Building ontologies for different natural languages

Article *in* Computer Science and Information Systems · June 2014

DOI: 10.2298/CSIS130429023A

3 authors, including:

Emhimed Salem Alatrish

2 PUBLICATIONS **56** CITATIONS

SEE PROFILE

Dušan Tošić

University of Belgrade

83 PUBLICATIONS **732** CITATIONS

SEE PROFILE

Computer Science and Information Systems 11(2):623–644 DOI:10.2298/CSIS130429023A

Building ontologies for different natural languages

Emhimed Salem Alatrish[1], Dušan Tošić[2], and Nikola Milenković[3]

[1] Faculty of Mathematics, Studentski Trg 16,
11000 Belgrade, Serbia
emhimed_alatrish@yahoo.com
[2] Faculty of Mathematics, Studentski Trg 16,
11000 Belgrade, Serbia
dtosic@matf.bg.ac.rs
[3] Faculty of Mathematics, Studentski Trg 16,
11000 Belgrade, Serbia
nikola.milenkovic@live.com

Abstract. Ontology construction of a certain domain is an important step in applying the Semantic web. A number of software tools adapted for building domain ontologies of most wide–spread natural languages are available, but accomplishing that for any given natural language presents a challenge. Here we propose a semi-automatic procedure to create ontologies for different natural languages. Our approach utilizes various software tools available on the Internet most notably DODDLE-OWL - a domain ontology development tool implemented for English and Japanese languages. By using this tool, WordNet, Protégé and XSLT transformations, we propose a general procedure to construct domain ontology for any natural language.

Keywords: Semantic Web, Ontology, Natural language, DODDLE-OWL.

1. Introduction

Semantic Web has lately been a popular and prolific field of research with numerous scientific papers published on the topic so far. Ontology is an important component of the Semantic Web and a lot of papers about applying ontology in specific fields have been published (see [27, 28]). Ontologies are closely connected to Natural Language Processing (NLP) - a field of artificial intelligence, computer science and linguistics. As such, NLP is related to the area of human–computer interaction. Ontologies provide an explicit and formal way for data interpretation, integration and sharing, helping to understand natural (human) language. Understanding of natural language is not an aim per se, but it is useful in different fields, such as: Information Extraction (IE), Machine Translation (MT), Question Answering (QA), etc. (Fig. 1.). Because of that the production of software tools to support ontology and Semantic web has accelerated. A number of these tools are free and available on the Internet (see: [2, 5, 13]).

Unfortunately, most of them are made to work with only a small set of widely used languages such as: English, Spanish, French etc. Some natural languages are not represented in these tools and it is a challenge to create domain ontologies for text written in these languages.

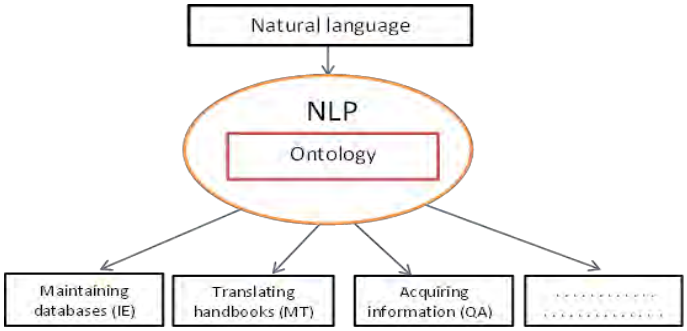

Fig. 1. Relation between natural language, NLP and ontology

Our idea is to combine different accessible software tools for the purpose of semi-automatic construction of Natural Language Ontologies (NLOs) from specific domains. This approach aims to be general and applicable for any natural language. The proposed approach is very ambitious because the problem NLO construction is very general and difficult. Understandably, certain adaptations and constraints are necessary depending on the features of the natural language in question. The domain of applying ontology is important too. Usually, we should make text classification before ontology construction. A good methodology for text classification is proposed in [14].

2. Related work

The problem of NLO construction became apparent immediately after ontologies appeared in computer sciences and it is still present today. This problem is considered in some books (for example, [3], [21]) and in many articles ([15], [29], [8], [30]). In [15], an automatic ontology building method is proposed. The authors described a system which starts from small ontology kernel and constructs the ontology by automatically understanding the text. This system is implemented in the project named Hasti and applied to Persian (Farsi) texts. Paper [8] contains a project description where ontologies are part of the reasoning process used for information management and for the presentation of information. Both accessing and presenting information are mediated via natural language. In [1], an automatic construction of ontology from Arabic texts is proposed, by using statistical techniques to extract elements of ontology. In this work initialization of the ontology is started manually and it is difficult to describe it as a fully automatic process. An approach to converting hierarchical classifications (whose nodes are assigned natural language labels) into lightweight ontologies is proposed in [11]. In paper [12] a model of a Conversational Recommendation Agent (CoRA) is described. It is a domain-specific dialogue system, which implements an ontology-based Natural Language Processing system for shopping situations. The problem of content determination in natural language generation (NLG) is considered in [16]. The authors try to answer the question "What is an A?" where A is a that building ontologies for different natural languages is currently a challenging problem. In recent years, CNL (Controlled Natural Language) has received much attention with regard to ontology ([20], [5]). CNLs, as subsets of natural languages, can eliminate ambiguity of natural

languages and successfully apply in ontology construction. Introducing CNLs, authors impose restrictions on used natural languages.

By developing the area of ontology construction, a lot of new problems are raised. The growing number of ontologies available in different natural languages leads to an interoperability problem. This problem is considered in [9] and a new architecture for a multilingual ontology matching service is proposed. A Framework for merging the heterogeneous ontologies based on WordNet is described in [4]. In fact, a new methodology for merging the different ontologies is introduced. Casual users often use large database. For these purposes it convenient to use Natural Language Interfaces (NLIs) (often referred as closed-domain Question Answering (QA) systems). In [6] system FREyA (Natural Language Interfaces to ontologies) is presented.

3. Semi-automatic creation of NLO

Even though automatic creation of domain NLO has been attempted (see [15]), it is still a difficult task in general. It is particularly challenging to do so for the texts written in different natural languages and related to some domain. In this case the domain ontology structure depends in some aspects on human users. Because of that, it is convenient to provide refined semi-automatic software tool for building NLOs. Those kind of tools are available on the Web ([24], [5]) and one of them is DODDLE-OWL (see: [18] and [26]). DODDLE-OWL is an interactive domain ontology development environment created for Japanese and English language. We adopted this environment for any natural language (that has a dictionary on WordNet) by applying translation of original text into English text and transforming the obtained English ontology. Since DODDLE-OWL is an essential tool in our approach, we are going to describe it in more detail.

