International Journal of Web & Semantic Technology (IJWesT) Vol.5, No.2, April 2014

# SEMANTIC WEB-BASED SOFTWARE ENGINEERING BY AUTOMATED REQUIREMENTS ONTOLOGY GENERATION IN SOA

Vahid Rastgoo[1] and Monireh-Sadat Hosseini[2] and Esmaeil Kheirkhah[3]

[1,3] Department of Computer Engineering,
Islamic Azad University-Mashhad Branch, Mashhad, Iran
[2] Department of Computer Engineering,
Lamei Gorgani Institute of Higher Education,Gorgan, Iran

### *ABSTRACT*

*This paper presents an approach for automated generation of requirements ontology using UML diagrams in service-oriented architecture (SOA). The goal of this paper is to convenience progress of software engineering processes like software design, software reuse, service discovering and etc. The proposed method is based on a four conceptual layers. The first layer includes requirements achieved by stakeholders, the second one designs service-oriented diagrams from the data in first layer and extracts XMI codes of them. The third layer includes requirement ontology and protocol ontology to describe behavior of services and relationships between them semantically. Finally the forth layer makes standard the concepts exists in ontologies of previous layer. The generated ontology exceeds absolute domain ontology because it considers the behavior of services moreover the hierarchical relationship of them. Experimental results conducted on a set of UML4Soa diagrams in different scopes demonstrate the improvement of the proposed approach from different points of view such as: completeness of requirements ontology, automatic generation and considering SOA.*

### *KEYWORDS*

*Requirements Ontology, Semantic Web; Service-Oriented Architecture; UML4Soa*

## 1. INTRODUCTION

Todays, speed and exactness in doing works and required cost and energy for doing them are so important. Accordingly, these parameters in software engineering for software generation are significant issues. Therefore, solutions that minimize service and requirement reproduction and repetitive tasks and integrate set of requirements based on a unique structure have an effective role in software generation process. Reusing of software components and services is one of common ways in software generation. Therefore, a method that being able to use in extracting software services from software archives and assists software developers to use it as a pervasive method for specification and extraction of requirements can have an operational role in software production industry[1].

Collection of proper requirements in a software project is one of the most important tasks in software generation process. Deficient requirements are one of the main reasons in failure of software projects. Therefor requirements engineering process is crucial for success of a project and must be done so carefully. With existence of heterogeneous terms and various requirements

DOI : 10.5121/ijwest.2014.5201

ontology defined in past, following cases must be considered: elicitation, analysis, specification, validation, verification and management of software requirements [2]. Scheduling and amount of attention to these activities are different in variant projects. It is studied that software projects would be vulnerable, when these activities are performed poorly.

Requirements extraction is one of the crucial steps in software developing. Ontology is a proper method for supporting this important task. Software reusing by semantic relationships and ontology is a time-consuming task and need to high precision. Therefore, automated ontology generation using a standard design format like UML diagrams can increase speed and exactness of requirements reusing dramatically. Also, this ontology can measure completeness, preciseness and unambiguous of a set of requirements. Using service specification and understanding behavior of them and retrieval related services by requirements ontology in SOA, speed, exactness and performance of this task will be increased.

## 1.1. Related Work

In recent years extensive researches have been devoted to the field of requirements engineering. In [3,4] requirements extraction and engineering of them by integrating requirements concepts is discussed. In [5] an ontology that makes standard concepts and notations of requirements modeling for perception by software engineers is made and evaluated. In [6] an ontology is generated from software artifacts which describe main concepts. In [7] an ontology is generated for software process which relationships between classes for representing dependencies is limited. The mentioned ontology is used as a meta-model. In [8] mutual convert between software artifact and domain ontology is studied. In [9] compatibility and completeness of requirements by domain ontology is investigated. Generation of domain ontology from software artifacts due to lack of sufficient exactness in finding keywords in a scope is not so useful in requirements ontology generation. In [10-13] generating domain ontology from UML diagrams is discussed. In [14-17] OWL-S language is used to describe general behavior of services and some absolute concepts are used to generate a domain ontology from one service. OWL-S language due to complexity and limitation in relationships description causes deficiency in designing complete requirements ontology.

