INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 13(1), 117-128, February 2018.

Modern Interfaces for Knowledge Representation and Processing Systems Based on Markup Technologies

A. A. Mohammed Saeed, D. Dănciulescu

Ali Amer Mohammed Saeed*
Doctoral School of Informatics
University of Piteşti
110040 Piteşti, 1 Târgul din Vale Str., Romania
*Corresponding author: ali.amer81@gmail.com

Daniela Dănciulescu
Computer Science Department
University of Craiova
200585 Craiova, 13 A. I. Cuza Str., Romania
danadanciulescu@gmail.com

Abstract: The usage of markup technologies to specify knowledge to be processed according to a specific field of application is a common technique. Representation techniques based on markup language paradigm to describe various types of knowledge including graph based models is considered and details on using Knowledge Representation and Processing (KRP) Systems in education are presented. XML, and VoiceXML were selected to implement smart interface for KRP systems.
Keywords: KRP systems, markup technologies, intelligent interfaces, VXML

1 Introduction

This article deals with Markup mechanisms for knowledge, but also for voice interfaces. It is based on [13] being an extended version of the previous work of the first author.

The coverage of the subject follows. The next section deals with the usability and efficiency of the following approaches to be used in KRP context: SGML / XML, RDF extensions, state-based modeling - SCXML, and Voice XML.

From the RDF (Resource Description Framework) category, in the context of KRP systems, CKML (Conceptual Knowledge Markup Language), Ontology Markup Language (OML) and DLML (Descriptive Logic Markup Language) are useful.

Other approaches are based on the Ontology Interface Layer (OIL) and the DARPA Agent Markup Language (DAML). Of the ontological development tools, the most commonly used are: DUET (UML Enhanced Tool), UBOT, Protege, and Ontolingua. An example of processing using descriptions in natural language is illustrated using SCXML.

SCXML and VoiceXML are covered in the third section.

Interaction of knowledge bases using JAVA technologies is demonstrated in the fourth section. For this purpose, the legacy knowledge model is modeled by a graph that indicates the inheritance relationship of object attributes.

The fifth section is dedicated to the usage of KRP systems in education. It is shown that, for visually impaired users, the usage of VoiceXML based technologies to translate various educational resources is feasible.

2 Markup models and knowledge representation

By models and markup technologies, in the context of this paper, we understand such models and technologies obtained from SGML (Standard Generalized Markup Language; ISO 8879:1986

[28]). SGML is a meta-language, i.e. an artificial language which allow us to describe other languages, in general for the formatting of documents [13].

SGML was used initial by the Association of American Publishers. Then it has become a powerful model with applications and multiple influences. For example, Coleman and Willis (1997) proposed the usage of SGML in the conservation of the publications of the libraries [4]. In the same year, already appeared HTML (HyperText Markup Language, 1990 [23]) useful for WWW, and Extensible Markup Language (XML, 1996) as the language of the description of the structured information [31]. Therefore, SGML is known as being the father of both HTML and XML [13]. However HTML is a court specifies its DTD of SGML (with markers predefined), and XML is a subset of SGML where users can define their own tags and attributes.

An XML document is composed of *markers* (tags) and *data* "character" (char, character). A *marking* is a string of characters bounded by the symbols "<" and ">". An XML file contains three sections: a *header* (<?xml version="1.0" encoding="UTF-8?">), the *definition* of document type internal or external (example: <!DOCTYPE document SYSTEM "location of its DTD">) and the *root* (XML Information in this part may be set as a tree structure).

A XML schema to describe the set of rules used by Knowledge Representation and Processing (KRP) Systems can be given as below (regula.xsd).

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="KRPRule">
  <xs:complexType>
   <xs:sequence>
     <xs:element name="RLabel" type="xs:string"/>
     <xs:element name="RLeft" type="xs:string"/>
     <xs:element name="RRight" type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

XML Processors are used to verify whether the XML documents are well formed or not. To access and editing an XML document, initially is loading the XML document in associated task (example with JavaScript) [13]:

```
parser = new DOMParser();
xmlDoc = parser.parseFromString(text,"text/xml");
```

Then extract the elements of the XML document for processing.

In the context of the knowledge representation of the rules used by KRP systems we can consider <KR> as a root element which may include one or more elements of the rule type. Each item rule has a unique identifier *rid*. A rule of association (A $\rightarrow$ B) is formed of "The hypothesis of A" and "B - the consequent part". Therefore, each hypothesis must have one or more items. Each item hypothesis has a name that is represented by a sequence. This model is described below.

