

# An Approach for Automatic Bidirectional Mapping Between Data Models and RDF-S

Aissam Belghiat<sup>a</sup>

LaRIA Laboratory, University of Jijel, 18000 Jijel, Algeria

Keywords: RDF, RDF-S, UML, MDA, Transformation.

Abstract: RDF and RDF-S are the normative language for describing web resource information in the context of the Semantic Web. Constructing RDF-S from scratch is a painful task, and deriving them from existing data sources became an important research problem. Furthermore, updating and evolving established RDF-S documents is another problem which must be taken into account. UML is widely applied to data modeling in many application domains. Building RDF-S from existing UML models is a promising technique that will facilitate elaboration of RDF-S models. Moreover, mapping RDF-S to UML will allow their intuitive updating. Thus, this work proposes an approach for mapping UML to RDF-S and RDF-S to UML. The translation makes the data modeled in UML class diagrams available for the Semantic Web and vice versa. The aim is facilitating building and evolving RDF-S documents using UML and vice versa.

## 1 INTRODUCTION

In the last decade, the Semantic Web has made giant steps to become a reality. It is an extension of the current World Wide Web by making the information available in it treatable by the machine. The aim is to constitute an environment in which data could be shared and reused across application, enterprise, and community boundaries. The W3C (World Wide Web Consortium) (W3C, 2004), which is the organization in charge of evolution of Semantic Web, has proposed many recommendations to promote common data formats and exchange protocols on the Web, fundamentally the RDF (Resource Description Framework) (W3C, 2014a) and the RDF-S (RDF Schema) (W3C, 2014b). In fact, RDF and RDF-S are the normative language recommended by the W3C to describe the Web resource information and their semantics.

RDF-S is a key language in Semantic Web. It is used to annotate web resources in order to enable their access by machines. In spite of its adoption and its growing acceptance, RDF-S remains a new technology that most information resources still did not make their mapping to this new technology for different reasons. Actually, building RDF-S from scratch is a quite hard task. Therefore, how to build


RDF-S from existing information resources is still a research problem. The conceptual diagrams for a specific domain, developed in the context of software development, constitute an important and useful source of information for RDF-S documents.

UML (Unified Modeling Language) (OMG, 2017) should help in this direction. It is a standardized language for visualizing, specifying, building and documenting all the aspects and artifacts of a software system. It could give effective solutions by allowing already existing resources modeled in this language to be reachables by transforming their data models represented in Class Diagrams to RDF-S. The MDA (Model Driven Architecture) (OMG, 2014) states how to do such transformation.

We intentionally do such mapping due to the observed convergence between the two formalisms, as already discussed in sources such as (Chang, 1998). In fact, UML and RDF-S comprise some components which are similar in several aspects, such as: classes, associations, properties, packages, types, and instances although their different aims.

This paper aims at providing an approach to simplify the development of RDF-S files by transforming UML models to RDF-S and vice versa.

Mapping UML to RDF-S allows avoiding building from scratch and takes advantage of the

<sup>a</sup> <https://orcid.org/0000-0002-5968-609X>

flexibility and clarity of UML models to simplify the task. The inverse operation, i.e. from RDF-S to UML, allows getting updating and evolving existing RDF-S models easily by making the changes on graphical and intuitive UML models and return back to RDF-S.

The rest of the paper is organized as follows. In Section 2, related works are presented. In Section 3, some preliminaries are exposed. In Section 4, the proposed approach is explained. Section 5 presents an example to understand the approach. Section 6 finishes the paper and gives some perspectives.

## 2 RELATED WORK

Many approaches have been proposed to build RDF-S from data sources.

Since the relational model for databases is employed widely for storing, treating and retrieving data in almost all domains, there were multiple attempts to make a bridge to RDF-S documents. In (Korotkiy and Top, 2004) the authors have proposed an approach for converting and integrating of relational-style information resources into RDF-S-aware systems. In (Krishna, 2006), The author made a semantic retaining mapping of relational databases to RDF. In (Sequeda et al., 2012), relational databases with their integrity constraints are directly mapped to RDF and OWL with information and query preservation. In (Mallede, 2013), the authors have presented algorithms to map entirety relational databases by adopting a methodology that not only map the data but also the domain specific knowledge. In (Michel, 2013), a detailed review of seventeen RDB-to-RDF translations has done, considering the projects that developed operational tools. Other less spread databases, such as object-oriented databases, are also used to automatically construct RDF-S as in (Shan et al., 2023), which presented formal mapping rules and a tool named OODB2RDF to validate the method. Also, (Tong, 2018) did the same work by carrying more thorough correspondences.

