SemantiConverter: A Flexible Framework to Convert Semi-Structured Data into RDF

Julien Tscherrig[1], Philippe Cudré-Mauroux[2], Elena Mugellini[1], Omar Abou Khaled[1], and Maria Sokhn[1]

[1] HES-SO, University of Applied Sciences of Western Switzerland, Fribourg, Switzerland, Computer Science `julien.tscherrig@hes-so.ch`, `elena.mugellini@hes-so.ch`, `omar.aboukhaled@hes-so.ch` and `maria.sokhn@hes-so.ch`

[2] UNIFR, University of Fribourg, Switzerland, Computer Science, `pcm@unifr.ch`

Abstract. Online data can be presented in various forms, and often contains semi-structured data that are serialized in XML or JSON. Such serialized data is either subsequently stored or used as an exchange format between two or more services. Since the burgeoning of Semantic Web technologies, RDF became more popular, mainly due to its flexibility. Unfortunately, semantic technologies work almost solely with RDF and it is thus often necessary to convert other formats into RDF prior to any semantic processing. Currently, only a handful of solutions allow to automatically convert semi-structured data into RDF. The Framework presented on this paper offers the possibility to automatically realize a conversion of either XML or JSON data into a valid RDF document. Our Framework analyzes the structure of a given document and creates generic conversion rules out of it. Subsequently, the user can customize and adapt the generated rules and the structure of the output document through a graphical user interface.

1 Introduction

The current formats used with Semantic Web technologies and ontology representation are RDF and OWL mainly. RDF and its derived formats like RDFS and OWL is a fairly new family of formats. These new formats offer a semantic layer that proved the possibility to link pieces of information with one another. However, a large quantity of online data is still structured using XML or JSON on the Internet. Semantic technologies do not work well with such formats that represent data in a hierarchical, semi-structured manner. As a result, a large fraction of the online data are inaccessible to Semantic Web technologies. Faced with this issues, developers routinely code custom scripts to convert specific XML or JSON documents into RDF. Hand-coding such scripts take a lot of time and manual effort for each new type of document that needs to be converted.
Based on those observations, we propose in this paper a solution to realize an automatic conversion of XML or JSON documents into RDFS based on rules created on the fly. In second time, an expert user can edit the conversion rules

to customize the RDFS output and then create his own output structure.
A number of tools for working with Semantic Web technologies already exist but are not always wide spread. RDF documents are still uncommon in several communities and are under-represented on the Internet in comparison of XML or JSON. The idea of this framework is to provide a graphical interface to the user in order to customize the translation based on several semantic transformation rules in order to personalize the RDF output. Since RDF documents can be very different, the graphical editor gives the user a visual way to edit the rules for automatic conversions. Our framework then incorporates the structure of these formats and prepares conversion rules that can then be used immediately and automatically for additional conversion of XML or JSON data into RDF.
We start this paper by reviewing the related work and the similar software frameworks. We then describe our framework in Section 3. Several use cases showing the utility of such a project are then discussed in Section 4. Finally, we discuss the evaluation of our solution in Section 5 before concluding.

2 Related Work

Semantic information plays basic roles in the data processing or exchange. Currently, when you have semi-structured data only a few solution exist. Take the XML format case. The most used transformation from XML to RDF still very often XSLT. This transformation method is discussed in [1]. Other methods also allow converting from XML to RDFS, but all bring an XSLT conversion as we see in[2] or [3] where several methods are proposed. One of the well-know solution used to transform semi-structured document in RDF is to create a short script to do this conversion.
After some researches on automatic transformation from structured formats, several Frameworks and softwares are relevant. On the Internet it is possible to find easy RDF converters. [4] presents a non-exhaustive list of solutions to convert specialized types (eg. iCalendar, plist, Microformats, etc.) into RDF. In [5], authors use a friendly semi-structured to RDF implementation named tripliser[6]. This tool offers the possiblity to customize the transformation using an XML configuration file that describes transformation rules. Krextor[7] uses the same solution. We can consider that this solution is certainly the most quickest to realize but dependent of the input file. The bad point is that the programmers need to write a transformation script for each different structured file. The tests and validations on these selected solution are really long and often complicated because the programmers need to consider all exceptions of the input structure.
Most of the solutions proposed for realizing this transformation are based on a XSL transformation. Using XSL transformation is an arguable choice. This choice is good because the XSLTs are created to do a structure transformation on a XML file. As one RDF, RDFS and OWL structure representation can be in XML this solution is understandable. But it is not easy to deal with this XSL transformation to realize a correct output as desired. The time to create a correct transformation as desired takes a long time in comparison to scripting