XML document:

```
<KR><Rule rid="1">
      <Hyp>
          <ItemHyp>
              <Name>A</Name>
          </itemHyp>
      </Hyp>
      <Infer>
          <ItemInfer>
              <Name>B</Name>
          </ItemInfer>
       </Infer>
</Rule> </KR>
```

The DTD of the model:

```
<!ELEMENT KR (Rule+)>
<!ELEMENT Rule (Hyp Infer)>
<!ATTLIST Rule rid CDATA>
<!ELEMENT Hyp (ItemHyp+)>
<!ELEMENT ItemHyp (Name)>
<!ELEMENT Infer (ItemInfer+)>
<!ELEMENT ItemInfer (Name)>
<!ELEMENT Name (#PCDATA)>
```

To describe structures useful to outline knowledge in the field of science of the soil for agriculture, the authors of [12] have converted XML declarations in a format useful for application, called KBML (Knowledge Based Markup Language). All meta-information is stored in a file KBML, while the actual data may be available in any data source (distributed, etc.). According to [13], KBML is not a markup language, but merely an application of XML.

In the context of modeling and knowledge processing many specialized Markup notations have been developed, such as: RDF/XML (model supertitles for expressing graphs as RDF documents that XML [27]), CKML (The Conceptual Knowledge Markup Language, 2000 [10]), OML (Ontology Markup Language [26]), DLML (Logical Description Markup Language [22]). OML is an extension of the SHOE and supports the lambda expressions. OML and CKML are based on the conceptual graphs introduced by Sowa (2008) in [17].

Querying RDF data is possible by specific languages, some in the lines of traditional database query languages, others based on logic and rule language [1]. Stratified graphs can be used to automatic generation of queries in formal or natural language [5, 6, 14].

The kernel of a RDF model is made up of *nodes* and *pairs of attached attributes/values*. A description of the RDF syntax is presented in [3] and can be understood on the basis of the following example that describes the creator of the file tey.rdf located in a folder on a Windows Server:

```
<rdf:RDF>
  <rdf:Description
    rdf:about="file:///home/ali/tey.rdf">
     <xf:Creator>
       <rdf:Description
          rdf:about="http://www.upit.ro/">
           <xf:Name>A L I</xf:Name>
       </rdf:Description>
```

```
      </xf:Creator>
  </rdf:Description>
</rdf:RDF>
```

For RDF diagrams one shall specify the space of rdfs *names*. The fundamental RDF *classes* are: rdfs:Resource (class resources), rdf:Property (describes the properties of the resources) and rdfs:Class (for specifying the type or category). To define a new class of RDF diagram, the corresponding resource class has the property rdfs:type whose value resource is rdfs:Class. The resources which belong to the defined class are called *courts*. An example that describes a collection of resources is:

```
<rdf:Bag rdf:ID="docs-apply">
  <rdf:li
      rdf:resource="file:///lucru/teza.docx" />
  <rdf:li
      rdf:resource="https://www.upit.ro/\_doc/8806/a_27_c_taxe.pdf" />
  <rdf:li
     rdf:resource=" https://www.upit.ro/\_doc/11836/proc_mencs.pdf" />
</rdf:Bag>
```

New versions CKML have included the ideas and techniques on the informational flow (IF - *Information Flow*) and the design of the logic of the distributed systems. The final version CKML is both a language based on the logic of the information document and a language based on frames. In accordance with Kent(2000), "in CKML the specification requires the use of the concept of mathematical lattice or the most practical notion of conceptual space" [10]. The basis of the theoretical portion of the practice based on CKML is the CKP Theorem which states *the equivalence between data structures of type conceptual lattice and formal context (classification).*

OML provides three levels of further specify the restrictions [26]: top - sequences (corresponding informational flow); the intermediate pipe - calculation of binary relations; Lower logical expressions (corresponding to concept graphs).

Expressing an ontology is possible using the languages of specification such as [13]: KIF (Knowledge Interchange Format), CL (Common Logic), OIL, DAML+OIL AND ALLURE.

KIF is based on the logic of the predicates [25], but provides a LISP oriented syntax for this. From the point of view of the semantic, there are four categories of constant in KIF: constant of type object, constant of function type, constant of relation and logical constant.

OIL (Ontology Inference Layer [7]) extends RDF diagram to provide an intuitive syntax and a great power of expression and a semantics more clearly defined with easy to use descriptive logic within the framework of the schemes of reasoning. Such OIL brings together and unifies three directions: descriptive logic, modeling based on frames and modeling RDF/XML.