Other approaches tried to take advantage of the widespread adopting of XML in representing web documents and transform them to RDF-S in order to allow their semantic annotation. In (Klein, 2002), a procedure for translating XML documents into RDF statements via an RDF-S specification is presented to permit semantic annotation of XML documents via external RDF schema ontologies. In (Thuy et al., 2007), the authors present another procedure for transforming valid XML documents into RDF by using RDF schema vocabularies, the integrity of the structure and meaning of the original XML

documents while transforming are ensured. In (Kumar and Babu, 2013), the authors give a study of RDF characteristic, then they proposed an approach for constructing RDF models from well formatted valid XML document. In (Riaz et al., 2019), a method is proposed for generating RDF metadata from legacy software models by transforming UML models into RDF triples through XML parsing.

Other approaches could be recalled such as in (Amato et al., 2008), where the authors made an approach to build RDF from semi-structured legal documents. Also, in (Han et al., 2008) where the authors presented the rules of constructing RDF from spreadsheets.

Regarding establish correspondences between the UML and the Semantic Web, multiple studies have been done trying to make the bridge between them as in (Cranefield, 2001) (Baclawski, 2001) (Guizzardi, 2004), while the mapping between UML and RDF-S was done in (Chang, 1998), (Cranefield, 2000), (Kim, 2005), (Belghiat and Bourahla, 2012), (Tong et al., 2014), and (Lieber et al., 2022).

The main contribution of our approach in comparison with the previous presented works is that it enables the automatic mapping from UML to RDF-S and also the inverse translation, i.e. from RDF-S to UML.

## 3 PRELIMINARIES

### 3.1 UML Class Diagram

The class diagram models the internal structure of a system. It allows showing the entities of a system and their relations. A Class diagram is essentially composed of classes and relations. A class is used to represent the entities of a system. It uses attributes to represent data and operations to represent treatments. A relation is used to represent a link between two classes. It can be: an association, a dependence, an inheritance, or a class association.

Nowadays, Class diagrams are becoming quite important and unavoidable in data modeling in many application domains. Actually, they are used in an application domain to describe the static structure of its information.

### 3.2 RDF-S

RDF-S (Resource Description Framework Schema) (W3C, 2014b) is an extension of the RDF vocabulary (W3C, 2014a) that provides a data-modeling vocabulary for RDF data. Both formalisms are

normative languages and they are used to describe the web resource information. RDF-S is composed essentially of classes and properties intended to structure RDF resources which are saved in a triple store to reach them later with the query language SPARQL (W3C, 2013). A class denoted “rdfs:Class” is used to represent groups of resources, and a member of a class is known as an instance of the class. The classes are themselves resources, and they are described using RDF properties. There is a difference between a class and the set of its instances, named the class extension. In fact, two different classes may have the same set of instances. Properties are instances of the class “rdf:Property”, they describe a relation between subject resources denoted by “rdfs:domain” and object resources denoted by “rdfs:range”.

### 3.3 MDA and Tree Transformation

MDA consists of using models in all phases of development and proceeding to their refinements and enrichments by successive transformations. This latter can be defined by a set of rules that allows passing from a high abstract model (meta-model) to another, by defining for each source elements their equivalents among the target elements. Tree transformation is a very common used technique to realize Model transformations.

Tree transformation is usually based on XML documents, where the information contained in an XML type document is structured in a tree-like manner. Thus, the XML transformation, in other words the transition from one XML type document to another, is actually an operation that transforms a tree into another tree structure. It consists of using XSLT (W3C, 2017a) or XQuery (W3C, 2017b) to do the transformation. The original document is called the source document, while the document resulting from the transformation is called the target document as indicated in Fig. 1.



Figure 1: XML tree transformation.

