Dereferencing Service for Navigating Enterprise Knowledge Structures from Diagrammatic Representations

Mihai Cinpoeru(✉)

Business Informatics Research Center, Babeş-Bolyai University,
Cluj-Napoca, Romania
mihai.cinpoeru@econ.ubbcluj.ro

Abstract. This paper aims to offer a straightforward solution to the need of cross-diagram semantic navigation in business diagrams while it also emphasizes the potential and importance of a core Linked Data concept which has yet to gain widespread recognition or implementation, the dereferencing. Business process models that are exported as queryable data in graph databases, using the Resource Description Framework (RDF) for enterprise data storage and retrieval, so that resources have HTTP identifiers which can be interpreted as Web locations (URLs) to gain direct access to a resource instead of querying it. The work at hand implements the URI dereferencing concept through RESTful services. A specific business example will be discussed to demonstrate the new way of accomplishing navigation between diagrammatic enterprise knowledge structures outside the modelling tool, thus making model elements queryable in an external environment.

Keywords: Enterprise resource identity · Linked data · RESTful services · Cross-diagram · URI dereferencing

1 Introduction

Diagrams play an increasingly important role in business applications development, as they are not only used to illustrate structures anymore, but are also able to store extensible and exportable pieces of information and connections to other diagrams, in order to access their items.

At the same time, Smart Data technologies are becoming more and more popular, being able to deliver solutions to problems that have existed for a long time in other types of storage such as relational databases. Considerable attention is being paid to graph databases and the possibility they offer for linking data, as there are constant developments in this field in making data more accessible than ever.

Linked Data follows four important principles proposed by Sir Berners Lee [1]:

1. Use URIs as names for things.
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs, so that they can discover more things.

W. Abramowicz (Ed.): BIS 2017 Workshops, LNBIP 303, pp. 85–96, 2017.
https://doi.org/10.1007/978-3-319-69023-0_9

Whilst the first two principles are used as rules in Linked Data, in this paper the focus is set on the last two, introducing the concept of dereferencing for rule number 3 by dynamically transforming the URI (Uniform Resource Identifier) into a URL (Uniform Resource Locator) through a RESTful Web Service designed to link the URIs to the data they represent when accessed via HTTP.

Furthermore, by respecting the fourth principle, everything needs to be linked in the application developed to enable movement from one resource to another or even from one diagram to another through the hyperlinks elements have, which, in Linked Data, are stored or, more specifically for this situation, exported as statements.

This paper further develops ideas briefly introduced in the author's prior work [2], in the context of the EnterKnow Project [3] (Enterprise-Aware Application Based on Hybrid and Formal Representation of Enterprise Knowledge), which aims to demonstrate the notion of "information system aware of enterprise semantics". This article aims to contribute to this project by enabling enterprise knowledge structures navigation in diagrammatic representations by using Semantic Web specifics.

The next section provides background on the problem described, followed by the goal statement of this paper, to prove that the solution proposed in this article can bring value to the field. Afterwards, there is an example of such a Web Service and a Web Client, that use URI dereferencing to navigate through diagrams, showing how this paper offers an up-to-date, straightforward method to respond to the need mentioned, that of navigating between diagrammatic representations that are exported in RDF and accessed through a dereferencing Web Service developed by the authors and adapted to the specific need of manipulating structures outside the modelling tools. At the same time, there will be details of the architecture and implementation of such a service and then there will be the conclusions to show how the techniques proposed in this paper suit and solve the problem described.

2 Background and Related Works

2.1 Background on Linked Data

Linked Data is a very efficient way for storing and connecting large amounts of data using graphs, benefiting from the unlimited and highly extensible number of connections they can offer and also from the good performance Graph Databases allow.

During the last decade, there has been considerable efforts made in the field of Linked Data, a type of NoSQL that is also referred to as Smart Data or Semantic Web and has been dubbed as Web 3.0 [4].

