View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Bradford Scholars

Journal of Physics: Conference Series

PAPER • OPEN ACCESS

Building a Semantic RESTFul API for Achieving Interoperability between a Pharmacist and a Doctor using JENA and FUSEKI

To cite this article: T Sigwele *et al* 2019 *J. Phys.: Conf. Ser.* **1376** 012015

View the article online for updates and enhancements.

IOP ebooks™

Bringing you innovative digital publishing with leading voices to create your essential collection of books in STEM research.

Start exploring the collection - download the first chapter of every title for free.

This content was downloaded from IP address 86.180.57.67 on 05/01/2020 at 12:42

Building a Semantic RESTFul API for Achieving Interoperability between a Pharmacist and a Doctor using JENA and FUSEKI

T Sigwele[1], A Naveed[1], Y.F Hu[1], M Ali[1], J Hou[1], M Susanto[2] and H Fitriawan[2]

[1] University of Bradford, Bradford, United Kingdom
[2] Department of Electrical Engineering Faculty of Engineering, University of Lampung, Indonesia

E-Mail: t.sigwele1@bradford.ac.uk

Abstract. Interoperability within different healthcare systems (clinics/hospitals/pharmacies) remains an issue of further research due to a barrier in sharing of the patient's Electronic Health Record (EHR) information. To solve this problem, cross healthcare system collaboration is required. This paper proposes an interoperability framework that enables a pharmacist to access an electronic version of the patient's prescription from the doctor using a RESTFul API with ease. Semantic technology standards like Web Ontology Language (OWL), RDF (Resource Description Framework) and SPARQL (SPARQL Protocol and RDF Query Language) were used to implement the framework using JENA semantic framework tool to demonstrate how interoperability is achieved between a pharmacy and a clinic JENA was used to generate the ontology models for the pharmacy called pharmacy.rdf and clinic called clinic.rdf. The two models contain all the information from the two isolated systems. The JENA reasoner was used to merge the two ontology models into a single model.rdf file for easy querying with SPARQL. The model.rdf file was uploaded into a triple store database created using FUSEKI server. SPARQL Endpoint generated from FUSEKI was used to query the triple store database using a RESTFul API. The system was able to query the triple store database and output the results containing the prescription name and its details in JSON and XML formats which can be read by both machines and humans.

1. Introduction

Recently, there has been a paradigm shift in the healthcare sector as many healthcare organisations have gradually migrated paper-based patient medical records to digital electronic ones by the implementation of Electronic Health Records (EHR) systems [1]. The EHR information to be shared is stored in healthcare systems that are heterogeneously distributed. The information of these systems are in different file formats and the systems themselves are mainly proprietary [2]. These systems need to interact and be accessed in a uniform and transparent way, anywhere and anytime. Nevertheless, even with the introduction of EHR, healthcare systems are still isolated from each other with the lack of collaboration and interoperability. Healthcare interoperability can be defined as the ability of two or more distinct healthcare systems like smart hospitals, clinics, smart homes, pharmacy etc. to share the information reliably and quickly from other each other in order to operate on them together without the occurrence of errors hence improving availability [3].

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

This paper extends our previous work in [4] and proposes a semantic based healthcare collaboration framework that is able to offer a cross-system information exchange platform between different healthcare systems seamlessly that is readable by both machines and humans. The healthcare systems considered are heterogeneous and geographically distributed pharmacy and clinics that need to interact and collaborate with each other. The output of this work is a working software system tested in our laboratory that enables a pharmacist to access an electronic prescription prescribed by a doctor using semantic web technologies.

2. Related Work

There have been several efforts in the health sector to address interoperability issues, however this remains an open issue for further research. One way of solving the interoperability issue is using some standards. Some of the common healthcare standards implemented for interoperability are as follows [5]: The Health Level Seven International (HL7) standards like HL7 V2 and V3 [6] are considered the more adaptable standards. The standard developed for terminologies is called Systematised Nomenclature of Medicine - Clinical Terminology (SNOMED-CT) [7]. The standard for clinical information and patient's records include openEHR [8] and HL7 Clinical Document Architecture (CDA) [9]. Recently HL7 International organization has issued a semantic base draft standards called Fast Healthcare Interoperability Resources (FHIR) [10]. Hapi FHIR in [13] is an open-source implementation of the FHIR specification in Java. FHIR uses the concept of semantic web and elements (known as "resources") and a a Representational State Transfer (REST) Application Programming Interface (API) [11] as a means of achieving interoperability for exchanging EHR's. A RESTful API uses HTTP requests to GET, PUT, POST and DELETE data [11]. A RESTful API or RESTful web service, is an architectural style and approach to communications often used in web services development. REST technology is generally preferred to the more robust Simple Object Access Protocol (SOAP) [12] technology because REST leverages less bandwidth, making it more suitable for use on the internet. Organizations such as Google, Amazon, Netflix, and Facebook have all created REST APIs that allow third-party developers to access information from their systems so they can use it and build new applications.

