EasyChair Preprint

№ 3489

A Model-Driven Approach for Semantic Data-as-a-Service Generation

Hela Taktak, Khouloud Boukadi, Michael Mrissa,
Chirine Ghedira-Guégan and Faiez Gargouri

May 27, 2020

A Model-Driven Approach for Semantic Data-as-a-Service Generation

Hela Taktak*†, Khouloud Boukadi*, Michael Mrissa ‡, Chirine Ghedira Guégan † and Faiez Gargouri*
Sfax University, MIRACL Laboratory, FSEG Sfax, Tunisia*
Lyon University, Lyon3 University, LIRIS UMR5205, France†
InnoRenew CoE, University of Primorska, Slovenia ‡
Email: hela.taktak@univ-lyon3.fr, khouloud.boukadi@gmail.com, michael.mrissa@innorenew.eu,
chirine.ghedira-guegan@univ-lyon3.fr and faiez.gargouri@isims.usf.tn

***Abstract*—Nowadays, with the increasing number of data sources, especially in environmental domain, earth observation programs face major challenges for environmental data exploitation, mainly due to data sources heterogeneity of different types such as access techniques, used protocols, languages, data formats, etc. Although typical solutions abstract from this heterogeneity with a layer of data services, the development of such systems remains tedious in this context. In this paper, we propose an approach based on Model-Driven Engineering (MDE) combined with semantic annotations, to automate data service development on top of data sources. Our work contributes to the development of integrated service-based architectures driven by automatic service generation, data integration from existing environmental systems and automatic service annotations. Our solution, applied to the detection of natural disasters, provides 1) appropriate modelling of data sources and services to apply model-to-text (M2T) transformations, 2) automatic generation of Representational State Transfer (REST) data service code template, 3) automatic generation of semantically annotated Hypermedia-based descriptors of these services. We have implemented and evaluated our solution with a set of real data sources provided by the Sahara and Sahel Observatory (OSS), OpenWeatherMap and CHIRPS.**

***Index Terms*—Semantic RESTful Services, Model-Driven-Engineering, Hypermedia Driven APIs.**

I. Introduction

Nowadays, climate change and environmental crisis are a major concern as noted by the United Nations (UN) through the Sustainable Development Goals (SDGs)[1]. Several SDGs relate to environmental monitoring and weather prediction, such as the efforts to combat desertification, land degradation and biodiversity loss. Therefore, ICT solutions monitoring the environmental state and predicting natural disasters are more than ever required. In fact, in the era of big data, several environmental and earth observations information systems are using and providing huge amounts of data offering several access methods (APIs, database, files, etc.), supporting various data formats (NetCDF, HDF, GRidded-Binary, etc.) and using different protocols (ODBC and JDBC for databases, HTTP for web APIs, etc.). Therefore, aggregating and exploiting data, for disasters prediction, from various data sources, rises multiple challenges. Although typical solutions abstract from data sources heterogeneity with a layer of data services, its development is usually performed manually and requires a lot of expertise. Also, semantic descriptions enabling data integration are missing. Hence, the challenge is to provide a set of automated steps for services generation and their semantically-based extensions, to support the composition processes for alerts detection purpose. Moreover, the semantic linking of the data access services enables the combination of pieces of knowledge ensured by inferences. These latter help producing warnings and real-time decisions to effectively prevent natural disasters. We, therefore, propose a generic solution for automating the deployment and use of Web services through an MDE-based approach [13], and flexible to the change of the domain knowledge and the used semantics.

Our proposal automatically generates the source code templates and the semantically annotated descriptions of RESTful [2] services from heterogeneous environmental data sources. These services will be the potential participants in further composition process for alerts detection. The remainder of this paper is structured as follows: Section II provides a scenario highlighting the need to automatically generate RESTful services source code, accessing the data sources. Section III overviews related works dealing with the integration and querying of heterogeneous data sources. Section IV describes the global architecture, before proposing a data source model and a semantic model for RESTful services, in Section V. Section VI shows how to generate semantically annotated RESTful services using the M2T transformation technique. Section VII demonstrates the applicability of our approach. Finally, we conclude and present our future work in Section VIII.