a small transformation code. Once the XSL transformation is written, it is not easy to adjust the output file and often it is easier to restart from a blank XSL transformation.

3 Framework

When the XML or JSON document is first inserted, the program learns its structure or format. Series of rules by default are then generated. The user will subsequently have the ability to edit its rules. The Basic conversion is based on a JSON transformation to RDFS. For XML to RDFS, a first conversion from XML to JSON is made. The interest of this tool is to overcome the lack of flexible tools to perform this kind of operation.

3.1 Workflow process

In figure 1, you can see what is the compound and also the interactions between each of these components.

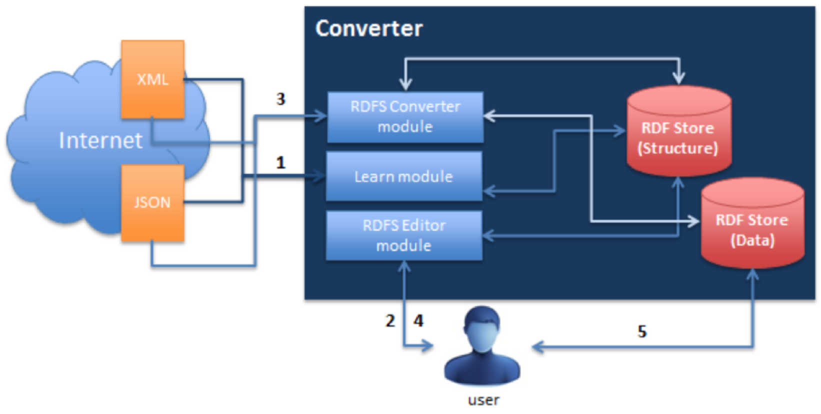


Fig. 1. Workflow Process of the Framework to Convert Semi-Structured Data into RDF

The elements the RDFS Converter contains include the following:

- A database containing the structure as well as the conversion rules necessary to convert a JSON or XML format to the RDFS format conversion criteria imposed by the user.
- A database containing the data converted to RDFS format after having been imported by the user.

– A learning module, which aims to analyze and understand the structure of the XML and JSON documents and create the conversion rules at first time. The conversion rules can be adapted specifically according to the user's choice.
– A publishing module allowing the user to edit the rules of conversion in a graphical manner after creating the structure via the learning module.
– A conversion module, RDFS Converter, which can track either through the conversion rules established automatically or the rules adjusted by the user, to convert a document in JSON or XML into RDFS (creation of triplets in the turtle format - n3).

3.2 Software Implementation

The interface we chose is simple and user-friendly. The user performs a crawler in accordance with some basic detail in the implementation. It will then perform recovery of corresponding results to its crawler. Once the first results are recovered (and not inserted yet), the user can go to the SFSR Converter conversion Editor. He may wish to adapt the outcome of conversion by setting these Data properties and Object properties. He can also change the format of the output structure. Once, the RDFS output corresponding to a crawler defined, it can again initiate the creation of a crawler, and finally import results. The results will be then recorded in RDF directly in Sesame.

3.3 Ontology Rules

The rules of conversion are based exclusively on the RDFS. The proposed ontology below on figure 2 allows containing information (rules) required for the conversion. These rules are planned to be editable by the user for his free adaptation.