The model for these data interchange specifications is RDF (Resource Description Framework). In RDF, as mentioned in the Introduction, data is stored as triples of the following form:

```
@prefix : <http://example.org/> .
:John :WorksAt :UBB .
```

Linked Data enables the understanding of data semantics on Web and allows creating new RDF statements through inferences based on the existing ones. RDF Statements do not only give information about a resource, but can also store unlimited relationships between resources, known as metadata.

In a triple, the first resource is the Subject, the second one is the Predicate, and the third one the Object. The prefix sets a symbol that replaces an expression: in this case, when encountering the ":" symbol the machine will read it as http://example.org/ and form an URI such as http://example.org/John. In this example, there are no ontologies used, only terms created in the context presented. The format used is N-Triples, but there are more formats such as TriG or Turtle.

At the same time, there is another format, JSON-LD [5], which is the latest added and the most popular and promising, considering the fact that it is an adaptation of JSON, widely used by Web applications today, which means that client applications developers do not need to be familiar with RDF specific formats.

To ease this up even more, the dereferencing service proposed here can request data by simply calling the resource by its name, sparing the client from having to learn the RDF specific query language, SPARQL, which is included in the service we will describe later in this paper. The client only has to be able to manipulate JSON responses.

2.2 Background on RESTful Web Services

Considering the fact that the URIs from RDF have an HTTP form and that, for the purpose of dereferencing, there is a need for direct calls to URLs, it results that RESTful Web Services are the obvious response to this matter.

REST (Representational State Transfer) is a stateless communication architecture where HTTP protocol is used. RESTful Web Services require every resource to be accessible by HTTP requests such as GET, POST, PUT, DELETE. The RDF based systems include a RESTful API [6], which makes it possible to perform SPARQL queries on the repositories accessed. Though, they do not offer direct support for URI dereferencing. For the latter concept, REST is also the right architectural style, not only because RESTful services are widely used nowadays, but especially because the RESTful architecture relies on HTTP likewise URIs. To make the concept highly scalable, a distributed application is necessary, which involves cross-domain requests. This type of Web Services is also highly suitable for this need.

2.3 Related Works

In the last years, there has been a growing interest in the way that diagrams can contribute to the Web development process. While, years ago, the diagrams had the singular role of illustrating steps or structures, thanks to more recent studies it has come to the point where by designing a diagram the developer can actually develop a well-structured, maintainable model for an application.

Researches conducted by OMILAB, a collaborative research environment where an international community of conceptual modelling researchers contribute with knowledge and resources for modelling method engineering, have added much value to this

field, as it is now possible to export diagrams as RDF, enriching Graph Databases with diagrammatic model information [7]. Another solution, referring to UML diagrams is described in [8], but the downsize of this is that is only refers to UML diagrams, while the previous reference allows any kind of diagrammatic representation to be exported.

For retrieving enterprise knowledge stored diagrammatic representations, prior studies have provided methods in the field of model queries such as GMQL (Generic Model Query Language) [9]. In most of the prior studies though, including the referenced one, the data gathering happens inside the modelling tools, whereas for this study it is necessary to access this type of knowledge outside the tools, at client level.

The start point for this work is BPMN (Business Process Model and Notation), a widely-used standard model for representing processes graphically. In BPMN, Business Process Diagrams (BPD) are based on events which have a Start point and an Ending point, between which there are Tasks and Gateways. The Tasks specify the action to be done, whilst the Gateways represent decision points where, depending on the condition and results, the event is going on one direction or another.

Along with the BPD, there is the Working Environment Model (WEM), which provides specifications for describing the business's organigram. The designer is able to create Organizational Units such as Departments, Task Performers such as Employees or Automated performers and assign them to specific roles, such as Manager or Sales Assistant. The BPM and the WEM can be linked through hyperlinks from the first one to the latter. At task level, it is possible to store various pieces of information, one of which is the task assignment to a Task Performer or a Role. In this project, hyperlinking relies on the Turtle format.

As dereferencing is concerned, to the author's best knowledge, very few publications can be found in the literature that address the issue of URI dereferencing. One paper that discuss it is [10], which though discusses the matter in a specific context which does not suit the purpose this paper aims to achieve.