## 1.2. Our Proposed Method

By investigating related works in this area, we believe that generating a comprehensive requirements ontology which exceeds absolute domain ontology and considers the behaviour of services moreover the hierarchical relationship of them can be more effective. One of the important issues in software engineering is reusing of predefined requirements. This issue can be a significant step in acceleration of software generation process and automation of it up to most possible limitation. Creation a requirements ontology related to a specific scope can be useful for requirements extraction in order to generate new software and also increase speed and accuracy of software services reusing.

Considering behaviour of requirements causes behaviour of each requirement being described separately and it leads to complete perception of requirements and using them in related software generation. The other case is considering relationships between services which causes related services being discovered the chain and missed requirements to be retrieved. Lack of existence a complete ontology for investigation of completeness and unambiguous of requirements and also extraction most requirements of a software process is another challenge.

By considering the mentioned issues, our goals in this paper are summarized as follows:

a) more complete and automated requirements ontology generation.
b) considering behaviour of services in order to resolve ambiguous in requirements extraction. c) considering relationships between services.
d) semantic analysis of UML diagrams using ontology.
e) generating a standard method for exchanging information between software developers.

The rest of this paper is organized as follows. Section 2 presents conceptual framework of proposed method. Section 3 describes concept extraction from XMI code achieved from UML4Soa diagrams. Section 4 explains automated ontology generation from extracted concepts. In section 5, experimental results are reported and discussed. Finally we conclude the paper in section 6.

## 2. CONCEPTUAL FRAMEWORK OF PROPOSED METHOD

The conceptual framework of our proposed method consists of four layers, so that lower layers are sources for upper layers. In other words, no layer can form without existence of its lower layers.

### A. First layer: requirements

This layer includes all requirements achieved from consumers in a specific scope and a structured document is created by a software engineer. In this document, work flow of a software system is described completely and for every set of related requirements, a service is considered which make structured relationships between those requirements.

### B. Second layer: UML4Soa

This layer includes descriptions of software requirements based on SOA. In order to being able to model all concepts in SOA using UML diagrams, we must use from UML4Soa diagrams [18]. These diagrams resolve disadvantages of activity diagrams for designing services and relationships between them. In this design method, moreover specifying behaviour of requirements and packaging them as a service, their relationships with other services are also considered that is a useful action in discovering related services [19].

### C. Third layer: requirements ontology & protocol ontology

Third layer in our proposed method consists of two sections that complement each other. These two sections are discrete from performance point of view but indiscrete from the structural perspective. Relationships between the two sections are shown in Figure 1.

- Requirements ontology
  This ontology describes semantic behaviour of every service separately. One of its more important advantages is helping to identifying services and requirements that have more behaviour similarity to the demands of engineers and designers.

- Protocol ontology
  Protocol means progress of a software process from the beginning to the end [17]. According to this statement, in SOA a protocol shows relationships between services

from the beginning to the end of a service-oriented software process. Protocol ontology can have a significant role in discovering related services and services that are missed or not discovered.

### D. Forth layer: meta-concepts

In our proposed method, key concepts used in SOA and activity diagrams are considered as meta-concepts. These key concepts finally cause extracted services and requirements to be investigated from compatibility, unambiguous and correctness points of view. In fact, forth layer makes possible sharing of generated ontologies between software developers moreover their creators.

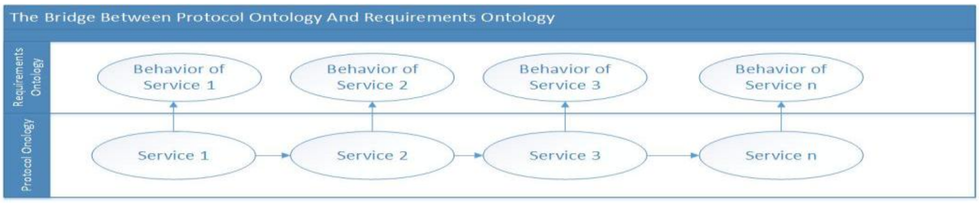


Figure 1. Relationships between requirements ontology & protocol ontology

## 3. CONCEPT EXTRACTION FROM XMI CODE

The overall framework of the proposed method is shown in Figure 2. In order to extract concepts from XMI codes, we first design UML4Soa diagrams in MagicDraw UML software and extract XMI codes through it. Figure 3 shows a part of extracted XMI code. In order to generate a meaningful and practical ontology from XMI code, we must first determine its fundamental concepts because the main base of an accurate ontology generation is correct identification of concepts and relationships between them. In each ontology four main concepts exist that are class, object property, data property and entity. Table 1 shows the keywords categorization of extracted XMI code based on these four concepts.