– **RecordGroup:** The RecordGroup class is the general element associated with a set of useful conversion rules group. Set of rules for the conversion implies the necessary rules for the conversion of an XML or JSON to a certain structure. For example, the structure of the JSON format returned by LinkedIn public profiles will always be similar for each profile.
– **Page:** The Page class refers to a web page (reachable via a URL). Thanks to its base URL, it will also be used to determine what conversion rules to use. For example, if my Page make reference to http://www.linkedin.com, all elements XML or JSON retrieve via a URL beginning with http://www.linkedin.com will be treated with the same conversion rules.
– **Rule:** The Rule class is one of the most important classes to perform the conversion. A rule is somehow associated with a RecordGroup. A rule is associated with elements or objects of the JSON tree using unique identifier path to reach the object or element followed by the name of the object or element. The rule actually fixes the conversion of the output RDFS, either for an element type or a class object.

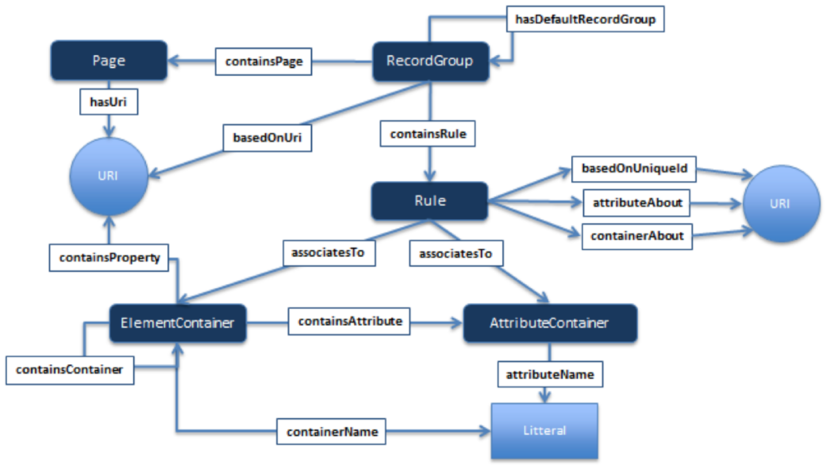


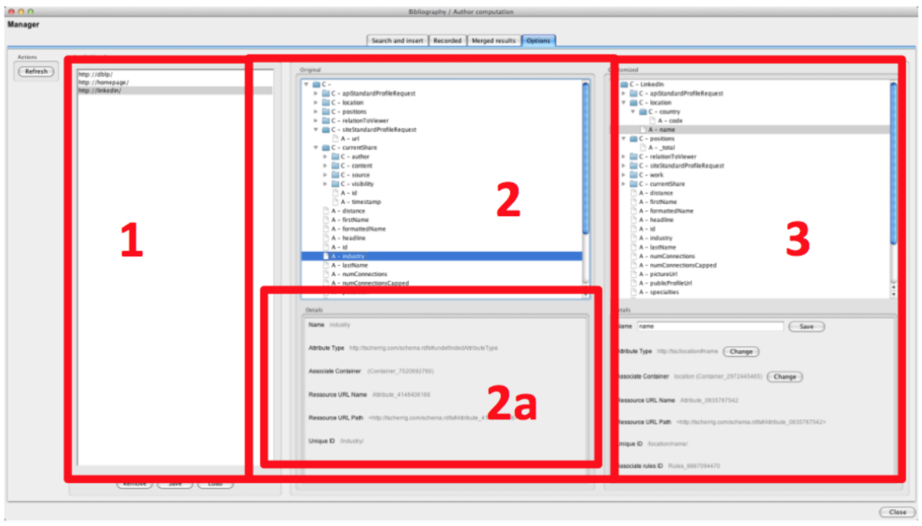
Fig. 2. The proposed RDFS Converter Ontology

- **ObjectContainer:** The ObjectContainer class contains the JSON objects. It is a way to reconstruct the tree entered to the JSON in RDF format (for storage in the DBMS).
- **ElementContainer:** The ElementContainer class contains the elements of the JSON.