3.1. DODDLE-OWL Overview

DODDLE-OWL (a Domain Ontology rapiD DeveLopment Environment - OWL extension) is a domain ontology development tool for the Semantic Web. It is written in Java language. According to [7], "DODDLE-OWL reuses existing ontologies such as WordNet and EDR as general ontologies to construct taxonomic relationships (defined as classes) and other relationships (defined as properties and their domains and ranges) for concepts". An initial concept hierarchy is constructed as a (is-a) hierarchy of terms. Here, it is assumed that there are one or more domain specific documents and that the user can select important terms needed to construct domain ontology. DODDLE-OWL has the following six main modules: Ontology Selection Module, Input Module, Construction Module, Refinement Module, Visualization Module, and Translation Module. We assume that there are one or more domain specific documents, and we also assume that the user can select important terms needed to construct domain ontology (Fig. 2, see [18] and [26]).

First, as an input to DODDLE-OWL, the user selects several concepts in Input Module. In Construction Module, DODDLE-OWL generates the basis of the ontology, an initial concept hierarchy and set of concept pairs, by referring to appropriate

reference ontologies and documents. In Refinement Module the initial ontology, generated by Construction Module, is refined by the user through interactive support by DODDLE-OWL. The ontology constructed by DODDLE-OWL can be exported with the representation of OWL. Finally, Visualization Module (MR3) (described in [18]) is connected with DODDLE-OWL and works with a graphical editor ([26]).

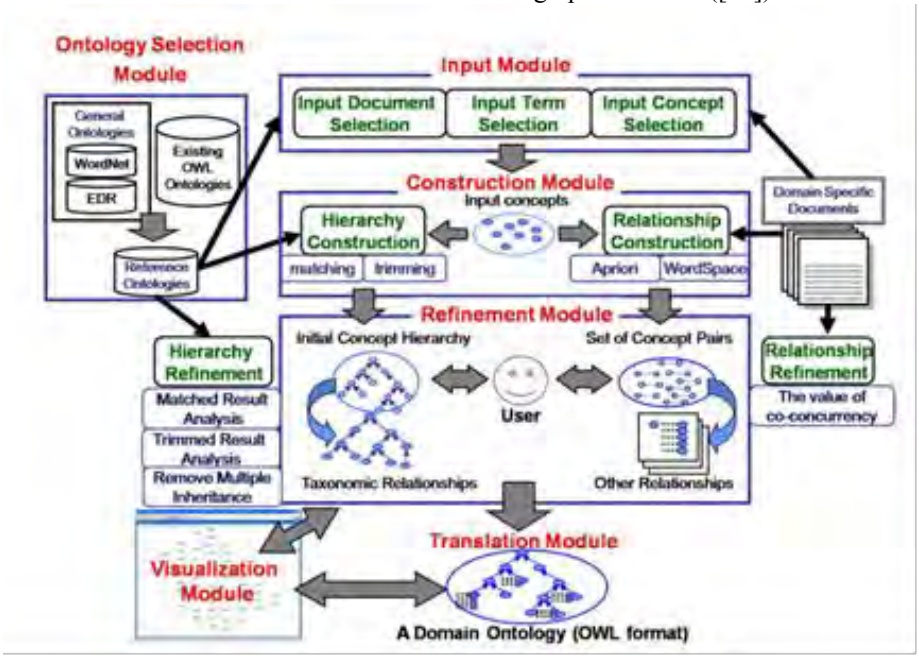

Fig. 2. Overview of DODDLE-OWL

3.2. Construction of domain NLO for different languages

In order to construct the NLO for different languages, text document from any natural language is translated to English language. English ontology is built by using DODDLE-OWL. In DODDLE-OWL the following steps are executed:

1. In the Ontology Selection Module, user selects reference ontologies on WordNet, EDR (general vocabulary dictionary or technical terminology dictionary), and existing OWL ontologies in the ontology selection as shown in Fig. 3.

2. In the Input Document Selection Module, user selects domain specific documents described in English. In this step, some words in the documents are extracted. During the same phase, user can select a part of speech (POS) for extraction of words from the documents. For example, if noun or verb words are extracted, checkbox "Noun" or "Verb" should be checked as shown in Fig. 3.

3. In the Input Term Selection Module, a list of extracted terms is formed. This list includes (for more details see [3] and [14]): compound words, part of speech (POS), Term Frequency (TF of term t in document d is defined as the number of times that t occurs in d), Inverse Document Frequency (IDF estimate the rarity of a term in the whole document collection - if a term occurs in all the documents of the collection, its

IDF is zero) and TF-IDF in the documents (TF-IDF is weight of a term - the product of its TF weight and its IDF weight). Domain specific documents contain many significant compound words. Therefore, accurate extraction of compound words is necessary to construct domain ontologies. At this step, while considering part of speech (POS), TF, and so on, the user selects input terms which are significant terms for the domain. For certain domains, important terms do not occur in the documents. In such case, the Input Term Selection Module has a function, allowing the manual addition of important terms as input terms by the user. In order to prevent the leakage of the selection of input terms from the documents, the Input Term Selection Module maintains the relationships between the extracted terms and the terms in the documents as shown in Fig. 3.