#### 3.3.1 XMI

XMI (XML Metadata Interchange) (OMG, 2015) is an OMG standard for model representation created to

define a way to represent a model as an XML document in order to exchange these models between modeling tools. Since most models are graphical and abstract entities, OMG has decided to standardize XMI to provide a concrete representation for models.

#### 3.3.2 XSLT

XSLT (eXtensible Stylesheet Language for Transformations) (W3C, 2017a) is a language that allows the transformation of XML documents. The transformation consists of either a modification of an XML and save the changes, or a transformation into another type of documents (e.g. HTML and PDF).

In our case, we use XSLT for transforming a XMI file that represents the source model, to another XML file that represents the target model. The source model is the Class diagram while the target model is the RDF-S document. The Papyrus (Lanusse et al., 2009) editor is used to edit UML Class diagrams and derives their corresponding XMI file. Papyrus uses a light format of XMI named “.uml”.

## 4 THE APPROACH

In our approach, we build rules for the transformation of a class diagram to an RDF-S model and vice versa. For the implementation of these rules we develop a XSL style sheet that translates a serialized class diagram into an RDF-S model described in XML.

This transformation is done by an automatic generation of the RDF-S file represented in RDF/XML format from an UML class diagram edited in Papyrus. This solution is implemented in Eclipse (Eclipse, 2004), it takes place in several steps (see Fig.2):

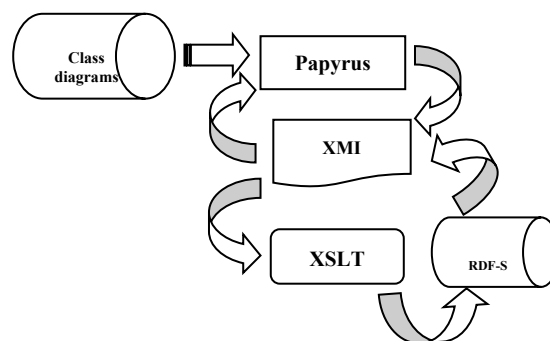


Figure 2: Architecture of the mapping UML, RDF-S.

- Graphical description of class diagrams in Papyrus using a file “.umldi”.

- Saving class diagram: it is automatically saved in two files “.umldi” and “.uml”.
- Using an XSLT processor, we apply an XSL style sheet (UMLtoRDFS.xsl) on the file “.uml” to transform it into an RDF-S file represented in RDF/XML.

### 4.1 The Transformation Rules

UML and RDF-S both are modeling languages where UML is used for object modeling and the RDF-S is used for knowledge representation. By quick observation of these two languages one can note the existence of a certain number of common notions between Class diagram and RDF-S: classes, relations, properties, etc. Nevertheless, there are several significant differences between these concepts.

We present here in Table 1, in a general way, the correspondences between UML class diagrams and RDF-S models.

Table 1: Class diagrams and RDF-S correspondances.

Class Diagram	RDF-S
Package	Resource
Class	Class
Class, Abstract class, Interface	Class
Inheritance	Sub-class
Attribute	Property
Dependency, Realization	Property
Association,	Property
Association, Aggregation, Composition	Property
Role	Sub- property

### 4.2 The Proposed XSL Style Sheet

To succeed transforming each class diagram edited in the UML editor of Papyrus to RDF-S, we propose an XSL style sheet composed of a number of templates. It should be noted that a template can contain several transformations and we present only templates related immediately to the transformation rules (we do not present the templates that help to realize the transformation rules). We take extracts from them:

#### Template 1: Package Transformation

- **Name:** get\_package
- **Role:** This Template (see Fig.3) allows you to transform a package into an <rdfs: resource> element. The name of the resource is the name

of the package, and the template takes that name from the name of the diagram package.

```
<!-- @@@@@@@@@@@@@@@@@@ TRAITEMENT DES PACKAGE UML @@@@@@@@@@@@@@@@@@ -->
<!-- @@@@@@@@@@@@@@@@@@ -->
<xsl:template name="get_package">
<xsl:element name="rdfs:Resource">
  <xsl:attribute name="rdf:about">#
    <xsl:value-of select="//packageElement[@xmi:type = 'uml:Package']/@name"/>
  </xsl:attribute>
</xsl:element>
</xsl:template>
<!-- @@@@@@@@@@@@@@@@@@ FIN TEMPLATE @@@@@@@@@@@@@@@@@@ -->
```

Figure 3: Template for the package transformation.