3 Goal Statement

Based on the approach presented, the purpose of this paper is to establish a connection between the concept of URI dereferencing and navigation between diagrammatic representations used outside the modelling tools, by using HTTP to call the resources by their names through the dereferencing RESTful Web Service, get the data about the concept and show the other URIs that are linked to the concept, making them accessible for dereferencing.

By using diagrammatic representations outside the modelling environment, the diagram's role is enhanced, becoming a run-time level usable piece within an application. The authors aim to add extra value to this concept through the dereferencing Web service, which facilitates access between the models represented in the diagram, allowing easy navigation and complex access between represented structures.

Specifically, this paper aims to dereference a task's URI, then, having the information about that task, to dereference the roles it is linked to, and when the roles are accessed, they offer direct access to the persons responsible for accomplishing the task.

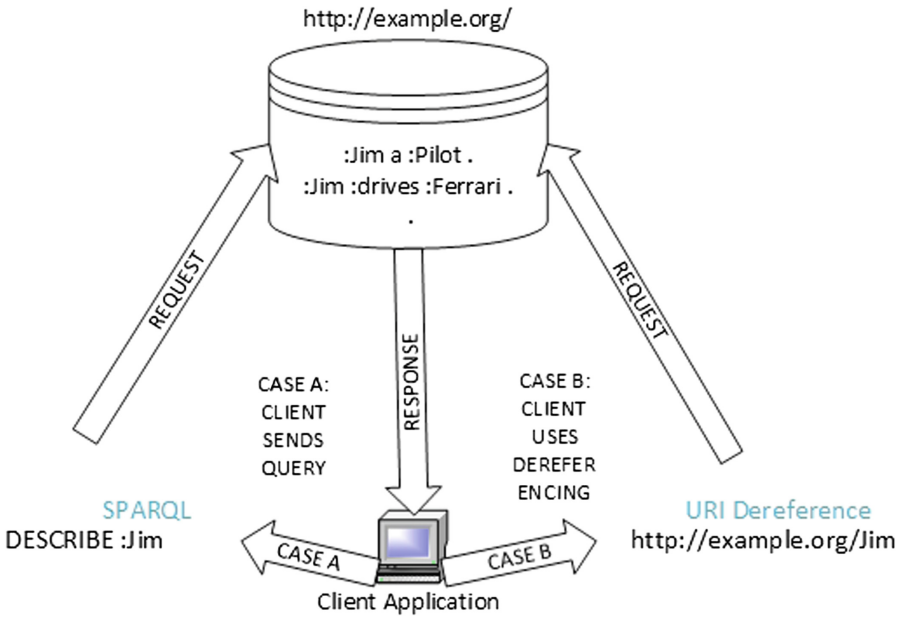

Fig. 1. URI dereference instead of SPARQL query

Figure 1 illustrates a graph database in the middle, which stores three statements in Turtle syntax. Now, in order to retrieve data from the database, the classic choice is to send a query to the RDF API, which would return a response in a format which depends on the header attached.

The option created and proposed within this paper is the URI dereferencing service, for which a second RESTful Web Service will be used. As it is shown in the figure, it is able to retrieve the same data as through the SPARQL query, by calling a resource by its unique identifier, the URI.

This succession of items dereference can show how this concept provides a very useful way to move between items, searching for concrete connections in order to perform the cross-diagram navigation.

The approach of this paper is to use a redesigned RESTful Web Service, which responds to the problem of URI dereference to provide an efficient, straight-forward solution enable navigation1 between diagrammatic representations in the process through Semantic Web and a specifically designed Web Service, contributing to the higher purpose of creating information systems aware of enterprise semantics, through the EnterKnow project.

4 Design Decisions

4.1 Requirements and Architecture

At first, in order to be able to develop a proof-of-concept application for the statement of this paper, it is required to formulate all the requirements and design the architecture.