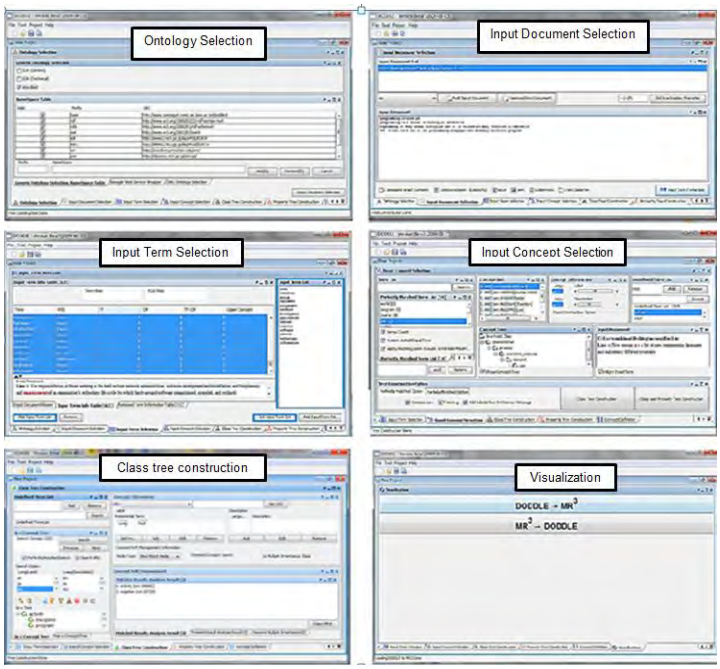

Fig. 3. Typical usage of DODDLE-OWL

4. In the Input Concept Selection Module, the user identifies the word sense of input terms in order to map those terms to the concepts in the reference ontologies selected with the Ontology Selection Module. A particular single term may have many word senses. Therefore, there may be many concepts corresponding to a word. The Input Concept Selection Module has a function enabling automatic word disambiguation. This function shows the list of concepts, ordered by some criteria, corresponding to the selected input term. Input term not corresponding to the labels of concepts in the reference ontologies is marked as undefined. The input terms are also undefined if the concept exists, but there are no appropriate concepts in the reference ontologies. The user defines the undefined terms manually in the refinement module, as shown in Figure 3.

5. The Hierarchy Construction Module automatically generates the basis of ontology, an initial concept hierarchy (by referring to reference ontologies) and documents. An initial concept hierarchy is constructed as a taxonomic relationship.

6. DODDLE-OWL uses MR3: Meta-Model Management based on RDFs Revision Reflection [18] as the Visualization Module. Figure 4. shows the product of MR3 as RDFs description and graphical representation. Finally, through the translation module, we can export the constructed domain ontology described in RDFs. For example, a portion of the obtained English Ontology OWL code is presented in the following document (1):

```
<rdf:Description rdf:about="use">
  <rdfs:subClassOf rdf:resource="activity"/>
  <rdfs:comment xml:lang="en">the act of using; "he warned
against the use of narcotic drugs"; "skilled (1) in the
utilization of computers"</rdfs:comment>
  <rdfs:label xml:lang="en">use</rdfs:label>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
```

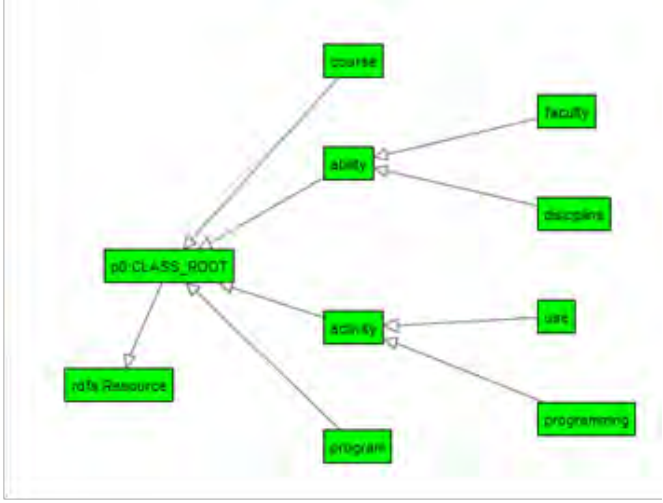

Fig. 4. Products of MR3

7. To build ontology represented by OWL, we use Protégé editor ([23]). Protégé has plugin to enhance ontology development, such as the OWL plugin (see: [10]). We use this possibility to get the OWL document. For example, the document (1) is transformed in the following text.

```
<owl:Class rdf:ID="use">
  <rdfs:label xml:lang="en">use</rdfs:label>
  <rdfs:comment xml:lang="en">
    the act of using; "he warned against the …
  </rdfs:comment>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="activity"/>
  </rdfs:subClassOf>
</owl:Class>
```

The similar ontology graph for the related English words could be generated as in Fig. 4. by using Protégé editor.

8. Very important step in localization process is translation of the ontology recognized in a source language into target language by using XSLT. We are looking for all tags `<rdfs:label>`, `<owl:Class>` (this includes `rdf:ID` and `rdf:about`) and `<rdfs:subClassOf>` (this includes attributes `rdf:about` and `rdf:resource`) and duplicates them. XSLT Translation script is included in this paper and also publically available online[1].

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <xsl:output method="xml" indent="yes"/>
  <xsl:variable name="dict"
                select="document('dictionary.xml')/*" />
  <xsl:variable name="sourceLanguage" select="$dict/@from" />
  <xsl:variable name="targetLanguage" select="$dict/@to" />

  <xsl:template match="@* | node()">
    <xsl:copy>
      <xsl:choose>
        <xsl:when test=".=rdfs:label">
         <xsl:apply-templates select="rdfs:label"/>
       </xsl:when>
        <xsl:when test=".=rdfs:subClassOf ">
         <xsl:apply-templates
                          select="rdfs:subClassOf "/>
         </xsl:when>
        <xsl:otherwise>
         <xsl:apply-templates select="@* | node()"/>
       </xsl:otherwise>
      </xsl:choose>
    </xsl:copy>
  </xsl:template>

  <xsl:template match="rdfs:label">
    <xsl:variable name="word" select="." />
```

[1] https://github.com/nikolamilenkovic/doddle-owl-rdf-translator

```
      <rdfs:label xml:lang="{$sourceLanguage}">
        <xsl:value-of select="$word"/>
      </rdfs:label>
      <rdfs:label xml:lang="{$targetLanguage}">
        <xsl:value-of select="$dict/word[@name=$word]"/>
      </rdfs:label>
  </xsl:template>

  <xsl:template match="owl:Class">
    <xsl:if test="@rdf:ID">
      <xsl:variable name="word" select="@rdf:ID" />
      <xsl:variable name="translated_word">
        <xsl:value-of select="$dict/word[@name=$word]"/>
      </xsl:variable>

      <owl:Class rdf:ID="{$translated_word}">
        <xsl:apply-templates />
      </owl:Class>
    </xsl:if>

    <xsl:if test="@rdf:about">
      <xsl:variable name="word"
                    select="substring(@rdf:about, 2)" />
      <xsl:variable name="translated_word">
        <xsl:value-of select="$dict/word[@name=$word]"/>
      </xsl:variable>

      <owl:Class rdf:about="{concat('#',$translated_word)}">
        <xsl:apply-templates />
      </owl:Class>
    </xsl:if>
  </xsl:template>