II. Motivating Scenario

Figure 1 depicts the scenario studied in the context of the PHC PREDICAT project, illustrating environmental data sources heterogeneity and their related data formats. It involves three different types and access mechanisms of data sources. The first data source CHIRPS is proposed by the Sahara

This work was financially supported by the "PHC Utique" program of the French Ministry of Foreign Affairs and Ministry of Higher Education and Research and the Tunisian Ministry of Higher Education and Scientific Research in the CMCU project number 17G1122. Michael Mrissa gratefully acknowledges the European Commission for funding the InnoRenew CoE project (Grant Agreement 739574) under the H2020 Widespread-Teaming program and the Republic of Slovenia.

[1]https://sustainabledevelopment.un.org/

and Sahel Observatory (OSS) and offers precipitation data, in different format files such as, *.BIL*, *.TIF* or *.NETCDF*. The second data source OpenWeatherMap observes the temperature and meteorological features. OpenWeatherMap offers access through HTTP using its dedicated URL. Harmonized World Soil Database (HWSD) is the third data source, made available by the OSS and renders information about soils textures'. Let us consider a user request for temperature, precipitation and soil texture in the Tunisia desert, in January 2020. In order to retrieve the results, the user should access and manipulate each data source individually, be aware of the necessary protocols and access techniques for each of the data sources. Indeed,

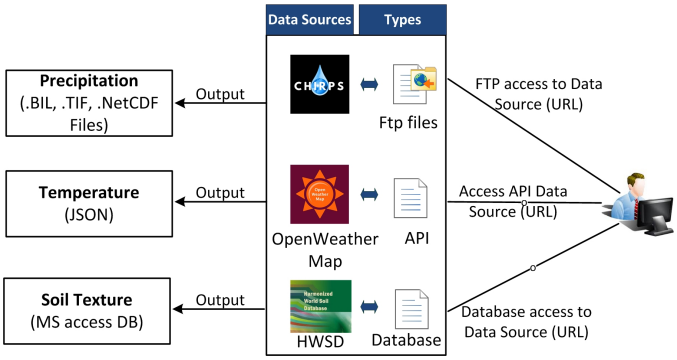

Fig. 1. Data source heterogeneity: access mechanisms and protocols.

to collect precipitation data, she should manually browse through the CHIRPS FTP server and read the relevant files. Moreover, to retrieve temperature, the user should interrogate OpenWeatherMap data source using an HTTP client (e.g: browser) and an API Key. For soil textures, data should be downloaded and browsed locally. Hence, accessing these heterogeneous data sources involves manual and tedious error-prone tasks for users. Therefore, there is a need to alleviate the burden of service development, by providing an abstraction layer based on a set of automatically generated RESTful services offering data sources uniform access (e.g: a RESTful service accessing an API or a database, etc.).

III. Related Work

Most approaches enabling interoperability, in the data integration process, use mediators and wrappers [20], [19] which are based on a unified view of the heterogeneous data. Wrappers allow accessing data sources using a common data model, whereas mediators provide an integrated view of the data provided by the data sources through wrappers. Mediators perform requests transformation into requests to the data sources. Nevertheless, these solutions need additional efforts to code them manually. Other existing approaches are Data Warehouses oriented [21] that replicate and integrate data from heterogeneous data sources maintained by online transaction processing systems. Several other solutions rely on the use of ontologies [24], [23] as a data access mechanism representing the domain of data stored in a data source. However, ontologies query answering require costly computational run-time inferences. In [16], authors presented a novel database physical operator for in-situ processing over raw data files. This solution is based on a parallel superscalar pipeline implementation, providing instant access to the heterogeneous data and integrating data, seamlessly.

Although these solutions have been proposed to cope with large-scale data heterogeneity, to our best knowledge, they do not support simultaneously modeling, linking and integrating heterogeneous data in a generic, efficient and inexpensive integration process. We, therefore, present in the following section, our proposed system architecture.