3. Proposed Semantic Application Development Framework

3.1. Extension from our Previous Work

In our previous work in [14], only a case study has been presented for exchanging prescription data between a pharmacist and a clinic doctor electronically. This paper extends our previous work by practically implementing the software system to address the case study. The output from the previous work in [14] was the clinic.owl and pharmacy.owl files which were created using the ontology creation tool called Protégé [15]. Protégé supports OWL 2.0 which is the latest version of OWL. The clinic.owl file contains all the data from the clinic system database including a list of GP's and patient's data. The pharmacy.owl file contains all the data from the pharmacy database like a list of pharmacists and drugs/medications within the pharmacy. With Protégé, it is not easy to perform operations such as combining the different owl files to form one OWL file. However, using the JENA framework [16], the OWL files can be easily combined and further manipulations can be performed. The two owl files will be recreated again in JENA using JAVA. It may be possible that in future JENA can be improved to adapt the latest version of OWL to be cross compatible with Protégé.

3.2. Creating Ontology Models for Pharmacy and Clinic with JENA

The proposed framework is shown in FIGURE 1 and its modules are explained step by step in the sections below. All tools used in this paper are open-source and include JENA Framework 3.8 [16], Eclipse-Oxygen [17], WebWOWL [18], Tomcat [19], JENA FUSEKI Server [20], XAMPP Server [21], Windows Command Line and Chrome browser.

Before coding the ontologies, it is a good idea to draft a graphical view of the ontology schema. This can be drawn on a piece of paper to see how the different ontology concepts, classes and their properties relate with each other. Alternatively, an online tool called WebVOWL [18] can be used to manually

draw or load and OWL/RDF file to see its graphical presentation. The ontology models *pharmacy.owl* and *clinic.owl* have already been created from our previous work in [14] and will be combined using the JENA Framework [16]. JENA is an open source Semantic Web framework for Java. JENA has an API to extract data from and write to RDF graphs and OWL ontologies. The Eclipse-Oxygen [17] software was installed on a Windows 8.1 desktop and a new JAVA project called *JavaSemantics* was created. The latest JENA 3.8 framework libraries were downloaded and imported into the *JavaSemantics* project in Eclipse. In JENA, the *ModelFactory* API was used to build the two ontology models which will now be *clinic.rdf* and *pharmacy.rdf*. The *OntClass* API was used to create the classes for the models. FIGURE 2, FIGURE 3 and FIGURE 4 show the JAVA code for creating the clinic ontology model. The pharmacy ontology model code is not shown here due to limited space.

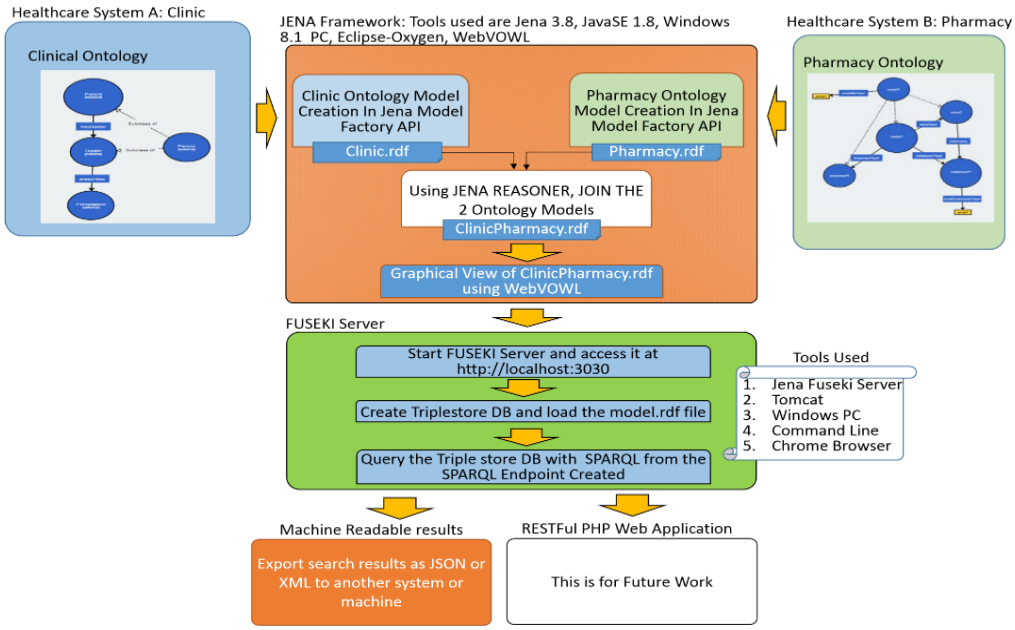

FIGURE 1. The proposed Interoperability Framework for sharing patient's prescription data between a Clinic and a Pharmacist.