  <xsl:template match="rdfs:subClassOf">
    <xsl:choose>
      <xsl:when test="@rdf:resource">
        <xsl:variable name="word"
                      select="substring(@rdf:resource, 2)" />
        <xsl:variable name="translated_word">
          <xsl:value-of select="$dict/word[@name=$word]"/>
        </xsl:variable>
        <rdfs:subClassOf
             rdf:resource="{concat('#',$translated_word)}" />
      </xsl:when>
      <xsl:otherwise>
        <xsl:copy>
          <xsl:apply-templates select="@* | node()"/>
        </xsl:copy>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
</xsl:stylesheet>
```

For example, if the input document includes these tags:

```
<owl:Class rdf:ID="use">
  <rdfs:label xml:lang="en">use</rdfs:label>
  <rdfs:comment xml:lang="en">
    the act of using; "he warned against the …
  </rdfs:comment>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="activity"/>
  </rdfs:subClassOf>
</owl:Class>
```

and the target language is Serbian-Cyrillic, we will get document like this one:

```
<owl:Class rdf:ID="користити">
  <rdfs:label xml:lang="sr">користити</rdfs:label>
  <rdfs:comment xml:lang="en">
    the act of using; "he warned against the …
  </rdfs:comment>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="активност"/>
  </rdfs:subClassOf>
</owl:Class>
```

After that, we can generate an ontology graph for related Serbian words from a specified text (Fig. 5.).

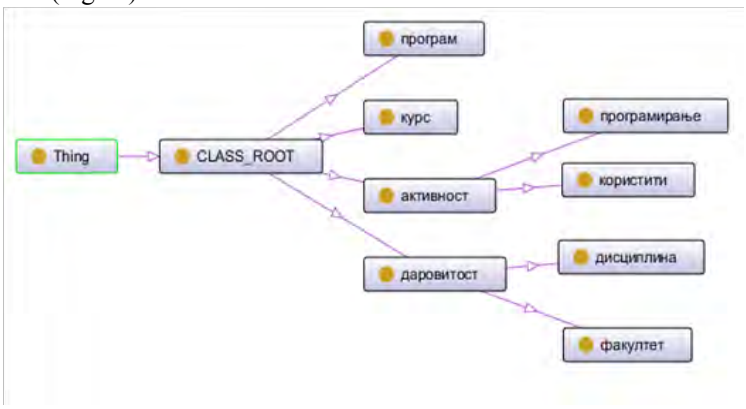

Fig. 5. Ontology graph for related Serbian words

The whole process could be graphically described as in Fig. 6.

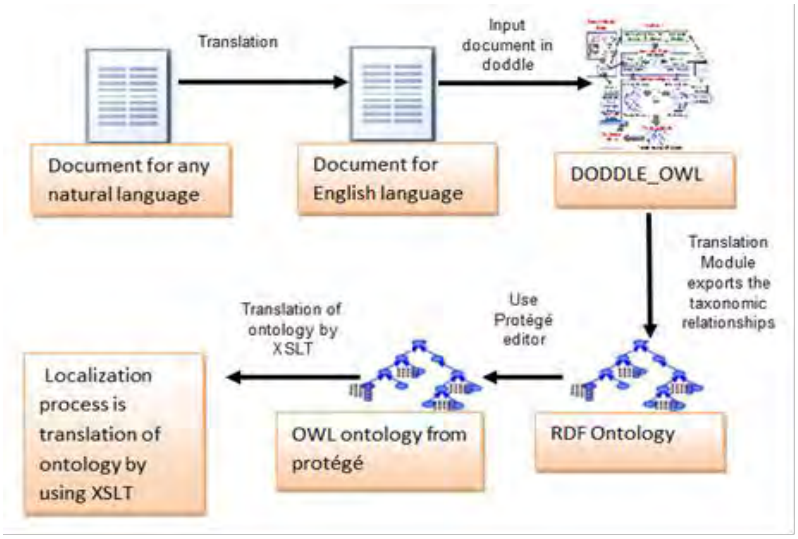

Fig. 6. Building of NLO

3.3. Creating dictionary for the translation process

By applying DODDLE-OWL, after finishing step 4, the project is completed and saved. When the project is saved, a file named "InputWordSet" is automatically created. This file contains all English words selected from Input Term Info Table. This file consequently contains all relevant words for the target ontology. These words should be translated into corresponding words in the target language. Here we utilize MyMemory online translation service [19]. MyMemory is the world's largest Translation Memory (TM). It has been created collecting TMs from the European Union, United Nations and aligning the best domain-specific multilingual websites.

MyMemory's translation service is accessible over the Internet via their translation API. We wrote WordTranslator[2] console application in .NET Framework 4.5 which utilizes their translation API. Performing translation of a single word is done by calling WordTranslator with the following arguments: word to be translated, language of the provided word and the desired language for the translation. Program outputs a single translated word. This greatly simplifies the translation process, since the complexity of

[2] https://github.com/nikolamilenkovic/word-translator

natural language translation is reduced to a single console command execution. (If some words are not translated, then we use Google translator or any available tool for translation.)

Since WordTranslator translates a single word at a time, we also created a script[3] which automatically iterates over each word in the input word list and translates each word individually. Script is written in Batch shell scripting language and its source code is:

```
@ECHO OFF & SetLocal EnableExtensions EnableDelayedExpansion
::Input file name
SET in=%1
::Input language (for example: en)
SET inLang=%2
::Output language (for example: ar)
SET outLang=%3
::Output file name (for example: dictionary.xml)
SET out=%4
::Check if output file exists
IF EXIST %out% (
  ECHO %out% already exists! Deleting...
    DEL %out%
    ECHO Deleted!
)

ECHO ^<?xml version="1.0" encoding="UTF-8"?^> >> %out%
ECHO ^<dictionary from="%inLang%" to="%outLang%"^> >> %out%

FOR /F "tokens=*" %%l in (%in%) do call :TRANSLATE %%l
ECHO ^</dictionary^> >> %out%
ECHO Dictionary created. Perform manual check before usage.