IV. Automatic Service Generator System Architecture

Our proposed system serves as an interface for the end-user to access, query the heterogeneous environmental data sources and retrieve results. The main challenge is to support a software stack able at the same time, to model heterogeneous data sources, generate automatically RESTful data access services templates, semantically annotate, link, and enhance services descriptors' with environmental domain knowledge. Our system adopts MDE to ensure data/service integration and automated service generation and annotation. Moreover, our proposed system envisions an intelligent creation of composed services participating in the deduction of new facts, such as the detection of environmental alerts, in further processes. Our system architecture, depicted in Figure 2, includes four layers: end-user, data service, semantic and data source layers. The

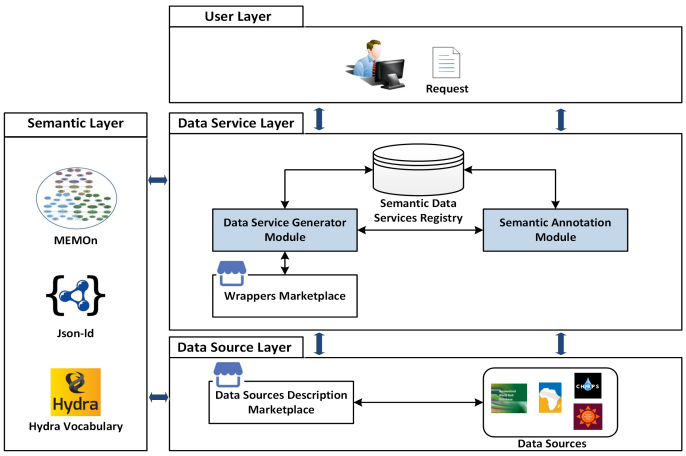

Fig. 2. Main layers of the system architecture.

end-user layer features a rich user interface that exposes data access services to the end-user. Generated services by the data service layer are presented in a lightweight portal built using Spring MVC Framework. The semantic layer includes the Modular Environmental Monitoring Ontology (MEMOn) proposed in [8]. The MEMOn ontology consists of a set of ontological modules covering sub-domains of environmental monitoring. In addition, this layer includes the JSON-LD serialization format and the Hypermedia-Driven APIs (Hydra) vocabulary, which are detailed in further sections. The data

service layer manages the registry of semantic data services. The automated service generation and semantic annotations are based on two modules: the Data Service Generator module is detailed in Section V and the Semantic Annotation module which is detailed in Section VI. The Data Service Generator is invoked when a new data source joins the system and its RESTful service is not yet implemented. This module relies on a marketplace of wrappers to handle the automatic generation of RESTful service based on an MDE approach. Wrappers are pieces of code that implement a protocol for a data source connection. The Semantic Annotation module aims at associating and tagging service descriptors using the MEMOn ontology. This module also describes and documents RESTful services through Hydra vocabulary. The data source layer encompasses several environmental data sources whose descriptions are defined in a dedicated marketplace.

V. Data Service Generator Module

The Data Service Generator Module relies on MDE principles to handle automatic model transformations. Meta-model-based transformations use only the elements of the meta-models, which are expressed in terms of the source and target meta-models [7]. In our proposed approach a source meta-model allows modelling data sources, while a target meta-model enables modelling RESTful services. The proposed meta-models and transformation rules to generate the RESTful service template are described below.

- Data Source Meta-model

The proposed meta-model aims to resolve the heterogeneity of data source types and properties by proposing an abstracted and unified structure for any data source and promotes the automatic generation of RESTful services. It is built upon the meta-model proposed in [15] and extends it with relevant concepts and relations related to data source request, response and their possible types (i.e: complex, simple), depicted in Figure 3. Since we operate with data sources on the Web, a *DataSource* is identified by an *URI* and has a *Name* attribute. The data source is queried by a *Request* and returns a *Response*. A *Response* is characterized by a *Type*, whether it could be *Simple* or *Complex*. Both *Request* and *Response* are identified and accessed by an *URI* on the Web. Each data source has a *DataModel* which represents the schema of the data. Each data source has *Characteristic* such that the *Protocol* and the *Port* attributes. These latter give information about how to access the data.