The concepts that are considered in this table will be used as meta-concept in our proposed method. We divide each of these concepts to some groups, so that each group is subtype for these meta-concepts.

Table 1.Correspondence between diagram concepts, descriptive and semantic web.

| UML | XMI | OWL |
|---|---|---|
| Service Activity | uml:StructuredActivityNode | Class |
| Send-Send&Receive | uml:CallOperationAction | Class |
| Receive | uml:AcceptCallAction | Class |
| Link Pin | Target | Entity |
| Link Pin | Result | Entity |
| Initial Node | uml:InitialNode | Entity |
| Activity Final | uml:ActivityFinalNode | Entity |
| Control Flow | uml:ControlFlow | Object Property |
| Compensation Edge | uml:Compensation | Object Property |
| Event Edge | uml:Event | Object Property |
| Decision/Merge | uml:DecisionNode | Object Property |
| Send Pin | Argument | Data Property |
| Receive Pin | Return Information | Data Property |
| Data | uml:OpaqueAction | Data Property |

In follow, we will identify subtypes of meta-concepts using proposed correspondence table.

- Identification of classes
  We consider two types of classes for requirements ontology include service and activity classes. For example, uml:StructuredActivityNode is a characteristic for a service and the name of that service which exists in the same row places as subclass from service class. This flow for activity class is the same as service class.

- Identification of entities
  Two groups of entities are considered, entities that are introduced as the beginning and the end of activities and entities that have connection with services through link and called foreign entities.

- Identification of object properties
  There are two general types of object properties. The first type that is recognizable directly from XMI codes, like Control Flow, Compensation and Event and the second type that does not exist directly in XMI code and must be concluded from it, like Send, Receive and Start. Domain and range of the first type are recognizable trough Source and Target. Domain of the second type is an activity and the range is an entity that is connected to it.

- Identification of data properties
  Every row with characteristic of Argument or Return Information is recognized as data that must be sent or received and place in data properties group with the name that exists in the same row.

