PIM into more than one PSMs. During this process transformation rules, will be established to generate required code. OCL rules will be defined for the validation purposes. For semantic annotation, the focus is to use annotation by association in future implementation by building annotation model to link design model with semantic model.

REFERENCES