GOTO :EOF

:: ----------------- PROCEDURE TRANSLATE -----------------------
:TRANSLATE
  SET inputWord=%1
  FOR /f "delims=" %%a in ('WordTranslator.exe --word %inputWord% --
inputLanguage %inLang% --outputLanguage %outLang%') DO SET
outputWord=%%a
```

[3] https://gist.github.com/nikolamilenkovic/9169523

```
   ECHO Translated %inputWord% to %outputWord%
   ECHO ^<word name="%inputWord%"^>%outputWord%^</word^> >> %out%
   EXIT /b
:: ------------ END OF PROCEDURE: TRANSLATE --------------------
```

WordTranslator can be used for any combination of input/output languages. Since we are starting with English ontologies, all our input words will be in English, and output in the target language. In the Batch script we use outputs generated by WordTranslator to construct dictionary used by XSLT transformation. Format of the dictionary as follows (mapping English to Serbian words):

```
<?xml version="1.0" encoding="utf-8"?>
<dictionary from="en" to="sr">
   <word name="activity">активност</word>
   <word name="abstraction">апстракција</word>
   <word name="group">група</word>
   <word name="creative_activity">креативна_активност</word>
   <word name="sun">сунце</word>
</dictionary>
```

After building the English OWL representation of our text in step 5, we transform this document into Serbian OWL representation by using XSLT transformer. For this purpose we use an XML editor. There are several available XML editors (see: [17] and [22]). For example, by using Oxygen XML Editor [10], we get the workspace organized as in Fig. 7.

Fig. 7. XSLT transformer applied to Oxygen XML Editor

4. Examples

In this section we present two examples of applying our procedure for Arabic and French. We started with the following Arabic text:

البرمجة هي فن جميل. البرمجة هي المقرر التعليمي في كلية الرياضيات. البرمجة هو الانضباط فعالية للغاية ويتم تعليمها في العديد من الكليات في العالم. الآن يمكننا استخدام الكثير من لغات البرمجة الجديدة وجعل العديد من البرامج المختلفة

The above Arabic text is translated into English as follows:

"Programming is a beautiful art. Programming is a course in the Faculty of Mathematics. Programming is a discipline very effective and it is learned in many colleges in the world. Now we can use a lot of new programming languages and make many different programs."

After applying the DODDLE-OWL, the obtained document is used by Protégé editor to get the ontology for the above-mentioned English text. The obtained ontology document is very long and here we present only a part of this document that is related to the notions: “discipline” and “course”:

```
</owl:Class>
  <owl:Class rdf:about="discipline">
    <rdfs:subClassOf>
      <owl:Class rdf:about="activity"/>
    </rdfs:subClassOf>
    <rdfs:label xml:lang="en">discipline</rdfs:label>
    <rdfs:comment xml:lang="en">training to improve strength
     or self-control</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="course">
    <rdfs:label xml:lang="en">course</rdfs:label>
    <rdfs:comment xml:lang="en">a line or route along which
     something travels or moves; "the hurricane demolished
     houses in its path"; "the track of an animal"; "the
     course of the river"</rdfs:comment>
    <rdfs:subClassOf>
```

By using the Protégé editor, we can generate ontology graph for related English words from specified text (Fig. 8.).

Fig. 8. Ontology graph for English words

A part of XSLT transformation for this text has the form as in Fig. 9.

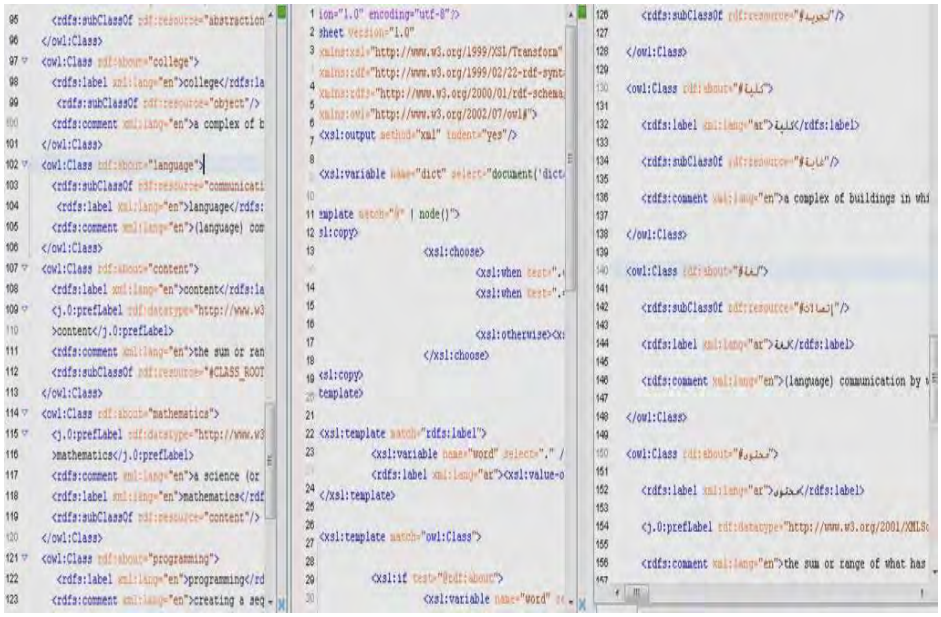

Fig. 9. XSLT transformer applied Arabian text

A corresponding piece of the obtained ontology for the previous Arabic text (the entire document is too long and will therefore not be presented in its entirety) looks like the following one:

```
</owl:Class>
  <owl:Class rdf:about="#ا ال نبخطا">
```

```
    <rdfs:subClassOf>
      <owl:Class rdf:about="#طاشن"/>
    </rdfs:subClassOf>
    <rdfs:label xml:lang="ar">الانـ ضبـاط</rdfs:label>
    <rdfs:comment xml:lang="en">training to improve strength
    or self-control</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="#المقرر_التعليمي">
  <rdfs:label xml:lang="ar">المقـرر التـعـلـيـمي</rdfs:label>
    <rdfs:comment xml:lang="en">a line or route along which
     something travels or moves; "the hurricane demolished
     houses in its path"; "the track of an animal"; "the
     course of the river"</rdfs:comment>
  <rdfs:subClassOf>
```

By applying the Protégé editor to the obtained document, we can generate an ontology graph for related Arabic word from a specified text (Fig.10.). From this graph we can see the relations between concepts in given text.

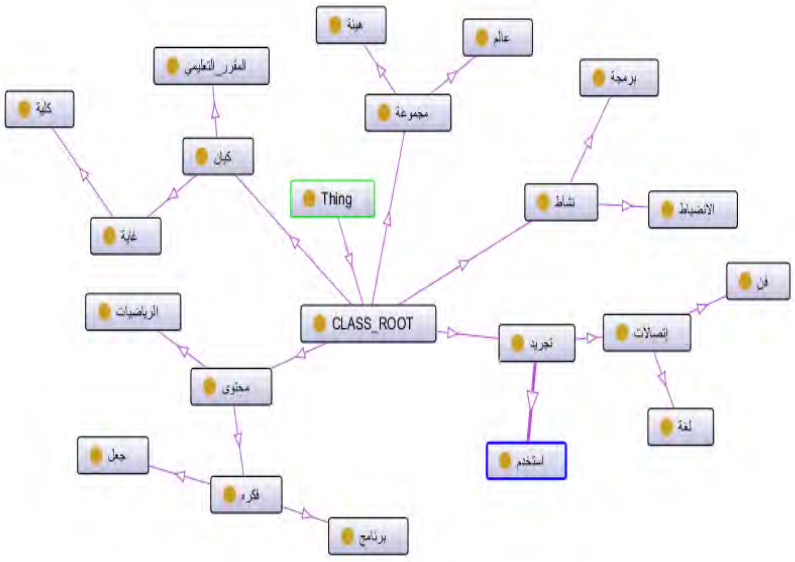

Fig. 10. Ontology graph for Arabic words

The second example is related to French language. For the following text:

Un thésaurus est constitué d'un ensemble organisé de termes, choisis pour leur capacité à faciliter la description d'un domaine et à harmoniser la communication et le traitement de l'information. Les termes d'un thésaurus sont reliés entre eux par des relations sémantiques (hiérarchique, équivalence, etc.).

A fraction the OWL document is below:

```
<owl:Class rdf:about="description">
  <rdfs:subClassOf>
```

```
    <owl:Class rdf:about="knowledge"/>
  </rdfs:subClassOf>
  <rdfs:label xml:lang="en">description</rdfs:label>
  <rdfs:comment xml:lang="en">sort or variety; "every
description of book was there"</rdfs:comment>
</owl:Class>
```

By applying the Protégé editor, as in previous example, we get an ontology graph for an English text (Fig. 11.).

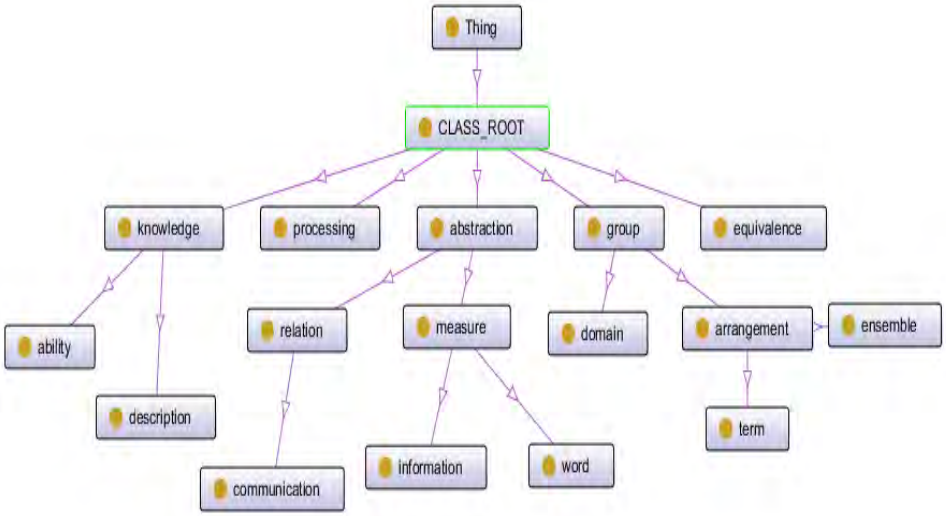

Fig. 11. Ontology graph for English words

By using the XSLT transformer (Fig. 12.), we generate a corresponding OWL document for French language.