3.4 Software Implementation

The screenshot shows below in figure 3 is our conversion application. It is the configurable part of the RDFS Converter. The configuration interface is divided into three distinct sections:

- **Part 1** indicates the source of the structures that will be processed. Indeed, a crawler is expected to return a similar structure for each search result returned. A URL identifies each crawler. This URL is in fact a default base URL from which the information is extracted. In the case presented below, only 3 crawlers are represented. The idea is to associate a structure of data processed in conversion, that is why, in our example, each crawler appears distinctly. Each crawler returning the information is presented under a different structure in JSON or XML. It is possible to remove a tree representation for these types of documents [8].
- **Part 2** of the interface represents this structure in the form of tree. The tree then contains nodes (object) and leaves (element). For each of these nodes or leaves, a full description of creation and information properties related to the object or element is available on *Part 2a*. This tree is only a representation; it is not possible to make changes to its structure.



http://dblp/
http://homepage/
http://linkedin/

```
▼ C -
  ▶ C - apiStandardProfileRequest
  ▶ C - location
  ▶ C - positions
  ▶ C - relationToViewer
  ▼ C - siteStandardProfileRequest
      A - url
  ▼ C - currentShare
    ▶ C - author
    ▶ C - content
    ▶ C - source
    ▶ C - visibility
      A - id
      A - timestamp
    A - distance
    A - firstName
    A - formattedName
    A - headline
    A - id
    A - industry
    A - lastName
    A - numConnections
    A - numConnectionsCapped
```

Name industry
Attribute Type http://tscherrig.com/schema.rdfs#undefinedAttributeType
Associate Container (Container_7520692760)
Ressource URL Name Attribute_4148408166
Ressource URL Path <http://tscherrig.com/schema.rdfs#Attribute_4
Unique ID /industry/

```
C - LinkedIn
  ▶ C - apiStandardProfileRequest
  ▼ C - location
    ▼ C - country
        A - code
      A - name
  ▼ C - positions
      A - _total
  ▶ C - relationToViewer
  ▶ C - siteStandardProfileRequest
  ▶ C - work
  ▶ C - currentShare
    A - distance
    A - firstName
    A - formattedName
    A - headline
    A - id
    A - industry
    A - lastName
    A - numConnections
    A - numConnectionsCapped
    A - pictureUrl
    A - publicProfileUrl
    A - specialties
```

Name name
Attribute Type http://tsc/location#name
Associate Container location (Container_2972445465)
Ressource URL Name Attribute_0835787542
Ressource URL Path <http://tscherrig.com/schema.rdfs#Attribute_0835787542>
Unique ID /location/name/
Associate rules ID Rules_6667094470

Fig. 3. Global preview of the RDF Converter

– **Part 3** is on the other hand, it is editable by the user. Part 2 represents the entry structure, and part 3 the output structure.

3.5 Output Customization by Expert User

The rules have an important role in determining the output format of the RDFS schema. Various mechanisms have been implemented to allow for a maximum adaptation of the format to use. There are unfortunately a number of (logical) restrictions that cannot be applied.

Location of the Elements in the Tree. It is possible to move elements in the tree to facilitate the understanding of the structure or ease the access to the data. Structured JSON and XML documents have in general a straightforward hierarchical structure and should not normally require modification pertaining to the position of information in their structure. The program still offers the possibility to modify this structure. In the example below on Figure 4, it is perfectly possible to move elements *A-code* contained in the CCountry object directly in the object CWork. When a change in structure occurs, several transactions are prohibited (to keep the information consistent). For example, it is inconceivable to place an element deeper in the tree and place an element on another node of the tree.