(DAML DARPA Agent Markup Language) + OIL has a syntax diagram type RDF, that inherits the primitives of RDF (subclass, domain, range) and primitive added extras like transitivity, cardinality etc. Schematic DAML+OIL is oriented on the objects in which the concepts are abstracted by grades and roles through the properties of the objects. Thus, the ontological model DAML+OIL is based on a lot of the axioms about the classes and properties, as well as a set of builders very useful from the perspective of the RPC systems [13]: intersectionOf; unionOf; complementOf; oneOf; toClass; hasClass; hasValue; minCardinalityQ; maxCardinalityQ; cardinalityQ.

The result of the foregoing the evolutionary process is [13]: 1) OIL extends RDF; 2) DAML extend RDF; 3. DAML+OIL DAML integrates and OIL and extends the RDF; 4) ALLURE extends DAML+OIL and RDF.

The final result of the research on ontological modeling using RDF/XML has led to the specification of the ALLURE, in three versions [13]: ALLURE LITE (simple hierarchy, hierarchy of classes with simple constraints), ALLURE DL (maximum expressiveness) and ALLURE FULL (very expressive). For the processing of meta-data described using specific Markup ontologies have been developed a variety of tools for annotation, navigation, utilities (API), edit, view graphics, marking, pan, validation, import, export, compilation, query, search etc. A list of them would be too long. We will be limited to the most important tools, the rest being described in the references indicated: DUET (DAML UML Enhanced Tool), UBOT, The Platform Protégé, and Ontolingua. Ontolingua Editor allows for the creation of ontologies, exploration and collaborative editing. Using Ontolingua, it is possible to export and import formats like: KIF, DAML + OIL, OKBC, Prologue, the LOOM, Ontolingua and CLIP. Can import data in the *protégé* format.

3 SCXML and Voice XML

SCXML provides a generic state-machine, an execution environment based on CCXML and Harel State Tables, according to W3C(2015) in [30]. Also in [11] it is mentioned that: "using SCXML as the representation of the state machine is seen as a benefit". The mentioned authors found that "large portions of the SCXML standard are not necessary for it to be useful to our customers and us." CCXML is designed to upgrade VXML dialog systems with advanced telephony functions. An example of the SCXML representation is for speech recognition in the natural language. For the implementation of the KRP systems, the role of the SCXML is active in the framework of the failures, through voice and natural language.

According to the above considerations, it was our choice to propose the usage of VXML to create voice-enabled applications [29]. VoiceXML (VXML) is a markup language for specifying the vocal dialog between a man and a software application, for example a KRP system. Thus, using VoiceXML 2.0 one can develop KRP applications which provides automatic recognition of speech (ASR - Automated Speech Recognition) and interactive vocal response (IVR - Interactive Voice Response).

The main elements of voiceXML are:

- <vxml> - start/close any vxml document;
- <var>, <assign>, <clear> used to declare, assign and delete variables;
- <grammar> to specify the grammar of the text under recognition;
- <catch>, <throw>, <error>, <noinput>, <nomatch> to manage exceptions;
- <menu>, <choice>, <enumerate> to deal with menu;
- <if>, <else>, <elseif> to describe conditional aspects;
- <initial>, <form>, <field>, <filled>, <option> to process forms;
- <prompt>, <reprompt>, <value> for input operations;
- <prompt>, <audio>, <record>, <reprompt> to deal with multimedia entities;
- <block> to describe the code to be executed;
- <disconnect>, <exit> for the management of the sessions;
- <meta>, <metadata> for metadata management;

- <noinput>, <nomatch>, <help> to manage events and actions;
- <subdialog>, <goto>, <return>, <link> for dialog control;
- <object>, <property>, <param>, <script>, <submit>, <transfer>, <log> for server oriented processing of parameterized queries.

Vxml applications may be of the type uni - or many - document. An application many - document allows us to define a root document which defines all the entities visible in and recovered by the documents son. VXML applications are oriented to the following categories:

- *Queries* - to retrieve information from Web-based infrastructures (like voice portals, web call centers);
- *Transactions* - to execute specific transactions with a Web-based back-end database.