```
<owl:Class rdf:about="#description">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#connaissance"/>
  </rdfs:subClassOf>
  <rdfs:label xml:lang="fr">description</rdfs:label>
  <rdfs:comment xml:lang="en">sort or variety; "every
  description of book was there"</rdfs:comment>
</owl:Class>
```

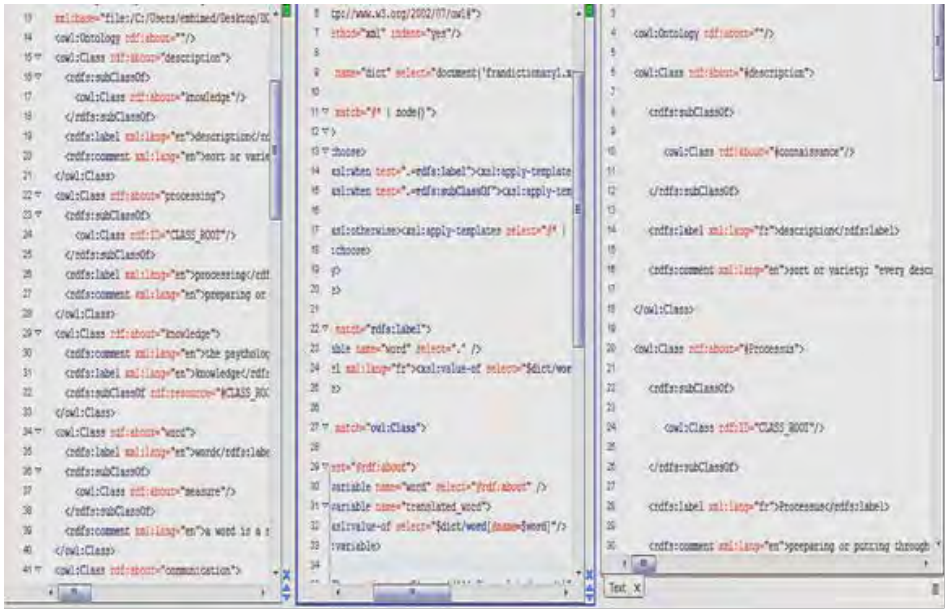

Fig. 12. Generating ontology for French language by applying XSLT transformer

The corresponding ontology graph is presented in Fig. 13.

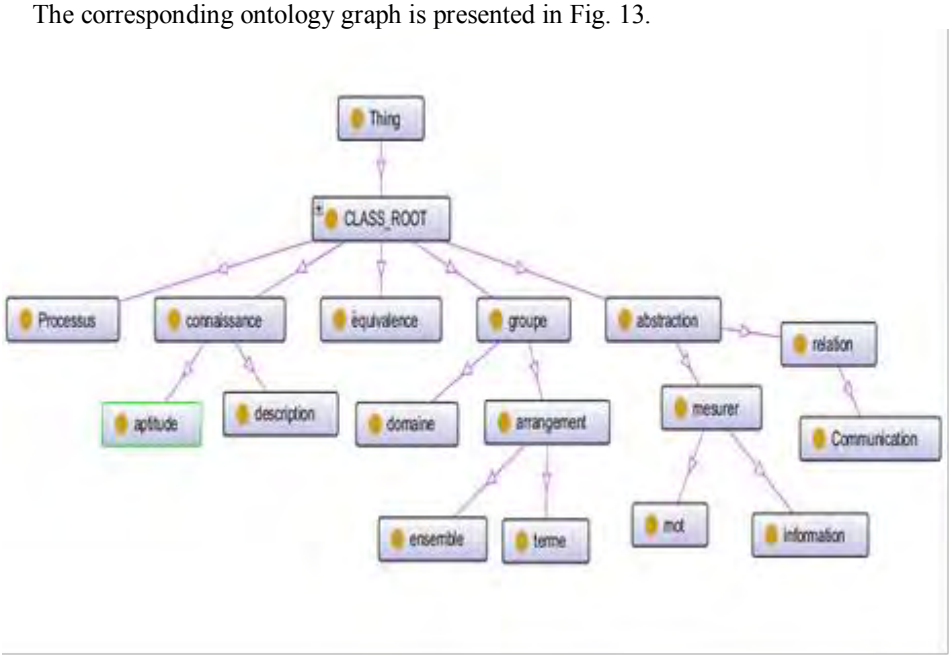

Fig. 13. Ontology graph for French words

Consider another more complex example. Let us have the following French text:

L'informatique est le domaine d'activité scientifique, technique et industriel concernant le traitement automatique de l'information via l'exécution de programmes informatiques par des machines : des systèmes embarqués, des ordinateurs, des robots, des automates, etc.

Ces champs d'application peuvent être séparés en deux branches, l'une, de nature théorique, qui concerne la définition de concepts et modèles, et l'autre, de nature pratique, qui s'intéresse aux techniques concrètes d'implantation et de mise en œuvre sur le terrain. Certains domaines de l'informatique peuvent être très abstraits, comme la complexité algorithmique, et d'autres peuvent être plus proches d'un public profane. Ainsi, la théorie des langages demeure un domaine davantage accessible aux professionnels formés (description des ordinateurs et méthodes de programmation), tandis que les métiers liés aux interfaces homme-machine sont accessibles à un plus large public.

Le terme « informatique » résulte de la combinaison des trois premières syllabes du terme « information » et des deux dernières syllabes du terme « automatique » ; il désigne à l'origine l'ensemble des activités liées à la conception et à l'emploi des ordinateurs pour traiter des informations. Dans le vocabulaire universitaire américain, il désigne surtout l'informatique théorique : un ensemble de sciences formelles qui ont pour objet d'étude la notion d'information et des procédés de traitement automatique de celle-ci, l'algorithmique. Par extension, la mise en application de méthodes informatiques peut concerner des problématiques annexes telles que le traitement du signal, la calculabilité ou la théorie de l'information.

After applying our method, the following ontology graph is obtained (we omit intermediate ontology graph for English words and parts of XSLT transformer):

Fig. 14. Ontology graph for French words

5. Discussion

The strong point of our approach is its generality, i.e. the possibility to apply it in same way for any natural language. Our starting goal was to create automatic method for building domain ontologies related to any natural language. Moreover, we conceive that it is not possible to do in this moment. So, we include expert in generating of ontology and make a semi-automatic approach. Participating of an expert (in selection some words) is probably the weakest point of our method. Also, the problems could appear during translation of some text into English language. A lot of new questions arise. For example, let we have a text in the natural language NL1 (denote it with t.NL1) and translate this text into natural language NL2 (denote it with t.NL2). After applying of our method to t.NL1 and t.NL2 will we get same ontology graph, at least will we get similar ontology graph. Special problem is how to measure the similarity of graphs. These problems could be subject of further research.

6. Conclusion and future work

Ontologies are very important in different scientific fields such as: knowledge engineering and representation, information retrieval and extraction, knowledge management, agent systems, and so on. We can say that ontologies represent the backbone of the semantic web. The possibility to create ontology for any natural language gives us an opportunity to work with information that can be processed by both humans and computers in a natural way which is, unfortunately, still difficult to do that automatically. However, semi-automatic implementation of this process, including a human expert, is possible. We described our approach for discovering taxonomic conceptual relations from text facility ontology by using open source software tool DODDLE-OWL. The main challenge we faced is that this software is available only for Japanese and English languages. To address that, we proposed the procedure where DODDLE-OWL is used as an auxiliary tool to build an ontology from the given text for any natural language (referred in our paper as target language). For this approach the other auxiliary tools are necessary as well as an existing WordNet database for the target language, Protégé semantic web editor and Oxygen XML Editor. The main contribution of this paper is the integration of different software tools, which gives new quality and provides the building of ontologies for different natural languages. We plan to perform further analysis of the results and compare the obtained ontology trees using different natural languages with the same input text. We will try to improve the proposed approach by integrating additional software tools and making certain steps simpler.

Acknowledgment. Research was partially supported by the Ministry of Education and Sciences Republic of Serbia, through project no. OI 174010 Mathemaical models and optimization methods of large systems.

References

1. Ahmed, C, M., Hassina, A., Zaia, A.: Automatic Construction of Ontology from Arabic Texts. ICWIT 2012: 193-202. (2012)
2. Alatrish S.E.: Coparison of Ontology Editors, eRAF Journal on Computing, Vol. 4, 23-38, 2012.
3. Antoniou G. and van Harmelen F.: Semantic Web Primer, The MIT Press, 2008.
4. Bajwa S. I.: A Framework for Ontology Creation and Management for Semantic Web, International Journal of Innovation, Management and Technology, Vol. 2, No. 2, April 2011.
5. Bernstein A. and Kaufmann E.: GINO – A Guided Input Natural Language Ontology Editor, Lecture Notes in Computer Science Volume 4273, pp 144-157, 2006.