- RESTful Service Meta-model

The proposed service meta-model is built upon the MOF (Meta-Object Facility) RESTful meta-model [11]. The rationale behind using this meta-model is that it enumerates almost all the necessary concepts related to RESTful services. The *RESTService* is defined as a set of *Resources*, each identified by an *URI* and accessed by an *URIService* attribute, has a *Description* attribute and is composed of a set of *Parameters*. Each *Resource* uses a set of *Methods*. A *Method* is identified by its *Path* and has its *HTTPMethod*. The *Method* includes a

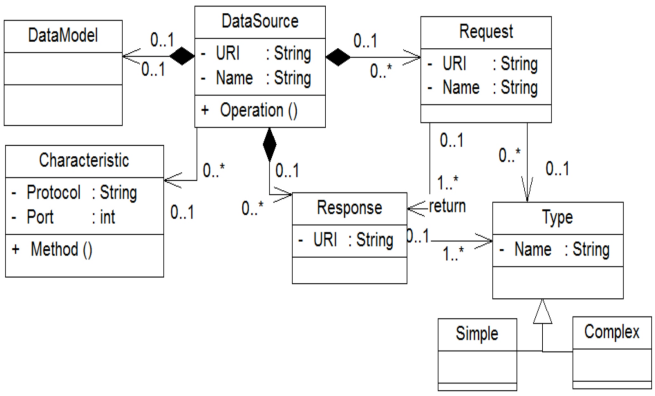

Fig. 3. Environmental data sources meta-model.

Request and produces a *Response*. Each *Request* is composed of a set of *InputParams* which are the specialization of the *Parameters*. A *Response* is composed of a set of *OutputParams* which are the specialization of the *Parameter* class. The *Parameter* has a *Type* which can be either a *SimpleType* or a *ComplexType*. Each *ComplexType* has several *Representations* (e.g: *TextReprensentation*, *JSONRepresentation* or *XMLRepresentation*). Besides the original classes, we extend this meta-model with a generic class named *Wrapper*, which is identified by its *Name* and is consumed by the *Methods* of the RESTful service. The *Wrapper* connects to any given environmental data source, given its type (i.e: API, Database, etc.) and its data-model structure. Wrappers uniformly integrate into the REST service. Figure 4 depicts a description of our extended MOF RESTful Service meta-model.

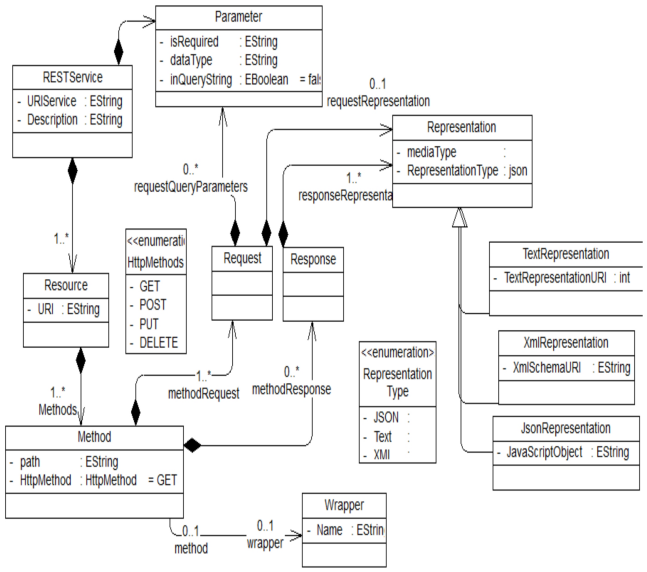

Fig. 4. MOF RESTful service meta-model.

- Transformation Rules

Transformation rules are semantic mappings between source meta-model elements to the target meta-model elements. When

executed, a new template is created, which represents the different components of a REST service generated from the data source model. These components among other, include the service name, the service path and the needed methods. For instance, each DataSource Name in the data source meta-model is transformed into a RESTService in the target meta-model. Each Data Source Operation in the data source meta-model is transformed into a Method class in the target meta-model and into a path attribute in the Method class in the target meta-model. Each Request of the data source meta-model is converted to a Request class in the target meta-model. Each Response of the data source meta-model is converted to the Response class of the target meta-model. Each Request URI in the source meta-model is converted to a Wrapper class in the target meta-model. For further example details see Section VII-A.