3.3. Combining the Clinic and Ontology Models to form one linked RDF file

A basic reasoner was then created using the *union* operation as illustrated in the following java code, *"model = pharmacySchema.union (ClinicSchema)*". Even though the models are from different systems but since they follow the same ontology, the union operation will link the two models and merge similar classes and concepts together. The created new model can be of various formats like RDF/XML, N-Triple, Turtle, TTL, RDF JASON etc. [23]. The RDF/XML format was used to save the new model as *model.rdf* using the java code in FIGURE 5.

```
//Clinic Schema
OntModel ClinicSchema = ModelFactory.createOntologyModel (); OntClass
Person2 = ClinicSchema.createClass (s1+"Person");
OntClass Doctor = ClinicSchema.createClass (s1+"Doctor");
OntClass Patient2 = ClinicSchema.createClass (s1+"Patient");
OntClass Prescription2 = ClinicSchema.createClass (s1+"Prescription");
Person2.addSuperClass (Doctor);
Person2.addSuperClass (Patient2);
Doctor.addDisjointWith (Patient2);
Doctor.addDisjointWith (Prescription2);
Prescription2.addDisjointWith (Person2);
//Adding Properties to Clinic Schema
```

FIGURE 2. Creation of the clinic model schema using the Model Factory JENA library

```
 Prescription2.addDisjointWith (Person2);
//Adding Properties to Clinic Schema
ObjectProperty hasDoctor = ((OntModel) ClinicSchema).createObjectProperty
(s1+"hasDoctor"); ObjectProperty prescribes = ((OntModel) ClinicSchema).
CreateObjectProperty (s1+"prescribes");
hasDoctor.addDomain (Patient2);
hasDoctor.addRange (Doctor);
prescribes.addDomain (Doctor);
prescribes.addRange (Prescription2);
```

FIGURE 3. Adding data and object properties to the classes created in the clinic model.

```
//Adding Individuals
Individual janeSmith = Patient2.createIndividual (s1+"JaneSmith");
Individual prescriptionXXX = Prescription.createIndividual (s1+"PrescriptionXXX");
//Add properties to the individual
janeSmith.addProperty (hasFullName, s1+"Jane Smith"); janeSmith.addProperty
(hasPatientID, s1+"PATIENT_ID_1");
janeSmith.addProperty (hasPrescription, s1+"PrescriptionXXX");
prescriptionXXX.addProperty (hasPrescriotionName, s1+"2 doses of Medication XXX per day");
```

FIGURE 4. Creation of individuals (patient and prescription) to add to the models

```
//Writing the output as model.rdf model.write (System.out, "RDF/XML");
```

FIGURE 5. Saving the model as an RDF/XML file format

3.4. Creation of Triple Store Database to Store RDF/XML File Using FUSEKI Server

After the ontology files are created, they need to be published on the web and made accessible by end-users. This is typically done by installing a Triplestore and adding your ontology to that Triplestore. FUSEKI is a SPARQL triple store server that works with JENA. A way of querying the RDF files is using a language called SPARQL very much similar to SQL. It provides RESTstyle SPARQL HTTP Update, SPARQL Query, and SPARQL Update using the SPARQL protocol over HTTP. The JENA FUSEKI 3.8 distribution was downloaded from [20] and the windows command line (CMD) was used to start running the server on Apache Tomcat [22], an open-source Java Servlet Container developed by the Apache Software Foundation. The CMD code that was used to start the FUSEKI server is *FUSEKI-server –update –mem /ds*. The FUSEKI server was then accessed on the browser at *http://localhost:3030* and a new persistent dataset was created called *BlessU-TDB* which is used as our knowledge base or Triple store database. The *model.rdf* file was then uploaded on the *BlessU-TDB* triple store database. This then forms an RDF (FIGURE 6) graph which on the FUSEKI dashboard has 206 triples.