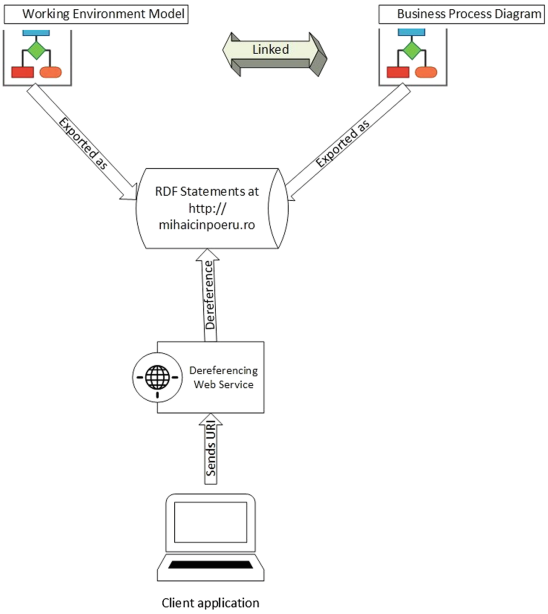

Fig. 2. Application architecture

As shown in Fig. 2, the Working Environment Model and the Business Process Diagram are exported as RDF to a graph database, where they are stored. Furthermore, all the metadata is also stored there, which will be helpful to the matter.

Separately from the storage site, there is the Client Application, which does not need to be aware of the diagrams or the graphs. It simply sends requests to the URI dereferencing Web Service, which is the main contribution enunciated within this paper, a concept developed within the author's recent researches, which requests data via the RDF API, requesting JSON-LD as the format, mainly because it is easy to manipulate even by clients who are not knowledgeable in Smart Data.

The client application, when receiving triples in JSON, can display the data as a table, but it is really important that it respects the fourth principle enunciated by Tim Berners Lee, that links to other URIs must be included in order to access any of the items in the purpose of dereference.

4.2 Business Scenario

In order to exemplify the concepts this paper previously enunciated, it is useful to give an example of a business scenario which could benefit from them.

In this case, to keep focusing on the most important parts, a small scenario would be enough, consisting in a TV store. It only has a Sales Department with one Sales Manager and one Assistant Salesman.

Also, the store only sales TVs, so the two employees are only responsible for selling these. This example could easily be expanded at a larger scale; the authors decided to keep it at this level to avoid the possibility of it becoming confusing.

5 Implementation Details

Diagrams and Graph Database. The separate elements will be implemented considering the order in which others depend on them. This means that at first, the diagrams are required, because the Graph Database depends on them (the diagrams will be exported as graph items).

The diagrams will be implemented through the Bee-Up Modelling Toolkit [11], a modelling toolkit developed at OMiLAB [12] that offers BPMN diagrams and contains the RDF export option, for building the Business Process Diagram, which will represent the succession of steps, and the Working Environment Model, representing the company's organigram.

At first, it is important to have to Working Environment Model, as the BPD will be the one holding references to the WEM. It will cover what was described in the business scenario.

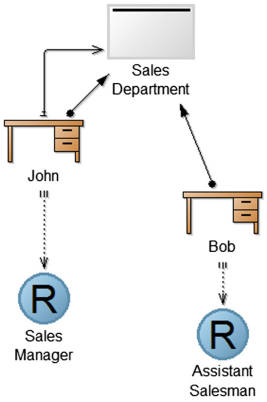

Fig. 3. Working Environment Model diagram

In Fig. 3 there is the small store's WEM, which only consists in the Sales Department. John is the Manager and is also a member of the Sales Department (the left arrow represents the "is Manager of" relationship, whilst the right one represents the "Belongs to" relationship). Bob, on the other hand, is an Assistant Salesman. He only has one relationship with the department, the "Belongs to" one. Further on, it is necessary to create a BPD with tasks and link them to the elements of it.

The RACI matrix [13] describes the participation of various roles in cross-functional or cross-departmental processes.

The acronym of RACI comes from:

- *Responsible.* The person or role that does the actual work in order to complete the task.
- *Accountable.* The one who is responsible for the task's achievement and deliverance. This person also delegates the work, assigning it to the ones responsible.