VI. Semantic Annotation Module

This module focuses on how to automatically generate and semantically enhance descriptors of the RESTful services with Hydra annotations. Although other alternatives to Hydra exist as cited in [25], to add descriptions to RESTful services (i.e: RESTdoc, RESTdesc, hRESTS, etc.), its advantage is that RESTful APIs state transitions are exchanged at run-time, making the client decoupled from the server. Moreover, Hydra annotations aim to facilitate services discovery and their composition for further disasters prediction processes. Indeed, these latter need relevant semantic services relationships based on schema compositions' which are driven by inferences specified by domain experts. This module follows two processes: on the one hand, the Hydra template generation process based on M2T transformation and on the other hand, the Semantic Annotation Process (SAP) of the Hydra descriptor, which is based on domain concepts matching. First, the first process mainly relies on MDE principles and automatic M2T transformations. The source meta-model represents the Web Application Description Language (WADL) [3] descriptor of the generated RESTful service and the target meta-model complies with Hydra descriptors. This transformation aims to turn a WADL descriptor into a machine-processible Hydra descriptor. Hydra descriptors based-vocabulary aim to describe the service functionalities in a Hypermedia-driven API, where all the resources are navigable and linked through URLs. Thus thanks to Hydra, the RESTful service is seen as a documented Web-API and Hydra operations describe the functionalities provided by this Web-API. Second, SAP consists in linking the output parameters of the service with ontology concepts describing the information that the service provides. In our approach, the MEMOn ontology [8] is used as the main source of environmental background knowledge. Figure 5 depicts the processes to automatically generate the Hydra template and enhance it with semantic annotations.

A. M2T Hydra Template Generation

In what follows, we detail M2T transformation rules to generate the Hydra annotated service descriptor for a RESTful service.

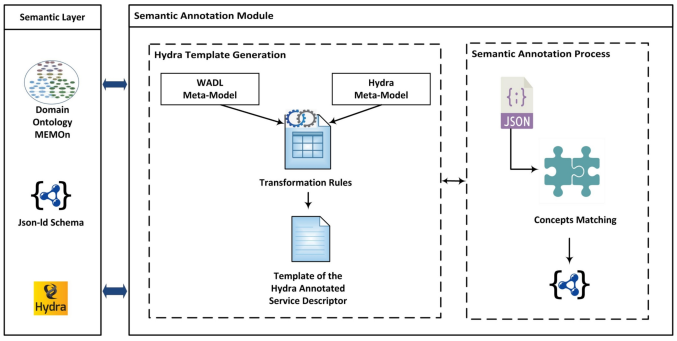

Fig. 5. Semantic Annotation Module.

- Service Descriptor Meta-model (WADL Meta-model)

The WADL document defines a contract specifying how to use the different set of resources of the service. It encompasses an *application* which contains a set of *resources*. Each *resource* class contains a set of *resources*. Each *resource* includes the *method* issued by the service request using the HTTP method. Each *method* produces a *response* which has a *representation*. The *representation* of the output of the RESTful service could be either in JSON or in XML format.

- Hydra Meta-model

The target meta-model is designed for the generation of a Hypermedia-driven API that describes and documents the RESTful service resources and operations through the Hydra vocabulary. This meta-model contains the *apiDocumentation* class, which represents the Hypermedia API documentation related in our case to a RESTful service. The *apiDocumentation* is seen as a *resource* and supports a set of *operations*. The *context* attribute is seen as a reference to the JSON-LD which is a semantically annotated document integrating the shared environment of terms from MEMOn. The *id* attribute designates the name of the service which is associated to the API documentation. The *entrypoint* is the URL to access the API Documentation. The *type* attribute refers to the type of an API documentation, which is either (e.g: Hydra:class, Hydra:link, Hydra:property) defined in the Hydra Core Vocabulary. An *apiDocumentation* is composed of dereferenceable resources named as *links*. A *link* is composed of *supportedOperations*. Each *supportedOperation* is composed of a set of *operations*, each of which is associated to a *Property* having an *id* attribute which consists of one of the HTTP methods (e.g: GET, POST, DELETE, PUT). An *operation* is identified by the *id* attribute which references the method name of the RESTful service. The *type* of the *operation* is hydra:link.