#### Template 2 : Classes, Interfaces and Generalizations Transformation

- **Name:** get\_class
- **Role:** This Template (see Fig.4) is used to transform a class from the class diagram to an RDF-S class. This class takes its name from the class of the source diagram. In this Template, we check if this class is: a specialization of another class, connected by an association, connected by a class-association, or realizes an interface. If so, we proceed to treatments according to the transformation rules. The excerpt below shows the transformation of the class, interface and inheritance. This transformation is carried out by browsing the path of the source tree and generating the appropriate RDF-S code.

```
<!-- @@@@@@@@@@@@@@@@@@ TRAITEMENT DES CLASSES UML @@@@@@@@@@@@@@@@@@ -->
<!-- @@@@@@@@@@@@@@@@@@ -->
<xsl:template name="get_class">
<xsl:for-each select="//packageElement">
  <xsl:if test="@xmi:type='uml:Class' or @xmi:type='uml:Interface'">
    <!-- nom et id de la classe -->
    <xsl:variable name="nom_classe" select="@name" />
    <xsl:variable name="id_xmi" select="@id" />
    <!-- creer un nouvel element rdfe:Class et attribut rdf:ID -->
    <xsl:element name="rdfe:Class">
      <xsl:attribute name="rdf:ID">
        <xsl:if test="@xmi:type='uml:Interface'">Interface</xsl:if>
        <xsl:if test="@isabstract='true'">Abstract</xsl:if><xsl:value-of select="@name"/>
      </xsl:attribute>
    </xsl:element>
  </xsl:for-each>
  <!-- Traitement des generalisations -->
  <xsl:for-each select="generalization">
    <xsl:variable name="id_classe_gen" select="@general" />
    <!-- appel template avec le parametre id generalisation -->
    <xsl:call-template name="avoir_pere">
      <xsl:with-param name="id_classe"><xsl:value-of select="@id_classe_gen"/></xsl:with-param>
    </xsl:call-template>
  </xsl:for-each>
```

Figure 4: Template for class treatment.

#### Template 3: Transformation of Attributs

- **Name:** get\_attribute
- **Role:** This Template (see Fig.5) is used to transform an attribute of a class in the class diagram to an RDF-S property. This property takes the name of the class concatenated with the name of the attribute. The domain and the image are respectively defined from the class to the type of the attribute. The code excerpt below shows the transformation of the attribute



## 5 CASE STUDY

To demonstrate how our proposed approach is used for automatic bidirectional mapping between UML Class Diagrams and RDF-S models, we provide a case study about transforming a Course Management System of a university. The latter illustrates the effectiveness of our approach.

An IT department of a university needs to transform their existing UML-based course management system into an RDF-S model to integrate with the Semantic Web for better data sharing and reuse. Additionally, any updates made in the RDF-S model should be easily incorporated in the source UML diagrams.

### 5.1 UML Class Diagram

The course management system of the university is already modeled using UML Class Diagrams, as we suppose that the conceptual design of this system is made by UML. Fig.10 shows a simplified version of the class diagram. It mainly includes the following elements:

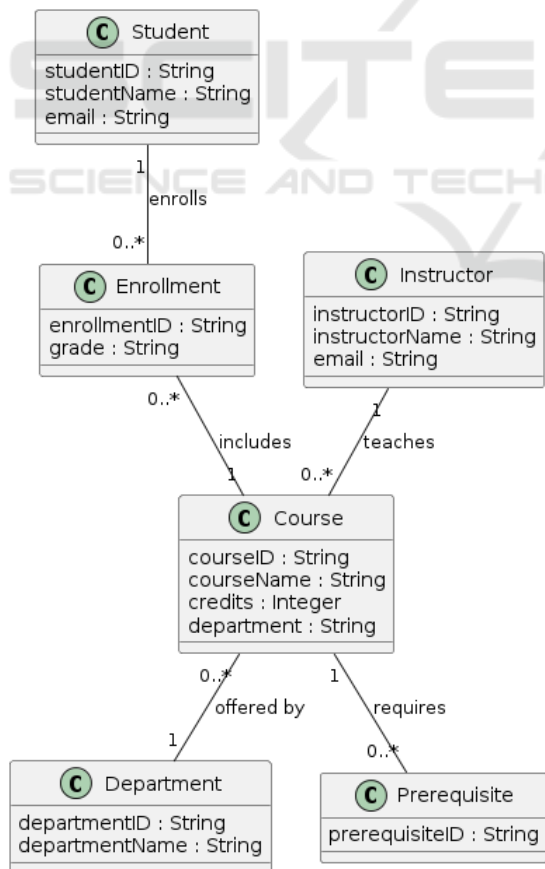


Figure 10: UML class diagram of the system.

- **Course:** Represents a course provided by the university. Attributes include courseID, courseName, credits, and department.
- **Student:** Represents a student enrolled at the university. Attributes include studentID, studentName, and email.
- **Enrollment:** Represents the enrollment of a student in a course. Attributes include enrollmentID and grade.
- **Instructor:** Represents an instructor teaching a course. Attributes include instructorID, instructorName, and email.
- **Department:** Represents a department within the university. Attributes include departmentID and departmentName.
- **Prerequisite:** Represents a prerequisite relationship between courses. Attributes include prerequisiteID.

### 5.2 Transformation to RDF-S

Using our approach, the UML class diagram is transformed into an RDF-S model. The transformation rules defined in XSLT stylesheet are applied to the UML class diagram saved as an XMI file. The tool EasyRdf (EasyRdf, 2012). is used to validate the derived RDF-S file. The full resulting RDF-S model (in RDF/XML format) is available at the open repository DOI: 10.5281/zenodo.13850481.

- 1. Course Class Transformation:**
  - o UML Class: Course
  - o RDF-S Class: <rdfs:Class rdf:ID="Course">
  - o Attributes: Transformed into RDF properties.
- 2. Student Class Transformation:**
  - o UML Class: Student
  - o RDF-S Class: <rdfs:Class rdf:ID="Student">
  - o Attributes: Transformed into RDF properties.
- 3. Enrollment Class Transformation:**
  - o UML Class: Enrollment
  - o RDF-S Class: <rdfs:Class rdf:ID="Enrollment">
  - o Attributes and relationships: Transformed into RDF properties.
- 4. Instructor Class Transformation:**
  - o UML Class: Instructor
  - o RDF-S Class: <rdfs:Class rdf:ID="Instructor">
  - o Attributes: Transformed into RDF properties.
- 5. Department Class Transformation:**
  - o UML Class: Department
  - o RDF-S Class: <rdfs:Class rdf:ID="Department">
  - o Attributes: Transformed into RDF properties.
- 6. Prerequisite Class Transformation:**
  - o UML Class: Prerequisite
  - o RDF-S Class: <rdfs:Class rdf:ID="Prerequisite">

- Attributes: Transformed into RDF properties.

Recently, the university has become a member of a consortium of institutions that share course information using Semantic Web technologies. Consequently, the course management system needs to maintain both UML class diagrams for internal development and RDF-S models for interoperability with external systems.

The head of the consortium curriculum committee is responsible for updating the course information to include the most recent curriculum changes. They collaborate with the IT departments to guarantee these updates are accurately represented in the system.

They have the need to add a new attribute *semester* to the "Course" class to indicate the semester in which a course is delivered. This change is essential for curriculum planning and student scheduling. Thus, they update the RDF-S model with the new *semester* attribute using a web-based ontology editor, which the curriculum committee utilizes to manage the shared course data.

Using our approach, the IT department can guarantee that the UML class diagram, used for internal development, received this new attribute which preserve consistency between the RDF-S model and the UML diagrams.

The new updated RDF-S is changed by adding the property "semester" as follows:

#### 1. Course Class Transformation:

- UML Class: Course
- RDF-S Class: `<rdfs:Class rdf:ID="Course">`
- Attributes: `<rdf:Property rdf:ID="courseID">`  
`<rdf:Property rdf:ID="courseName">`  
`<rdf:Property rdf:ID="credits">`  
`<rdf:Property rdf:ID="semester">`

The updating RDF-S model (in RDF/XML format) is available at the open repository DOI: 10.5281/zenodo.13850481.