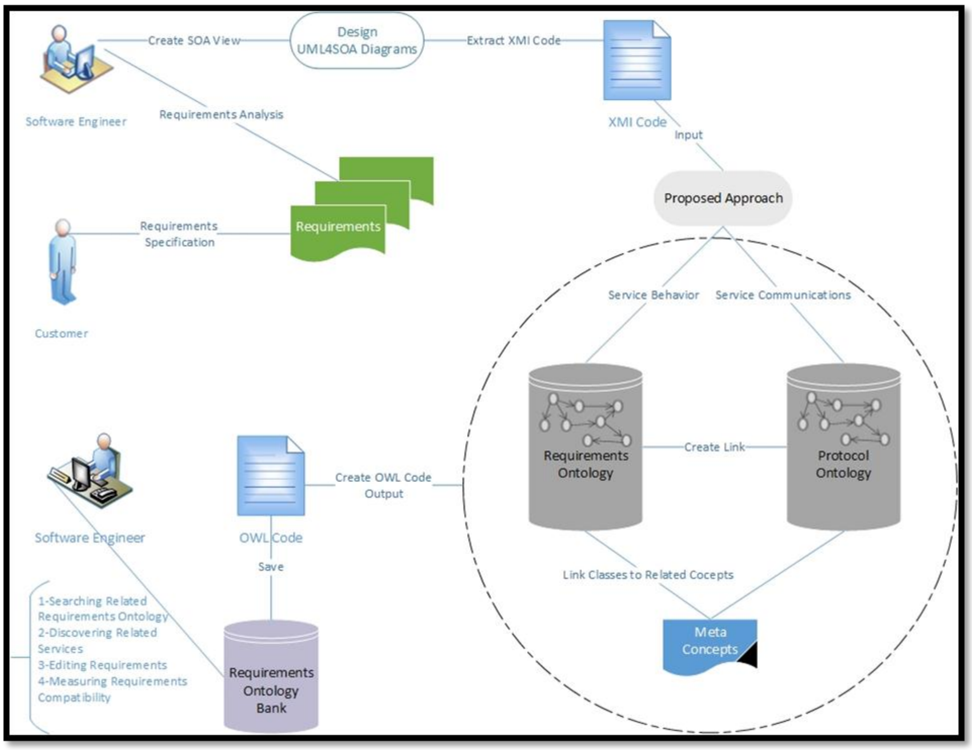


Figure 2. The overall framework of the proposed method

```
<node xmi:type="uml:AcceptCallAction"
xmi:id="_16_8_8ff0292_1373466685155_242944_1925"
name="acceptTopic" visibility="public"
outgoing="_16_8_8ff0292_1373466933585_699102_2009">

<result        xmi:id="_16_8_8ff0292_1383639752786_519586_2209"
name="student" visibility="public"/>

< returnInformation
xmi:id="_16_8_8ff0292_1383639790976_982383_2218"
name="thesisId" visibility="public"/>
```

Figure 3. A part of XMI code extracted from MagicDraw UML software

## 4. AUTOMATED ONTOLOGY GENERATION FROM EXTRACTED CONCEPTS

In this section, after categorization of concepts and required information, required ontology will be produced in OWL code format. The format of produced code can be in OWL/XML and RDF/XML. Both of these two formats are achieved from Protégé software. In Figure 4, a part of generated code in RDF/XML format is illustrated. In fact, main classes, sub classes, object properties following by their domain and rang, data properties and also limitations will be published in OWL code framework.

```
<owl:ObjectProperty rdf:about="#CF4">
   <rdfs:subPropertyOf rdf:resource="#ControlFlow"/>
        <rdfs:range>
            <owl: rdf:resource="#removeFromBoard"/>
        </rdfs:range>
        <rdfs:domain>
             <owl:rdf:resource="#acceptTopic"/>
        </rdfs:domain>
</owl:ObjectProperty>

<owl:DataProperty rdf:about="#thesisId">
</owl:DatatypeProperty>

<owl:Class rdf:about="#Registration">
         <rdfs:subClassOf rdf:resource="#Service"/>
</owl:Class>
```

## 5. EXPERIMENTAL RESULTS

In this section, we will introduce the test dataset and evaluation measures and evaluate our proposed method from different points of view.

### 5.1. Dataset

In this paper, four set of diagrams in different scopes are used for evaluation the proposed method. These sets are completely pervasive and could be used as a benchmark for representing the performance of our proposed method. Details of the datasets used in test process are presented in Table 2.

Table 2. Datasets for testing the proposed method

| Name | Scope | Creator | Year |
|---|---|---|---|
| Thesis Manager | University | Philip Mayer | 2010 |
| Agency Service | Travel Agency | Federico Banti | 2011 |
| Automotive Scenario | Car Manager | Rosario Pugliese | 2011 |
| Service InfoUpdate | Update Information | Francesco Tiezzi | 2011 |

### 5.2. Evaluation measures

After selection a proper benchmark for test, evaluation of methods must be done using efficient measures. In this paper, we use three measures to evaluate our proposed method from three points of view: requirements ontology creation, generation of ontology automatically and exactness and completeness of the ontology.

#### 5.2.1. First measure: the number of concepts covered by requirements ontology

By this measure, we can determine that how much our requirements ontology is complete than other ones. In Table 3, concepts that are covered by requirements ontology in our proposed method are compared with [1]. As the table shows, in our method, in addition to concepts that are considered in previous works, services and compensation of services are considered too. This measure shows that our method can cover services and SOA moreover consideration requirements based on activities. Also in [1] relationships between concepts are considered just as an edge, while in our method each relationship includes type, name, domain and range. Therefore, the generated ontology in this paper exceeds domain ontology.