- Transformation Rules

Transformation rules are mapping relations between meta-model elements of the WADL source meta-model to the Hydra target meta-model. For instance, each *resource* class which is the basic class of the WADL meta-model is transformed into a *resource* class in the Hydra meta-model. Each *resource* in the WADL meta-model is transformed into a *link* class in the

Hydra meta-model. Each *method* in the WADL meta-model is transformed into an *operation* class in the Hydra meta-model.

B. Semantic Annotation Process

In order to semantically enhance Hydra service descriptor, the semantic annotation process (SAP) is proposed. SAP mainly relies on two inputs: the JSON document consisting of the Representation class corresponding to the Response class of the executed RESTful service and the MEMOn [8] domain ontology. The JSON document contains the pairs (key/value) of the service output concepts, which are named entities. We implemented SAP through a dedicated Python script which uses the Owlready2[2] library to access and manipulate OWL ontologies. This script retrieves associated results in a linked-data perspective from MEMOn. This association is called the matching process parsing entities of the JSON document, performing string similarity measures to find concepts from MEMOn and associating each matched entity with its linked IRIs in MEMOn modules. There are several similarity measures proposed in the literature, such as Cosine similarity [18], Jaccard similarity measure [18] and Levenshtein similarity measure [4]. The matching in SAP is performed when the maximum score is computed among the three denoted similarity measures. Hence, once the named entity is matched, its type (i.e: "@type" : "owl:Class") is mentioned in the JSON-LD file of the current RESTful service. Moreover, we enhanced the JSON-LD by matched concepts with their corresponding IRIs from the MEMOn ontology. Thereafter, the last step of SAP consists in integrating the JSON-LD path to the Hydra descriptor. Thus, the Hydra annotated descriptor is semantically-enhanced with concepts from the MEMOn ontology. For instance, consider the "temperature" concept extracted from the JSON document. When the Temperature concept is matched with its equivalent in MEMOn, it is explicitly annotated by the concept "MeasurementUnitofTemperature" as an "Owl:Class" type and identified by its URI specified in MEMOn. This latter has a value as a "DegreeCelsiusMeasure". Finally the concept "temperature" is assigned the same numerical value returned by the RESTful service.

VII. Implementation and Evaluation

In this Section, we present the feasibility of our MDE-based approach by an implementation example of a RESTful service generation accessing an API data source. In particular, we demonstrate the effectiveness of the Data Service Generator module. First, we provide an implementation example of a RESTful service generation accessing an API data source. Second, we focus on the evaluation of the semantic matching process using a Benchmarking annotation tool GERBIL [14], which is a general framework for benchmarking semantic entity annotation systems. More information about GERBIL are on the project webpage[3] and code repository[4].