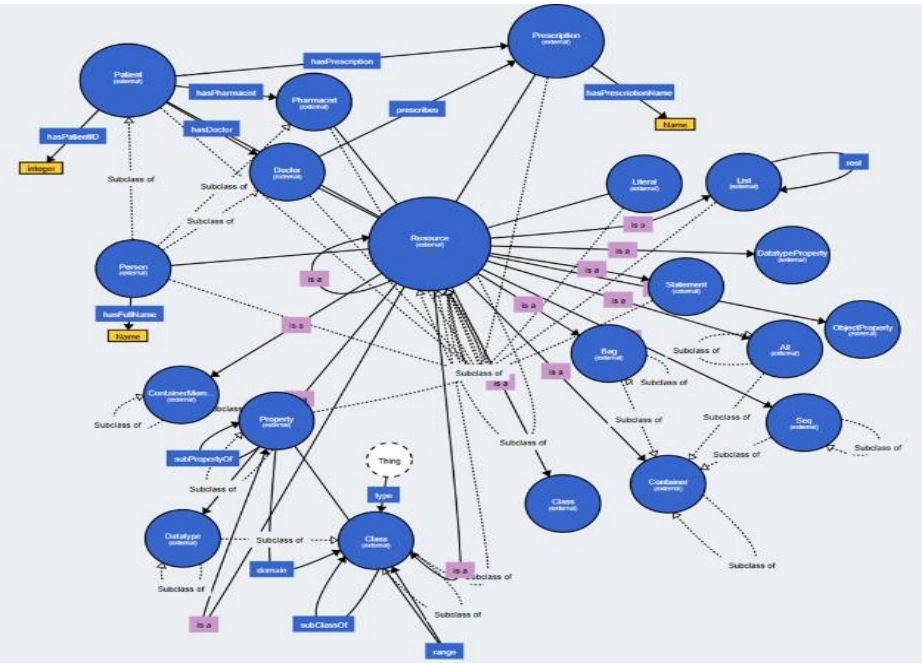

FIGURE 6. RDF Graph

The most important output of the triple store database created is the *SPARQL Endpoint* which enables users (human or other) to query a knowledge base via the SPARQL language. The SPARQL Endpoint is a URL pointing to our triple store database and in our case it is *http://localhost:3030/BlessU-TDB/query*. The triple store can be in a cloud server but in our case it is in a local machine.

4. Results and Analysis

It is the SPARQL Endpoint that is required on all triple store databases to access and query the ontologies. In our generated SPARQL Endpoint, the following query was run to access data in the *model.rdf*. This query below is executed on the pharmacist system when a patient visits a pharmacist to get the prescription. This query searches the triple store for a prescription for a patient named *Alen Brown* and only if *Alen Brown* has a prescription. The long text *www.blessuproject.semanticontology#* is the resource Uniform Resource Identifier (URI) that makes the prescription resource unique from others.

```
SELECT ?prescription
WHERE {
  <http://www.blessuproject.semanticontology#AlenBrown>
<http://www.blessuproject.semanticontology#hasPrescription> ?prescription
}
```

FIGURE 7. SPARQL query to get the prescription of a particular patient from the triple store database

Currently FUSEKI allows the results of the SPARL query in four formats; JSON, XML, CST and TSV. The output of the query in FIGURE 7 is shown below in the most popular formats of JSON (FIGURE 8) and XML (FIGURE 9). In both output formats, the results shows that *Alen Brown* has the prescription named *prescritionAAA* with the details as *5 doses of medication AAA per day*. The pharmacist can now dispense the prescription *prescritionAAA* to the patient. In future, the output will be presented in a graphical user interface like a mobile application or a browser for easier readability where a pharmacist just type the patient's name or ID in the search form and the HTML presentable results become rendered in an easy to read form by the end-user.

```
{
  "head": {
    "vars": [ "prescription" ]
  } ,
  "results": {
    "bindings": [
      {
        "prescription": { "type": "literal" , "value": "http://www.blessuproject.semanticontology#5 doses of Medication AAA per day" }
      } ,
      {
        "prescription": { "type": "literal" , "value": "http://www.blessuproject.semanticontology#PrescriptionAAA" }
      }
    ]
  }
}
```


FIGURE 8. JASON output of the prescription after running a SPARQL query on the SPARQL Endpoint.


```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="prescription"/>
  </head>
  <results>
    <result>
      <binding name="prescription">
        <literal>http://www.blessuproject.semanticontology#5 doses of Medication
AAA per day</literal>
      </binding>
    </result>
    <result>
      <binding name="prescription">

<literal>http://www.blessuproject.semanticontology#PrescriptionAAA</literal>
      </binding>
    </result>
  </results>
</sparql>
```


FIGURE 9. XML output of the prescription after running a SPARQL query on the SPARQL Endpoint.