- *Consulted.* The experts on the matter. When necessary, their feedback or advice is requested. They have the capacity to complete the work.
- *Informed.* It is not necessary to consult them, but they need to be notified on the results.

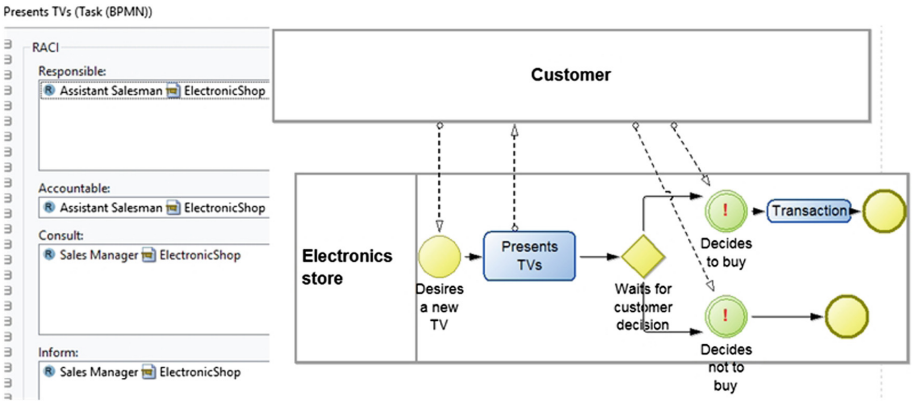

Fig. 4. RACI properties (left) and Business Process Diagram (right)

These four properties need to be attached to every task in the Business Processes Diagram, linking to the WEM. At modelling toolkit level, this enables cross-diagram access, which we aim to accomplish at client application level too.

As illustrated in Fig. 4. RACI Properties (left) and Business Process Diagram (right), this diagram's start event is the customer's buy intention ("Desires a new TV"). It is followed by a task, in which the employee responsible for its achievement presents the TV to the customer, then waits for his decision, which, in BPMN, is a gateway (exclusive, in this case). The actions go forward when the Customer makes a decision, which is communicated through Intermediate Events (Decides to buy/ Decides not to buy). If the customer decides to buy, this is followed by another task, the transaction, then the End Event point ("Deliver TV"). Else, it is followed by the "Customer Leaves Store" End Event.

The RACI properties for each task can be added in a special menu from Bee-Up, as it can be viewed in Fig. 4. RACI Properties (left) and Business Process Diagram (right). Here, is has been decided that while the Assistant Salesman is responsible and accountable for presenting the TV, the Sales Manager is to be consulted or informed (as the roles were explained in Sect. 2.2). The RACI properties were also assigned for the second task. The diagrams, along with the metadata must be exported now as RDF, for which purpose Bee-Up offers support. These are exported in a Turtle file.

In Fig. 5. A task exported in RDF, there is the "Presents TV" task, exported as RDF. This figure presents only the properties relevant to the matter this paper is describing, especially the RACI properties. This data needs to be uploaded to the Graph Database, using the platform offered by GraphDB [14], a Semantic Graph Server.

```
<http://mihaicinpoeru.ro/Task_BPMN-12453-Presents_TVs>
        a                               mm:o_Task_BPMN , cv:o_Modelling_object ;
        rdfs:label                      "Presents TVs" ;
        mm:r_Accountable                <http://mihaicinpoeru.ro/Role-12637-Assistant_Salesman> ;
        mm:r_Consult                    <http://mihaicinpoeru.ro/Role-12626-Sales_Manager> ;
        mm:r_Inform                     <http://mihaicinpoeru.ro/Role-12626-Sales_Manager> ;
        mm:r_Is_inside                  <http://mihaicinpoeru.ro/Pool_BPMN-12439-Electronics_store> ;
        mm:r_Responsible                <http://mihaicinpoeru.ro/Role-12637-Assistant_Salesman> ;
        cv:a_Name                       "Presents TVs" ;
        cv:described_in                 <http://mihaicinpoeru.ro/Working_Environment_Model-ElectronicShop> .
```

Fig. 5. A task exported in RDF

URI Dereferencing and Client. The other thing that needs to be done in order to prove the concept is to create a client application that uses URI dereferencing to get data, manipulates it by JavaScript and displays it in a table to be accessed further by user clicks.