To reflect this change in the UML class diagram using our approach, the RDF-S model is parsed and the corresponding UML class diagram in XMI format is updated to include the "semester" attribute. The detailed of the XMI file is available at the open repository DOI: 10.5281/zenodo.13850481.

This scenario illustrates the constant need for bidirectional transformation between UML and RDF-S. When domain experts like the head of consortium curriculum committee make updates to the RDF-S model, these changes must be reflected back in the UML class diagrams to ensure consistency across the system. This bidirectional capability guarantees that both the internal design documents and the shared

semantic models remain aligned, facilitating accurate system development, maintenance, and interoperability.

## 6 CONCLUSION

We have presented in this paper an approach for the automatic bidirectional mapping between UML Class Diagrams and RDF-S models. The construction of RDF-S documents is based on the annotation of business models expressed in UML class diagrams, and the inverse transformation is devoted to enable updating existing RDF-S file to allow their evolution. To do so, we have used a style sheet for transforming UML models, expressed in XMI, to RDF-S documents represented in RDF/XML and vice versa. Eclipse's Papyrus UML editor is used to describe UML class diagrams and generates their XMI files, and the XSLT processor is used to process the written style sheet in order to perform the transformation.

In future work, we plan to add other advanced elements of UML Class diagrams to the framework. This will provide rich models that meet the needs of users. We intend also to add other diagrams that could offer other information to the tool.

## REFERENCES

- Amato, F., Mazzeo, A., Penta, A., & Picariello, A. (2008). Building RDF Ontologies from Semi-Structured Legal Documents. In Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems (pp. 997-1002).
- Baclawski, K., Kolar, M., Kogut, P., et al. (2001). Extending UML to support ontology engineering for the Semantic Web. In Proceedings of the Fourth International Conference on UML (pp. 342-360).
- Belghiat, A., & Bourahla, M. (2012, March). Transformation of UML models towards OWL ontologies. In 2012 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT) (pp. 840-846). IEEE.
- Belghiat, A., & Bourahla, M. (2012). An approach based AToM3 for the generation of OWL ontologies from UML diagrams. International journal of computer applications, 41(3).
- Belghiat, A., & Bourahla, M. (2012). From UML Class Diagrams to OWL Ontologies: A Graph Transformation Based Approach. In ICWIT (pp. 330-335).
- Chang, W. W. (1998). A Discussion of the Relationship Between RDF-Schema and UML. Retrieved from <http://www.w3.org/TR/NOTE-rdf-uml/>
- Cranefield, S., Haustein, S., & Purvis, M. (2000). UML Based Ontology Modeling for Software Agents. In