Table 3. Comparison between proposed method and [1]
in terms of covered concepts by requirements ontology.

| No. | Our Ontology | Nisreen | Equivalent |
|---|---|---|---|
| 1 | Class | Class | Yes |
| 2 | Object Property | Relationship | Yes |
| 3 | Activity | Activity | Yes |
| 4 | Service | - | No |
| 5 | Entity | Entity | Yes |
| 6 | Data Property | Data | Yes |
| 7 | Compensation | - | No |

### 5.2.2. Second measure: comparison between manually ontology generation and automatic ontology generation

This measure shows difference between the number of discovered concepts by an expert human and the number of discovered concepts automatically. Figures 5-8 shows comparison diagrams between manually and automatic requirements ontology generation in different scopes. As the figures shows, in all cases the number of discovered concepts in our automatic proposed method is very close to the manual generation in protégé software.

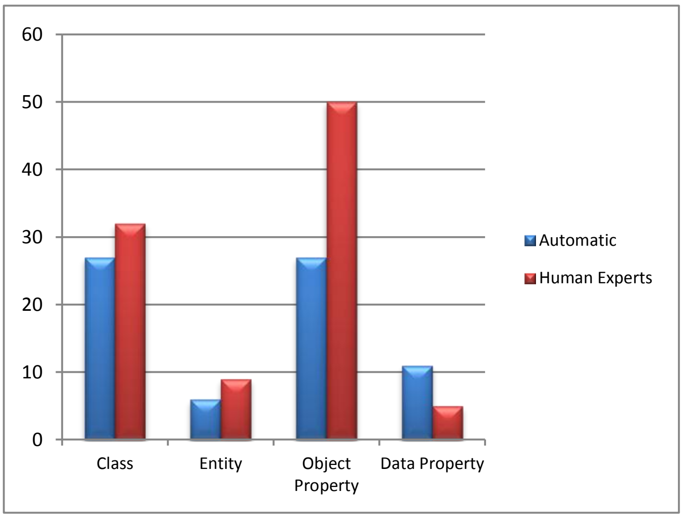


Figure 5. Comparison between manual and automatic requirements ontology generation in Thesis Manager scope.

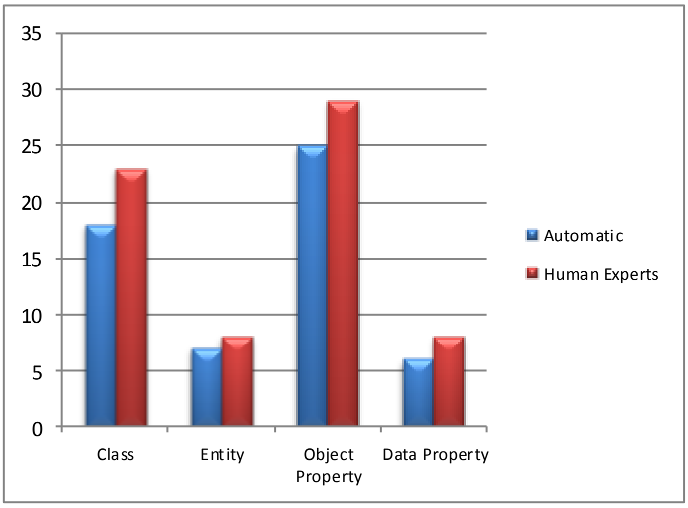


Figure 6. Comparison between manual and automatic requirements ontology generation in Agency Service scope.

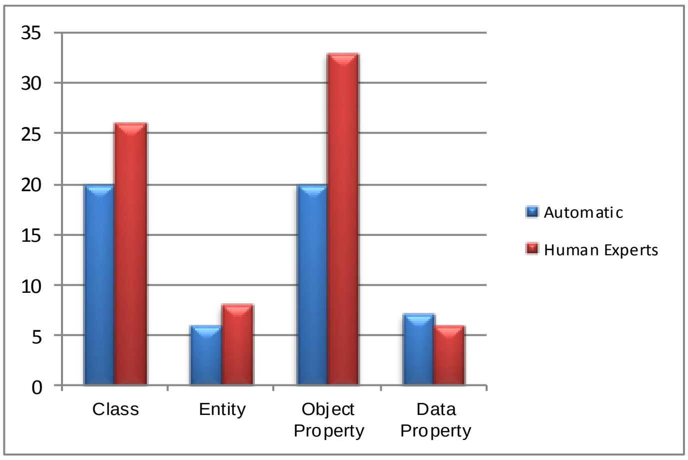


Figure 7. Comparison between manual and automatic requirements ontology generation in Automotive Scenario scope.