[2]https://pypi.org/project/Owlready2/

[3]http://gerbil.aksw.org

[4]https://github.com/AKSW/gerbil

A. RESTful Service Generation

To demonstrate the effectiveness of the Data Service Generator module, we chose to implement the case of an API data source "OpenWeatherMap". It searches and returns the temperature in a given city or by geographical coordinates. To generate the RESTful service, in a first step, developers need to use a software modeler to design the OpenWeatherMap data source model which conforms to the data source meta-model. In a second step, Acceleo [5] tool applies the M2T transformation rules previously defined to generate the the RESTful service source code which conforms to the RESTful service meta-model. The "API" class in the data source model is transformed into the class of the RESTful service which is the concatenation of the API name class in the data source model with the term "service". Moreover, the service is annotated with its path (@Path="APIname"), which is the concatenation of the name of the API class in the data source model with the term "name". The class "API" contains the method "getTemperature" in the data source model. It is mapped in the generated code of the RESTful service to its method name, which is identical to the name of the method in the data source model "getTemperatureByCity" and its path is annotated with the name of the method (@Path="getTemperatureByCity"). The "methodType" in the API model is mapped into the "outputParam" in the generated code of the RESTful service to produce the output in the "MediaType.APPLICATION_JSON" representation form. The method in the service is named "getTemperatureByCity", which calls the "getToken" method contained in the Wrapper module which retrieves a generated token (i.e: session ID) for the user using the API. This method generates the temperature by a city name. The second method in the RESTful service is "getTemperatureByCoord" which generates the temperature by the geographical coordinates. Figure 6 depicts the automatically generated RESTful service

```
import javax.ws.rs.Produces;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.client.*;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.WebTarget;
@path("APIname")
public class APIService
{
@Get
@Path("getTemperatureByCity")
@Produces(MediaType.APPLICATION_JSON)
public String getTemperatureByCity(String city)
{
    APIWrapper WAPI=new APIWrapper();
Client c= ClientBuilder.newClient();
String Buffer sb=new StringBuffer();
sb.append(url);
String token = WAPI.getToken();
String url = "https://api.openweathermap.org/data/2.5/weather?q="+city+"&appid="+token;
WebTarget target= c.target(sb.toString());
return target.request().get(String.class);
}
@Get
@Path("getTemperatureByCoord")
@Produces(MediaType.APPLICATION_JSON)
public String getTemperatureByCoord(int Lat,int Long)
{
    APIWrapper WAPI=new APIWrapper();
Client c= ClientBuilder.newClient();
String Buffer sb=new StringBuffer();
sb.append(url);
String token = WAPI.getToken();
String url = "https://api.openweathermap.org/data/2.5/weather?lat="+Lat+"lon="Long+"&appid="+token;
WebTarget target= c.target(sb.toString());
return target.request().get(String.class);
}
}
```

Fig. 6. Excerpt of the generated RESTful service for an API data source.

for an API data source.

B. Evaluation of the Semantic Annotation Module

To assess the evaluation of the Semantic Annotation module, we used a dedicated benchmarking framework called GERBIL which allows evaluating an annotation tool against other annotation systems. It provides an enhanced entity matching which comprises the following steps: (1) URI set retrieval, (2) URI checking, (3) URI set classification and (4) URI set matching. Moreover, it uses traditional information retrieval measures, namely precision, recall and F-measure. We tested two types of environmental data sources available on the Web: the OpenWeatherMap API, and CHIRPS FTP which accesses large volumes (i.e: NETCDF file, 477MB). The OpenWeatherMap API[5] gives access to current weather data for any location including over 200,000 cities from the OpenWeatherMap dataset. Data comes from more than 40,000 weather stations and is available in JSON, XML, or HTML formats through parameters in the URL. We performed the generation of their accessing RESTful services and retrieving the requested concepts by the user. We, then, performed a preliminary evaluation of the effectiveness of SAP. We compared the performance of our annotator with that of some of the GERBIL annotations systems'. Table I depicts the

TABLE I
SYSTEM PERFORMANCE MEASURES

Strategy	Precision	Recall	F-Measure
SAP	0.467	0.244	0.320
DBpedia Spotlight	0.489	0.249	0.329
Babelfy	0.134	0.096	0.111
AIDA	0.324	0.112	0.166

different system annotators performance measures: the DBpedia Spotlight [9] relies on a vector-space representation of entities and using the cosine similarity measure. The Babelfy [10] relies on terms disambiguation in-link graph algorithm to tackle the word sense disambiguation and entity linking tasks. The AIDA approach [17] relies on graph building and dense graph algorithm. We observed that measures related to our system are almost close to that of the DBpedia Spotlight since they both use the same similarity method. We noticed that both Babelfy and AIDA are different approaches with different measurements compared to SAP. The results of our experiments showed that SAP module performs better than the other methods relying on graph-building and graph algorithms.

VIII. CONCLUSION

In this paper, we showed through our proposed approach how appropriate modelling according to MDE and combined with M2T transformations can automate the generation of RESTful services. Moreover, we showed how to automatically generate Hydra descriptors and to add semantics in a linked-data perspective describing RESTful services as a Hypermedia-Driven APIs, in order to facilitate services discovery and their composition for eventual disasters prediction processes. In addition, our proposal provides a user-friendly environment for non-expert end-users, acting as a black-box gumming the stringent difficulties connecting to multiple environmental data sources, on the Web. Future work includes exploring how data services can be composed following a REST services composition driven by Hypermedia.