- Proceedings of the Ontologies in Agent Systems Workshop (pp. 21-28).
- Cranefield, S. (2001). UML and the Semantic Web. In Proceedings of the first Semantic Web Working Symposium (pp. 113-130).
- EasyRdf. (2012). <http://www.easyrdf.org/>
- Eclipse Foundation. (2004). <https://www.eclipse.org/>
- Guizzardi, G., Wagner, G., & Herre, H. (2004). On the Foundations of UML as an Ontology Representation Language. In Proceedings of the 14th International Conference Engineering Knowledge in the Age of the Semantic Web (pp. 47-62).
- Han, L., Finin, T. W., Parr, C. S., Sachs, J., & Joshi, A. (2008). RDF123: From spreadsheets to RDF. In Proceedings of the 7th International Semantic Web Conference (pp. 451-466).
- Kim, J. S., Yoo, C. S., Lee, M. K., & Kim, Y. S. (2005). Object Modeling of RDF Schema for Converting UML Class Diagram. In Proceedings of the International Conference on Computational Science and Its Applications (pp. 31-41).
- Klein, M. C. A. (2002). Interpreting XML Documents via an RDF Schema Ontology. In Proceedings of the 13th Database and Expert Systems Applications (pp. 889-894).
- Korotkiy, M., & Top, J. L. (2004). From Relational Data to RDFS Models. In Proceedings of the 4th International Conference on Web Engineering (pp. 430-434).
- Krishna, M. (2006). Retaining Semantics in Relational Databases by mapping them to RDF. In Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (pp. 303-306).
- Kumar, B. H., & Babu, M. S. P. (2013). Study and Constructing RDF model for a well formatted Valid XML document. International Journal on Computer Science and Engineering, 5(7), 648-652.
- Lanusse, A., Tanguy, Y., Espinoza, H., Mraidha, C., Gerard, S., Tessier, P., ... & Terrier, F. (2009). Papyrus UML: an open source toolset for MDA. In Proc. of the Fifth European Conference on Model-Driven Architecture Foundations and Applications (ECMDA-FA 2009).
- Lieber, S., De Meester, B., Heyvaert, P., Brückmann, F., Wambacq, R., Mannens, E., ... & Dimou, A. (2022). Visual notations for viewing RDF constraints with UnSHACLed. Semantic Web, 13(5), 757-792.
- Mallede, W. Y., Marir, F., & Vassilev, V. T. (2013). Algorithms for mapping RDB Schema to RDF for Facilitating Access to Deep Web. In Proceedings of the First International Conference on Building and Exploring Web Based Environments (pp. 32-41).
- Michel, F., Montagnat, J., & Faron Zucker, C. (2013). A survey of RDB to RDF translation approaches and tools. Report, 1-24.
- OMG. (2014). Model Driven Architecture (MDA). <https://www.omg.org/mda/>
- OMG. (2015). XML Metadata Interchange (XMI). <https://www.omg.org/spec/XMI/About-XMI/>
- OMG. (2017). Unified Modeling Language (UML), Superstructure,v2.5. <https://www.omg.org/spec/UML/About-UML/>
- Poveda-Villalón, M., Chávez-Feria, S., Carulli-Pérez, S., & García-Castro, R. (2023). Towards a UML-based notation for OWL ontologies. In VOILA@ ISWC (pp. 18-27).
- Riaz, A., Bajwa, I. S., & Ali, M. (2020). Automatic RDF, Metadata Generation from Legacy Software Models. In Intelligent Technologies and Applications: Second International Conference, INTAP 2019, Bahawalpur, Pakistan, November 6–8, 2019, Revised Selected Papers 2 (pp. 385-397). Springer Singapore.
- Sequeda, J. F., Arenas, M., & Miranker, D. P. (2012). On directly mapping relational databases to RDF and OWL. In Proceedings of the 21st World Wide Web Conference (pp. 649-658).
- Shan, J., Lu, J., Chen, X., Yan, L., & Ma, Z. (2023). A Semantics-preserving Approach for Extracting RDF Knowledge from Object-oriented Databases. Journal of Web Engineering, 22(2), 197-220.
- Thuy, P. T. T., Lee, Y. K., Lee, S., & Jeong, B. S. (2007). Transforming Valid XML Documents into RDF via RDF Schema. In Proceedings of the International Conference on Next Generation Web Services Practices (pp. 35-40).
- Tong, Q., Zhang, F., & Cheng, J. (2014). Construction of RDF(S) from UML class diagrams. Journal of Computing and Information Technology, 22(4), 237-250.
- Tong, Q. (2018). Mapping object-oriented database models into RDF (S). IEEE Access, 6, 47125-47130.
- World Wide Web Consortium.(2004) <https://www.w3.org/>
- World Wide Web Consortium. (2013). SPARQL 1.1 Overview.<https://www.w3.org/TR/sparql11-overview/>
- World Wide Web Consortium. (2014a). RDF 1.1 concepts and abstract syntax. <https://www.w3.org/TR/rdf/>
- World Wide Web Consortium. (2014b). RDF Schema 1.1. <https://www.w3.org/TR/rdf-schema/>
- World Wide Web Consortium. (2017a). XSL Transformations (XSLT) Version 3.0. <https://www.w3.org/TR/2017/REC-xslt-30-20170608/>
- World Wide Web Consortium. (2017b). XQuery 3.1: An XML Query Language. <https://www.w3.org/TR/2017/REC-xquery-31-20170321/>.