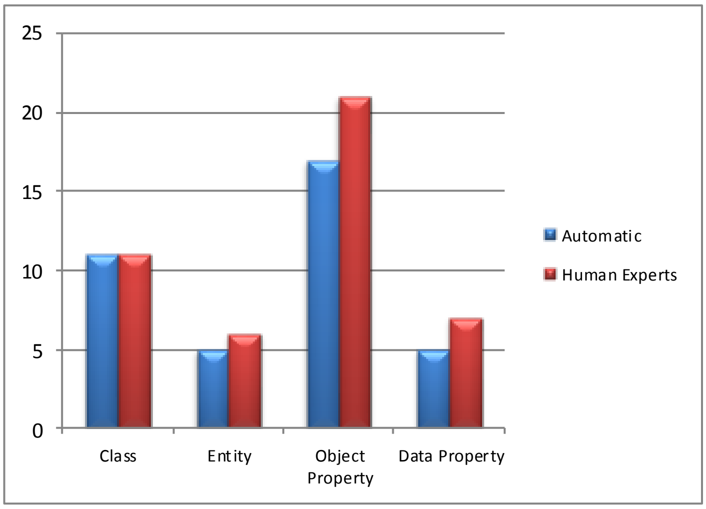


Figure 8. Comparison between manual and automatic requirements ontology generation in Service UpdateInfo scope.

### 5.2.3. Third measure: precision, recall and F-measure

Moreover, the above measures, we use other measures to show the effectiveness of our proposed method. These measures include precision, recall and F-measure that are defined below respectively. Table 4 shows the obtained values according to these measures.

Precision= Correct Extracted Concepts / Total Extracted Concept (1)

Recall=Correct Extracted Concepts / Correct Extracted Concepts + Missing Concepts (2)

F-measure=2 * Precision * Recall / Precision + Recall (3)

Table 4. Performance of the proposed method in terms of precision, recall and f-measure

| Scope | Precision | Recall | F-measure |
|---|---|---|---|
| Thesis Manager | 0.59 | 0.73 | 0.65 |
| Agency Service | 0.61 | 0.82 | 0.69 |
| Automotive Scenario | 0.60 | 0.72 | 0.65 |
| ServiceUpdateInfo | 0.64 | 0.84 | 0.72 |

## 6. CONCLUSION AND FUTURE WORK

In this paper, a method for automated requirements ontology generation using UML diagrams in SOA is proposed. In the proposed method, the extracted codes from UML4Soa diagrams using proposed correspondence table and general format of ontology generation convert to OWL codes. In fact, three main level of ontology existed in our method which considered behaviour of services, relationships between services and also cover standard description related to requirements engineering scope. Using this ontology helps to discovering of required services, related services and missing services and also reusing of software components. The proposed method can be used with just a little adjustment to other fields in engineering designs. Also, this method can extend to other design diagrams such as sequence diagrams and activity diagrams.

## REFERENCES

[1] Vangipuram Radhakrishna , C.Srinivas, (2013)"Document Clustering Using Hibrid XOR Similarity Function For Efficient Software Component Reuse", *Procedia Computer Science*, Vol.17,pp.121–128.

[2] Ricardo de Almeida Falbo , Julio Cesar Nardi, (2011)" Evolving a Software Requirements Ontology".

[3] Ge Li , Zhi Jin & Yan Xu & Yangyang Lu, (2011)" An Engineerable Ontology Based Approach for Requirements Elicitation in Process centered Problem Domain", *Springer*, pp.208–220.

[4] Bertrand Verlaine, Ivan J.Jureta, Stéphane Faulkner, (2011)" Requirements Engineering for Services:An Ontological Framework", *ACM*, pp.21-25.