6. Damljanovic D., Agatonovic M. and Cunningham H.: Natural Language Interfaces to Ontologies:Combining Syntactic Analysis and Ontology-based Lookup through the User Interaction, in: Proc. of the 7th Extended Semantic Web Conference (ESWC) (ed. Lora Aroyo and al.), Springer Verlag, 106-120, 2010.
7. DODDLE-OWL,a Domain Ontology rapiD DeveLopment Environment - OWL extension, [Online]. Available: http://doddle-owl.sourceforge.net/en/.
8. Dominique, E., Chris, N., Andrew, Z.: Towards Ontology-based Natural Language Processing". RDF/RDFS and OWL in Language Technology: 4th Workshop on NLP and XML (NLPXML-2004), ACL 2004, Barcelona, Spain, [Online]. Available: www.aclweb.org/anthology-new/W/W04/W04-0609.pdf. (2004)
9. Haytham A-F., Schafermeier R. and Paschke P.: An Inter-lingual Reference Approach For Multi-Lingual Ontology Matching, IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 2, No 1, pp. 497- 503, 2013.
10. Holger, K., Ray, F., Natalya, N., Mark, M.: The Protégé OWL plugin: An open development environment for semantic web applications, The Semantic Web–ISWC 2004, 229-243. (2004)
11. lya, Z., Lei, S., Fausto, G., Wei, P., Qi, J., Mingmin, C., Xuanjing, H.: From Web Directories To Ontologies: Natural Language Processing Challenges, Technical Report # DIT-07-029. May (2007)
12. Janzen, S., Maass, W.: Ontology-based Natural Language Processing for In-store Shopping Situations, Third IEEE International Conference on Semantic Computing (ICSC 2009), Berkeley, California. USA (2009)
13. Kapoor B. and Sharma S.: A Comparative Study Ontology Building Tools for Semantic Web Applications, International Journal of Web & Semantic Technology , vol. 1, no. 3, pp. 1-13, 2010.
14. Karanikolas N. and Skourlas C.: A parametric methodology for text classification, Journal of Information Science, Vol. 36 (4), pp. 421-442, 2010**.**
15. Mehrnoush, S., Ahmad, A.: Learning Ontologies from Natural Language Texts, International Journal of Human-Computer Studies, Volume 60, Issue 1, January, Pages 17-63. (2004)
16. Mellish, C., Pan, J.: Natural Language Directed Inference from Ontologies, Artificial Intelligence 172(10): 1285-1315. (2008)
17. Microsoft Core XML Services (MSXML) 6.0, [Online]. Available: http://www.microsoft.com/enus/download/details.aspx?id=3988.
18. Morita, T., Izumi, N., Fukuta, N.: A Graphical RDF-Based Meta-Model Management Tool. IEICE Transactions 89-D(4): 1368-1377. (2006).
19. MyMemory: next generation Translation Memory technology, [Online]. Available: http://mymemory.translated.net/doc/.
20. Namgoong H. and Kim H-G.: Ontology-Based Controlled Natural Language Editor Using CFG with Lexical Dependency, ISWC/ASWC, 353-366, 2007.
21. Nitin, I., Fred, J, D (editors).: Handbook of Natural Language Processing, Second Edition, Taylor & Francis. (2010)

22. Oxygen XML Editor 14.2. [Online]. Available: http://www.oxygenxml.com/doc/Editor-UserManual.pdf
23. Protégé ontology editor developed by Stanford Medical Informatics, Stanford University School of Medicine, further information, [Online]. Available: http://protege.stanford.edu/.
24. Szulman S., Charlet J., Aussenac-Gilles N., Nazarenko A., Sardet E. and Téguiak H.V.: DAFOE: An Ontology Building Platform - From Texts or Thesauri, KEOD 2009: 372-375
25. Takeshi, M., Naoki, F., Noriaki I., Takahira, Y.: DODDLE-OWL: Interactive Domain Ontology Development with Open Source Software in Java. IEICE Transactions 91-D(4): 945-958. (2008)
26. Takeshi, M., Naoki, F., Noriaki I., Takahira, Y.: DODDLE-OWL: A Domain Ontology Construction Tool with OWL. ASWC 2006: 537-551. (2006)
27. Vesin, B., Ivanović, M., Klašnja-Milićević, A., Budimac, Z.: Ontology-Based Architecture with Recommendation Strategy in Java Tutoring System. Computer Science and Information Systems, Vol. 10, No. 1, 237-261. (2013)
28. Wei, M., Xu, J., Yun, H., Xu, L.: Ontology-Based Home Service Model. Computer Science and Information Systems, Vol. 9, No. 2, 813-838. (2012)
29. Wilson, W., Liu, W., and Bennamoun, M. 2012. Ontology learning from text: A look back and into the future, ACM Comput. Surv. 44, 4, Article 20 (August 2012), 36 pages, DOI=10.1145/2333112.2333115 http://doi.acm.org/10.1145/2333112.2333115
30. Wolf, F. and Bernhard, B.: Combining Ontologies And Natural, Language, [Online]. Available: krr.meraka.org.za/~aow2010/Fischer-etal.pdf. (2010)

Emhimed Salem Alatrish was born in 03/09/1972 at Yefren in Libya. He finished primary and secondary school in Yefren, and then graduated from the Faculty of Science and Arts in 1996. After that he started master study in Cultural University in Istanbul and ended it in 2005/2006 school year. From 2008 year he is enrolled in the doctoral program at the Faculty of Mathematics in Belgrade.In the period from 1997 to 2001 year he worked as an assistant professor at the Department of Physics, Faculty of Science and Art in Yefren in Libya.

Dušan Tošić received his B.Sc. degree in mathematics 1972, M.Sc. in mathematic 1977 and PhD in mathematics 1984 from University of Belgrade, Faculty of Mathematics. Since 2003 he is a full professor of computer science at Faculty of Mathematics, University of Belgrade. Professor Tošić is an associate editor for the Journal ComSIS (Computer Science and Information Systems) and reviewer of Zentarblat. His publication list contains more than 100 titles of articles and books. He has supervised about 20 candidates for PhD and M.Sc. degree. On the beginning his research interest was numerical solving of differential equations. After that he was oriented toward parallel algorithms, optimizations, genetic algorithms and teaching of computer sciences. He is Fellow of Serbian Mathematical Society and member of Committee Serbian Society of Computer Sciences.

Nikola Milenković was born in 11/18/1987 in Belgrade. He finished primary school in Barič and secondary school "Nikola Tesla" in Belgrade, and then graduated at the Faculty of Mathematics in 2009, Computer Science course. After that he started master studies at the Faculty of Mathematics and ended it in 2011. On the same year he enrolled doctoral studies at the Faculty of Mathematics. Since 2009 he worked at a various IT companies as a software developer, team leader and project manager. In 2013 he cofounded software development company in Belgrade where he currently holds position of director.

Received: April 29, 2013; Accepted: March 30, 2014