The dereference service is developed through Java with the Spring framework, using Spring Boot to create an application with an embedded servlet container. It needs to be able to receive GET requests and forward them with a hardcoded SPARQL describe query. Also, as a distributed application is desirable, it needs to be able to receive cross-domain requests. For this, we need to enable CORS (Cross-Origin Resource Sharing); this can be done in Spring with the @CrossOrigin annotation.

```
@CrossOrigin
@RestController
public class Controller {
    public Controller() {
    }

    @RequestMapping({"/{uri}"})
    public ResponseEntity requester(@PathVariable String uri) {
        Caller call = new Caller(uri);
        return call.getJson();
    }
```

The service takes what is after http://mihaicinpoeru.ro/ (/{uri}) as a parameter and invokes an object that attaches it to a DESCRIBE query, sent forward to the RDF API with the Content-Type set as "application/ld+json" in the header. The service will return a JSON-LD response, returned to the one that makes the request to the Web Service. The results in JSON-LD returned when asking about the Presents TV task (http://mihaicinpoeru.ro/Task_BPMN-12453-Presents_TVs) look like this:

```
  "http://austria.omilab.org/psm/content/bee-up/1_2#r_Accountable" : [ {
    "@id" : "http://mihaicinpoeru.ro/Role-12637-Assistant_Salesman"
  } ],
//[…] two more similar results here for Consult and Inform Items
  "http://austria.omilab.org/psm/content/bee-up/1_2#r_Responsible" : [ {
    "@id" : "http://mihaicinpoeru.ro/Role-12637-Assistant_Salesman"
```

Table 1. Example of displayed results

Task_BPMN-12453-Presents_TVs	#r_Accountable	Role-12637-Assistant_Salesman
Task_BPMN-12453-Presents_TVs	r_Consult	Role-12626-Sales_Manager
Task_BPMN-12453-Presents_TVs	r_Inform	Role-12626-Sales_Manager
Task_BPMN-12453-Presents_TVs	r_Responsible	Role-12637-Assistant_Salesman

The tasks can be seen at the end of the url sent to the server, after the last ' / ' sign.

The client needs to be able to make cross-domain calls to the dereferencing service, receive JSON responses and manipulate them as a HTML table with clickable links in order to be able to dereference each item from it that belongs to the http://mihaicinpoeru.ro prefix (it only dereference resources created within the context – all the items created in the Bee-up modelling toolkit were exported with this prefix). The client application will practically establish a connection with the URI dereference service and will have a stateless communication with it; the client is responsible for displaying the properties of each item that is dynamically dereferenced. For this paper, the script will not sort out any properties, but in case of a more complex application, this would happen in the background, manipulating the items that are necessary.

In Table 1 there are just four of the statements displayed by the client application when accessing the item from the first column (the "Presents TV" task), the ones that were also shown in the JSON-LD example. All the prefixes have been filtered out, as they are added with JavaScript in the background. A click on the URI of the role item will retrieve and provide to the client an RDF description of the role responsible for the task. The dereference service will create an URL out of it and bring back data to the application in the same way as for the first task.

```
Performer-12634-Bob                    @type      o_Performer
```

This is one of the statements retrieved when clicking on the Role-12637-Assistant_Salesman item. Following the former steps, an available Assistant Salesmen for the job requested was discovered. This statement was chosen because it is most important in this context; doing this, this paper's statement has been accomplished, showing how the URI dereference service contributed to accomplishing the cross-diagram navigation: starting from a task from the BPD diagram, the WEM diagram was accessed, retrieved data needed for the task from it.