[5] Nisreen Innab , Ahmad Kayed & A.S.M.Sajeev, (2012)"An Ontology for Software Requirements Modelling", *Information Science and Technology.*

[6] Dang Viet Dzung , Atsushi Ohnishi, (2009)"Improvement of Quality of Software Requirements with Requirements Ontology", *Quality Software*.

[7] Ricardo de Almeida Falbo , Julio Cesar Nardi, (2011)" Evolving a Software Requirements Ontology".

[8] Li Shunxin, (2010) "Requirements Engineering Based on Domain Ontology", *Information Science and Management Engineering*.

[9] Haruhiko Kaiya, Motoshi Saeki, (2011)" Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach*"*, *QSIC*, pp.223-230.

[10] Dragan Gaševic , Dragan Djuric, (2009)"Mappings of MDA-Based Languages and Ontologies", Ontology Development, *springer*, pp. 245-261.

[11] Dragan Gaševi´c , Dragan Djuri´c & Vladan Devedži´c, (2007)" MDA-based automatic OWL ontology development", *Int J Softw Tools Technol Transfer* .

[12] Dragan Gašević , Dragan Djurić & Vladan Devedžić & Violeta Damjanović, (2004) "Converting UML to OWL Ontologies", *ACM*.

[13] Hong-Seok Na , O-Hoon Choi & Jung-Eun Lim, (2007)" A Method for Building Domain Ontologies based on the Transformation of UML Models", *Software Engineering Research,Management and Applications*.

[14] B.Verlaine , Y.Dubois, (2011)" Towards conceptual foundations for service-oriented requirements engineering: bridging requirements and services ontologies", *IET Software*.

[15] Il-Woong Kim , Kyong-Ho Lee, (2007)" Describing Semantic Web Services: From UML to OWL-S", *ICWS*.

[16] G. Meditskos , N. Bassiliades, (2011)" A combinatory framework of Web 2.0 mashup tools, OWL-S and UDDI", *Expert Systems with Applications* ,Vol.38, Issue 6.

[17] Matthias Klusch , Benedikt Fries & Katia Sycara, (2009)" OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services",*Web Semantics: Science, Services and Agents on the World Wide Web,* Vol.7, Issue 2.

[18] Federico Banti , Rosario Pugliese & Francesco Tiezzi, (2011)" An accessible verification environment for UML models of services", *Journal of Symbolic Computation 46* , pp.119–149.

[19] Yong Tian , Mutao Huang, (2012)" Enhance discovery and retrieval of geospatial data using SOA and Semantic Web technologies", *Expert Systems with Applications*, pp.12522–12535.

**Authors**

**Vahid Rastgoo** was born in Gorgan, Iran in 1986. He received BSc. degree in Computer Software Engineering from Payame Noor University, Behshahr, Iran, in 2009 and M.Sc degree in Software Engineering from Islamic Azad University, Mashhad branch, Mashhad, Iran in 2014. His research interests include Software Development Processes, Requirements Engineering , Semantic Web enabled Software Engineering and Software Reuse.

**Monireh-Sadat Hosseini** was born in Gorgan, Iran in 1986. She received BSc. degree in Computer Software Engineering from Payame Noor University, Behshahr, Iran, in 2009 and M.Sc. degree in Artificial Intelligence from Islamic Azad University, Qazvin branch, Qazvin, Iran in 2012. Her research interests include Soccer Video Analysis, Video Compression, Multimedia Information Retrieval and Semantic Web enabled Software Engineering.

**Esmaeil Kheirkhah** was born in Tabriz, Iran in 1969. He received BSc. degree in Mathematics Applied in Computer and M.Sc. degree in Mathematics from Islamic Azad University, Mashhad branch, Mashhad, Iran in 1992 and 1996 respectively and Ph.D. degree in Computer Science (Software Engineering) from National University of Malaysia (UKM) in 2010. In 1997 he was a lecturer of computer engineering at the Islamic Azad University, Mashhad branch, Iran. He is Manager of the Information and Communication Technology Center (ICTC) in Islamic Azad University, Mashhad Branch from 1997 to 2006 and 2011 to present. His research interests include Software Development Processes, Requirements Engineering , Semantic Web enabled Software Engineering and Knowledge-Based Systems.