The basic operation of the conversion at the structural level is simple:

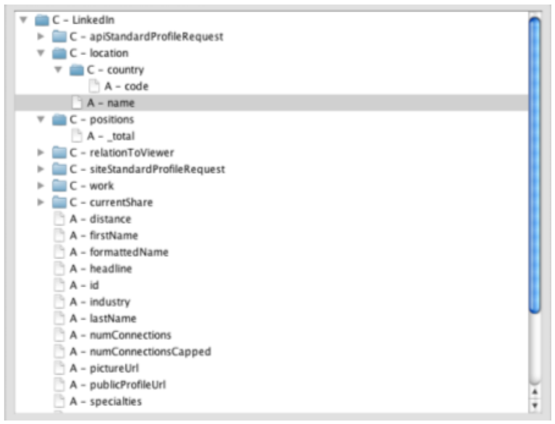


Fig. 4. Preview for an object rules conversion

– Any element that contains another element will be transformed into an RDFS class. Elements with no other elements will be transformed into properties and will have a relationship with their parent element.
– For elements turning into a class, a property object relationship binds the deepest element of the tree with its direct parent.
– The newly generated objects (classes, data properties and objects) will be created dynamically with unique names. The user will then have the possibility to edit and name them according to his choice.

Definition of Properties Associated with Objects of Type Leaves. Figure 5 shows the possible configurations for conversions associated with type leaves. Among the fields available only 3 are editable:

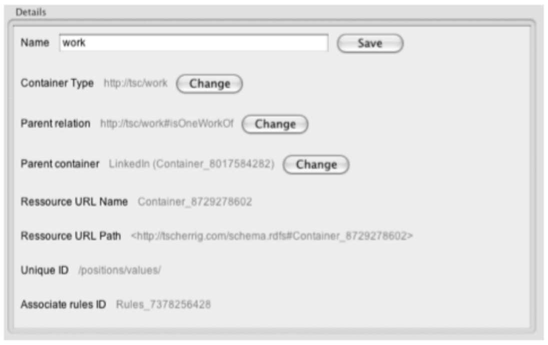
– **Name:** Allows naming an element. This name will then assign references to the elements of this type created during conversion.
– **Attribute Type:** Represents the data type of this element.
– **Associate container:** indicates its position in the tree.

There are also four non-editable properties, including:

– **Resource URL Name:** Short name of the resource URL Path (see below)
– **Resource URL Path:** Refers to the element in the schema where the rules are contained
– **UniqueId:** It is a unique identifier. It comes from the base location from which the element creating the rule. The path is the direct path to the element in a tree representation. The UniqueId begins and always ends with a slash symbol.
– **AssociateRulesID:** Refers to a Rule object defined in the ontology containing the schema and the conversion rule (web-scheme).

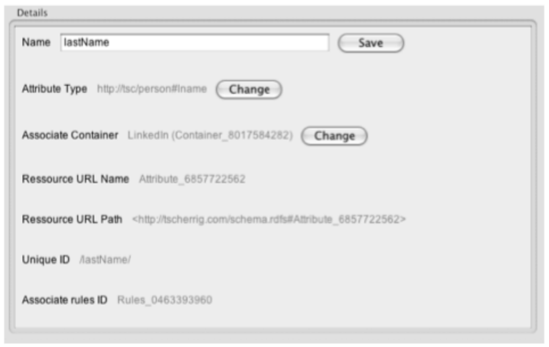
Setting the Property associated with the Nodes. Figure 6 represents the configuration possibilities for converting objects of type nodes. Many fields are identical to properties associated with sheets (see above).
Fixed properties are exactly the same as that of a sheet and provide the same information. The information that differs from the sheet is as follows:

- **Container Type:** Indicates the type of element in the form of a class.
- **Parent relationship:** shows in the form of a property object the relationship between the two properties.
- **Parent container:** indicates its position in the tree.



Name	work
Container Type	http://tsc/work
Parent relation	http://tsc/work#isOneWorkOf
Parent container	LinkedIn (Container_8017584282)
Ressource URL Name	Container_8729278602
Ressource URL Path	<http://tscherrig.com/schema.rdfs#Container_8729278602>
Unique ID	/positions/values/
Associate rules ID	Rules_7378256428

Fig. 5. Preview for an object rules conversion



Name	lastName
Attribute Type	http://tsc/person#lname
Associate Container	LinkedIn (Container_8017584282)
Ressource URL Name	Attribute_6857722562
Ressource URL Path	<http://tscherrig.com/schema.rdfs#Attribute_6857722562>
Unique ID	/lastName/
Associate rules ID	Rules_0463393960