[5]https://openweathermap.org/api

REFERENCES

[1] R.Alarcón and E.Wilde. Restler:crawling restful services. In *Proc. 19th Int.Conf on WWW*, pp., 1051-1052, ACM, 2010.

[2] R.T.Fielding, *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, UNIV. OF CALIFORNIA, 2000.

[3] M.Hadley. Web application description language. "https://www.w3.org/Submission/wadl/", 2009.

[4] P.A.V.Hall and Geoff.R.Dowling. Approximate string matching. *In ACM Comput. Surv.*, pp.,381-402, December 1980.

[5] N.Kahani, M.Bagherzadeh, J. R.Cordy, J.Dingel, and D.Varró. Survey and classification of model transformation tools. *(SoSyM)*, 2018.

[6] M.Lanthaler. Hypermedia driven-api. ''https://www.hydra-cg.com/spec/latest/core/", 2019.

[7] T.Levendovszky, G.Karsai, M.Maroti, A.Ledeczi, and H.Charaf. Model reuse with metamodel-based transformations. *Software Reuse: Methods, Techniques and Tools*, Springer, pp.,166-178, Berlin, Heidelberg, 2002.

[8] M.Masmoudi, S.BA.Ben Lamine, H.Baazaoui-Zghal, MH.Karray, and B.Archimede. An ontology-based monitoring system for multi-source environmental observations. Procedia Computer Science, pp.,1865-1874, KES, 2018.

[9] P.N.Mendes, M.Jakob, A.G.Silva, and C.Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *Proc. of the 7th Int.Conf on Semantic Systems (I-Semantics)*, 2011.

[10] A.Moro, A.Raganato, and R.Navigli. Entity linking meets word sense disambiguation: a unified approach. *Trans. of the Association for Computational Linguistics*, pp.231-244, 2014.

[11] A.Navarro and A.da Silva. A metamodel-based definition of a conversion mechanism between soap and restful web services. *Comput.Stand. Interfaces*, 48(C), pp.49-70, 2016.

[12] Acceleo m2t transformation tool. "http://www.eclipse.org/acceleo/", 2006.

[13] OMG. About the object management group, 2000.

[14] R.Usbeck, M.Röder, A-C.Ngonga Ngomo, C.Neto, et al. Gerbil–general entity annotator benchmarking framework, 2015.

[15] P.De Vettor, M.Mrissa, and D.Benslimane. Models and adaptive architecture for smart data management. In Sumitra Reddy, WETICE 2015, Cyprus, pp.164–169, 2015.

[16] Y.Cheng and F.Rusu. Parallel in-situ data processing with speculative loading. *Proc. ACM SIGMOD Int.Conf on Management of data*, pp.1287-1298, 2014.

[17] J.Hoffart, M A.Yosef, I.Bordino, et al., Robust disambiguation of named entities in text. In *Proc. of EMNLP'11*, pp.782-792, USA, 2011.

[18] M.Yu, G.Li, D.Deng, and J.Feng. String similarity search and join: a survey. *Frontiers of Computer Science*, pp.399-417, Jun 2016.

[19] G. Wiederhold. Mediators in the architecture of future information systems.pp.38-49, March 1992.

[20] H.Garcia-Molina, Y.Papakonstantinou, D.Quass, et al., The tsimmis approach to mediation: Data models and languages. *J. of intelligent information systems*, pp.117-132, 1997.

[21] G.Zhou and R.Hull. A framework for supporting data integration using the materialized and virtual approaches. In *ACM SIGMOD Int. Conf. on Management of Data*, pp.481-492, 1996.

[22] B.Hüsemann, J.Lechtenbörger, and G.Vossen. *Conceptual data warehouse design*. Univ.Münster.Mathematik und Informatik, 2000.

[23] R.Ghawi and N.Cullot. Database-to-ontology mapping generation for semantic interoperability. In *3rd InterDB*, 2007.

[24] D.Dou, P.LePendu, S.Kim, and P.Qi. Integrating databases into the semantic web through an ontology-based framework. In*22nd Int.Conf ICDEW'06*, pp.54-54. IEEE, 2006.

[25] C.Pautasso, E.Wilde, and R.Alarcon. REST: advanced research topics and practical applications, Springer, 2013.