5. Conclusion

This paper has proposed and implemented an interoperability framework in software that enables a pharmacist to access an electronic version of the prescription from the doctor with ease. Semantic technologies were used to build the framework including, OWL, RDF, SPARQL. JENA semantic tool was used to generate the ontology files and a triple store database was created using FUSEKI and a SPARQL Endpoint generated from FUSEKI was used to query the triple store database with SPARQL to get the prescription information for a particular patient prescribed by the patients doctor. The system was able to query the triple store database and output the results containing the prescription name and its details as JSON and XML formats. In the future, a graphical user interface (GUI) will be developed using PHP and the pharmacist will simply fire a web browser, login and search the patientID or name and retrieve the prescription via a web application or mobile application. FUSEKI does not currently offer security and access control itself as such in the future Access Control will have to be implemented. Security will be considered when accessing the data in the future using encryption, also tokens and passwords.

6. Acknowledgments

This work was supported by a Institutional Links grant, ID 261865161, under the Newton-Ristekdikti Fund partnership. The grant is funded by the UK Department for Business, Energy and Industrial Strategy and Indonesia Ministry of Research, Technology and Higher Education and delivered by the British Council. For further information, please visit www.newtonfund.ac.uk.

References

[1] T. Benson, *Principles of health interoperability HL7 and SNOMED* (Springer London: 2010).

[2] O. Iroju, A. Soriyan, I. Gambo, and J. Olaleke, “Interoperability in healthcare: benefits, challenges and resolutions,” in *International Journal of Innovation and Applied Studies-2013*, pp. 262–270.

[3] L. Cardoso, F. Marins, F. Portela, M. Santos, A. Abelha, and J. Machado, “The next generation of interoperability agents in healthcare,” in *International journal of environmental research and public health-2014*, pp. 5349-5371.

[4] T. Sigwele, A. S. Alam, P. Pillai, and Y. F. Hu, "Evaluating energy-efficient cloud radio access networks for 5g," in *IEEE Data Science and Data Intensive Systems-2015*, pp. 362–367.
[5] L. Cardoso, F. Marins, F. Portela, A. Abelha, and J. Machado, "The next generation of interoperability agents in healthcare," in *Procedia Technology*-2014, pp.1334–1341.
[6] HL7-International, "Introduction to HL7 Standards," in *http://www.hl7.org* (accessed March, 2018).
[7] NHS, "SNOMED-CT,*"* in *https://digital.nhs.uk/snomed-ct* (accessed March 19, 2018).
[8] openEHR.org, "an open domain-driven platform for developing flexible e-health systems," in *https://www.openehr.org/what_is_openehr* (accessed March, 2018).
[9] HL7-UK, "Clinical Document Architecture ," in *http://www.hl7.org.uk/version3group/cda.asp* (accessed March, 2018).
[10] HL7, "FHIR*"* in *https://www.hl7.org/fhir/overview.html* (accessed Aug 31, 2018).
[11] techtarget.com, "REST-API," *https://searchmicroservices.techtarget.com/definition/RESTful-API*(accessed Sept, 2018).
[12] techtarget.com, "*SOAP Simple Object Access Protocol,"* in *https://searchmicroservices.techtarget.com/definition/ SOAP-Simple-Object-Access-Protocol* (accessed March, 2018).
[13] A. Velykis, "HAPI FHIR," in *http://hapifhir.io/* (accessed Aug 31, 2018).
[14] T. Sigwele, Y. F. Fun, M. Ali, and J. Hou, "An intelligent edge computing based semantic gateway for healthcare systems interoperability and collaboration," in *IEEE FiCLOUD-2018.*
[15] protege.stanford.edu, "Protégé,*"* in *https://protege.stanford.edu/l* (accessed March 19, 2018).
[16] jena.org, "Jena," in *https://jena.apache.org/download/#jena* (accessed Sept 03, 2018).
[17] Eclipse-Foundation, "Eclipse*,"* in *https://www.eclipse.org/oxygen/* (accessed March 19, 2018).
[18] www.visualdataweb.de, "WebVOWL*,"* in *http://vowl.visualdataweb.org/webvowl.html* (accessed March 19, 2018).
[19] apache.org, "Apache Tomcat*,"* in *https://www.eclipse.org/oxygen/* (accessed March 19, 2018).
[20] apache.org, "*jena-FUSEKI,"* in *https://jena.apache.org/download/#jena-fuseki* (accessed Sept 03, 2018).
[21] ApacheFriends, "*XAMPP",* in *http://tomcat.apache.org/* (accessed March 19, 2018).
[22] apache.org, "apachetomcat" in *www.apache.org* (accessed Sept 03, 2018).
[23] apache.org*, "*RDF formats", in *www.apache .org* (accessed Sept 03, 2018).