Fig. 6. Preview for an attribute rules conversion

3.6 Preview of a Conversion

An example of a simple conversion is presented in this section. It is a simple conversion because no structural change in the input document is performed on the output result (the objects retain the same elements). With RDFS Converter it is possible to run more complex changes that could alter the structure (positions of the elements) through the tree. The example is the result of a query on *Julien Tscherrig* in DBLP.

```
{
  "author" : {
  "firstName" : "Julien",
  "lastName" : "Tscherrig" },
  "publications" :
    [{
      "title" : "Head-Computer...",
      "year" : "2011",
      "confFullName" : "Human-Computer...",
      "conf" : "HCI (2)",
      "coauthors" : [
        {
          "firstName" : "Francesco",
          "lastName" : "Carrino"
        },
        {
          "firstName" : "Julien",
          "lastName" : "Tscherrig"
        },
        {
          "firstName" : "Elena",
          "lastName" : "Mugellini"
        }
      ]
    }
  ]
}
```

The process converted the data from XML to JSON , and JSON from JSON into triples. If the structure proposed below is rebuilt, one realizes that it is identical to that of the JSON. It is actually the basic structure that was automatically learnt by the system.

```
tsc:dblp_20130
  http://tscherrig.com/schema.rdfs#recordFrom
    http://dblp.uni- trier.de/rec/pers/t/Tscherrig:Julien/xk

tsc:dblp_20130
  http://tscherrig.com/schema.rdfs#searchEngineId
    http://dblp/

tsc:dblp_20130
  http://www.w3.org/1999/02/22-rdf-syntax-ns#type
    tsc:dblp tsc:publication_201305061417310509

http://www.w3.org/1999/02/22-rdf-syntax-ns#type
  tsc:publication
    tsc:publication_201305061417310509
```

```
tsc:person_201305061417310541
  http://www.w3.org/1999/02/22-rdf-syntax-ns#type  tsc:person

tsc:person_201305061417310541
  tsc:author#isAuthorOf  tsc:publication_201305061417310509

tsc:person_201305061417310541
  tsc:person#lname  "Carrino"

tsc:person_201305061417310541
  tsc:person#fname  "Francesco"

tsc:person_201305061417310589
  http://www.w3.org/1999/02/22-rdf-syntax-ns#type  tsc:person

tsc:person_201305061417310589
  tsc:author#isAuthorOf  tsc:publication_201305061417310509

tsc:person_201305061417310589
  tsc:person#lname  "Tscherrig"

tsc:person_201305061417310589
  tsc:person#fname  "Julien"
...
```

4 Use-Cases and Deployment

The software has already been used in several academic projects (Bachelor and Master). Below, we describe some possible use-cases.

4.1 Use-Cases

The software proposed in this paper allows to transform semi-structured data into RDF. Several possible use cases are presented below. The advantage of using this framework compared to other currently available software discussed in the Related Work is the graphical interface for customizing the RDF output. Below we briefly describe two use-cases:

- **Case A:** Semantic developer has to work with a collection a semi-structured documents and needs to use a semantic formats. One of the main semantic formats is RDF and his derivatives. Our framework can help by creating a layer of abstraction between the data sources available in a semi-structured format and the RDF data. Indeed, the program may after customization converts just in time semi-structured data into RDF.
- **Case B:** A web service provides current information as semi-structured information. The developer would like to provides to his consumers this same information but in a semantic format. At this time, the web service's developer could create new data files based on a previous format or also create a special XSLT transformation for all other structured documents

and execute it. Or he cloud use the solution proposed in this paper. The setup is really easy and the process conversion can be processed just in time.

In these two cases, the operations to set up are the same. The developer runs the framework and put, into it one of each semi-structured document type. The system will learn the structure and will create all the corresponding rules for each document type. The developer can then customize these default rules and customizes the output format as desired. All semi-structured documents (already learnt by the system - and customized be the developer) can be converted.