6 Conclusions

Based on these results, it can be concluded that the method described has accomplished its purpose, the navigation between enterprise knowledge structures in diagrammatic representations at application level, outside the modelling tools. It has been exemplified on a toy example reduced to a minimum level in order to facilitate understanding, but this principle can be generalized to the numerous enterprise facets that can be modelled and connected through semantic links. Amongst these facets can document models, actors, products and many others. By dereferencing the URIs it is now possible to reach all the elements' machine-readable properties, and when some of these properties have hyperlinks to other diagrams, the latter are reachable through those links in any web application using the URI dereference.

This technique can be easily used by enterprise applications that use Graph Databases as their Model, offering easier access to the resources. As shown here, an organigram can be reached from a business process description, a fact that is useful to business applications.

Table 2 presents some performance results of the client application through the dereferencing service, taking into consideration the time it takes for the statements on an item to be shown to the client.

Table 2. Performance table

Item dereferenced	Time needed to dereference and display an item in client application	Number of statements
Task	19 ms	48
Role	16 ms	12
Performer	18 ms	15

To conclude, the main contribution of this study is proving how a URI dereferencing service can be used in the purpose of accomplishing cross-diagram navigation in applications based on Smart Data.

Acknowledgment. The work presented in this paper is supported by the Romanian National Research Authority through UEFISCDI, under grant agreement PN-III-P2-2.1-PED-2016-1140.

References

1. Berners-Lee, T.: Design Issues. W3 Homepage. https://www.w3.org/DesignIssues/LinkedData.html. Accessed 01 May 2017
2. Cinpoeru, M.: Design and implementation of a dereferencing service for enterprise resource identifiers. In: IE 2017 Conference Proceedings, pp. 501–506. Bucharest University of Economic Studies Press, Bucharest (2017)
3. EnterKnow Homepage. http://enterknow.granturi.ubbcluj.ro. Accessed 09 May 2017

4. Markoff, J.: Entrepreneurs see a web guided by common sense. New York Times. http://www.nytimes.com/2006/11/12/business/12web.html. Accessed 09 May 2017
5. JSON LD – the official website. http://json-ld.org/. Accessed 11 May 2017
6. RDF RESTful API Documentation – the official website. http://docs.rdf4j.org/rest-api/. Accessed 11 May 2017
7. Karagiannis, D., Buchmann, R.A.: Linked open models: extending linked open data with conceptual model information. Inf. Syst. **56**, 174–196 (2015)
8. Daniel, G., Sunyé, G., Cabot, J.: UMLtoGraphDB: mapping conceptual schemas to graph databases. In: Comyn-Wattiau, I., Tanaka, K., Song, I.-Y., Yamamoto, S., Saeki, M. (eds.) ER 2016. LNCS, vol. 9974, pp. 430–444. Springer, Cham (2016). doi:10.1007/978-3-319-46397-1_33
9. Delfmann, P., Steinhorst, M., Dietrich, H.-A., Becker, J.: The generic model query language GMQL – conceptual specification, implementation, and runtime evaluation. Inf. Syst. **47**, 129–177 (2015)
10. Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R.: Painless URI dereferencing using the DataTank. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8798, pp. 304–309. Springer, Cham (2014). doi:10.1007/978-3-319-11955-7_39
11. OMILAB Bee-up – the official website. http://austria.omilab.org/psm/content/bee-up/info. OMILAB, Accessed 09 May 2017
12. Karagiannis, D., Buchmann, R.A., Burzynski, P., Reimer, U., Walch, M.: Fundamental conceptual modeling languages in OMiLAB. In: Karagiannis, D., Mayr, H., Mylopoulos, J. (eds.) Domain-Specific Conceptual Modeling, pp. 3–30. Springer, Cham (2016). doi:10.1007/978-3-319-39417-6_1
13. Jacka, M., Keller, P.: Business Process Mapping: Improving Customer Satisfaction, pp. 257–260. Wiley, Hoboken (2009)
14. GraphDB – the official website. http://graphdb.ontotext.com/. Ontotext, Accessed 09 May 2017