4.2 Academic Projects and the follow-up of this Framework

This framework has been used in several projects. It was used primarily for converting collection of documents to XML and JSON RDF. There was a particular use of the project about Name Disambiguation for Scientist in Computer Science. For this project, several data sources had to be crossed (digital library, collection of publication, etc.). Unfortunately the data were in XML format and it was therefore necessary to convert it.
A web version of the framework presented in this paper has been designed and is currently in development. This application will offer the same opportunities and graphic customization via web services.

5 Tests and Evaluation

Several tests and real deployments were performed for this software. The conversions from XML to RDF have given excellent results. The use of the conversion program was intensely used and many bugs could be resolved through this real deployment. In addition to usability tests carried out on the conversion which have been used, all opportunities were tested by numerous different document structures. Each of these structures was composed of a set of 200 different documents.
In the Related Work part of this paper, several packages were discussed. Each has its advantages and disadvantages. All programs shared a common disadvantage: none provides a graphical interface for personalized conversion. It is from this last point (data sources available in different formats and rarely compatible with a format related to semantic web technologies) that the idea to make a usable graphically tool to adopt to more widely used format conversion rules emerged. This second part helped to demonstrate the functioning of the converter part. This part of the RDFS Converter may in some sense show the lack of tools available for the use of current technologies related to the semantic web. This work proposes a new editing graphical conversion tool of XML or JSON data into RDFS within a fully customizable structure The development of the fusion part allowed demonstrating the functioning of the converter, while offering an innovative solution for searching for information related to the same person.
We present the results of a series of tests we performed for the RDF Converter.

After that the program had initially learnt and rebuilt the basic structure of the application, conversions rules were established. The results were conclusive. They have allowed to demonstrate the robustness of the application as well as facilitated the use of the interface by users with information processing needs. Conversions have given conclusive results and met the expectations of the users.

6 Conclusions and Future Work

The RDFS Converter was built to fill the gap in tools converting semi-structured data into RDF. This software proposes a graphical conversion of XML data into RDF within a fully customizable structure by the end-user expert. It was tested in several academic projects (Bachelor and Master's projects) and allowed demonstrating the functionalities of the converter. Although the software is currently stable, several improvements can still be added, including:

– The ability to perform dynamic updates on the basis of modification to the conversion rules. Indeed, once the conversion rules are laid down, it should no longer be amended to maintain consistency between the data already in the database. It is also important to stop the import of data before the creation of complete conversion rules.
– For the moment, data important does not establish any automatic relationships with objects already in the base. Building such relationships is part of our future work.

7 Acknowledgements

This work was supported by the Haslerstiftung in the context of the Smart World 11005 (Mem0r1es) project.

References

1. Bayu Erfianto, Ahmad Kamil Mahmood, Abdullah Sani, and Abdul Rahman. Modeling Context and Exchange Format for Context-Aware Computing. (December):9–13, 2007.
2. Davy Van Deursen, Chris Poppe, Gäetan Martens, Erik Mannens, and Rik Van De Walle. XML to RDF Conversion: A Generic Approach. *2008 International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*, pages 138–144, November 2008.
3. Mark H Butler, John Gilbert, and Andy Seaborne. Data conversion , extraction and record linkage using XML and RDF tools in Project SIMILE 1 Introduction 2 An overview of the data conversion process 3 Generating RDF encodings of ARTstor and OCW. 2004.
4. Converter to RDF. http://www.w3.org/wiki/ConverterToRdf.

5. Tayfun Gökmen Halaç, Bahtiyar Erden, Emrah Inan, Damla Oguz, Pinar Gocebe, and Oguz Dikenelli. Publishing and linking university data considering the dynamism of datasources. In *Proceedings of the 9th International Conference on Semantic Systems - I-SEMANTICS '13*, page 140, New York, New York, USA, September 2013. ACM Press.
6. Linkeddatatools. http://www.linkeddatatools.com/introducing-rdf-part-2.
7. Krextor An Extensible XML to RDF Extraction Framework, 2009.
8. Taro L Saito. Silk : A Scalable Data Format In-Between Relations and Trees. 4.