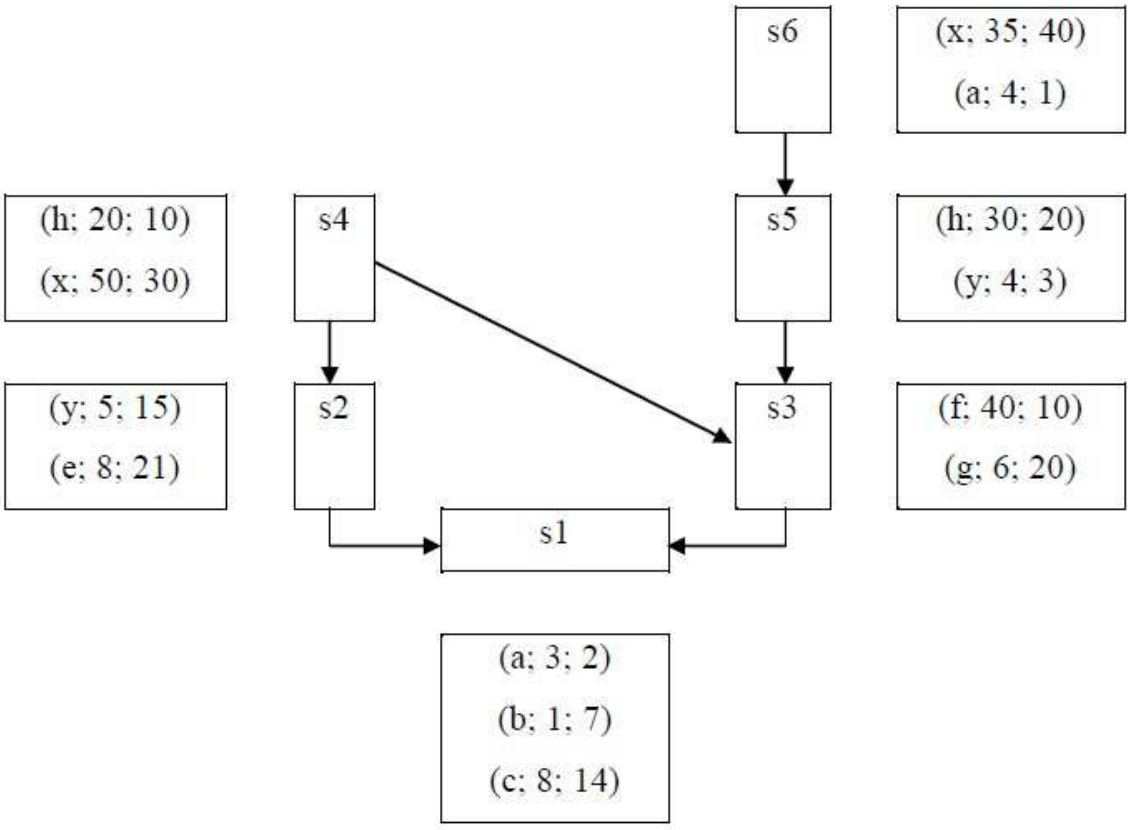

Figure 1: A Knowledge database - inheritance and its graph representation

To optimize the development of voice based user interfaces, the following facts should be understood:

1. A VXML application is a single VoiceXML document, or a set of documents which forms a *conversational finite state machine* (CFSM). The root document is loaded firstly and remains loaded while transitioning over documents belonging to the same application.
2. A *session* is opened by the user to start the interaction with the VXML interpreter, and is closed by a request from the user, a document, or the interpreter itself.

3. VXML has two types of dialogs: *forms* and *menus*. Each dialog has associated one or more speech and/or DTMF (Dual-Tone Multi-Frequency) grammars.
4. A *form* defines an interaction that obtains values for a set of variables.
5. A *menu* provides the user with some alternative options and follows other dialogs depending on the selection.
6. A *subdialog* is like a function call; after interaction returns to the original form.
7. There exist two types of grammars in VXML: machine directed (the form items are executed in the sequential order) and mixed initiative (the flow has to be directed both by the user and by the application).

The example described below is a skeleton query type using the voice-based interface when querying inheritance graphs of knowledge is considered, as [18, 19], proposed for text interfaces. This model can be extended for the implementation of the interfaces based on voice within web-based KRP systems.

To demonstrate the basic principles of voice-based interfaces, a simple knowledge database (described in Fig. 1) is considered. Based on a client-server implementation in Java, a dialog for the computation of the certitude factor is shown:

```
private Object[][] b3 = {
{"s1", "s2 33","atr(a,3,2) atr(b,1,7) atr(c,8,14)"},
{"s2", "s4","atr(y,5,15) atr(e,8,21)"},
{"s3","s4 s5","atr(f,40,10) atr(g,6,20)"},
{"s4","","atr(h,20,10) atr(x,50,30)"},
{"s5","s6","atr(h,30,20) atr(y,4,3)"},
{"s6","","atr(x,35,40) atr(a,4,1)"}
};

Dialogue: The first step for client1
>>> Welcome. Ready for you!
>>> Select your knowledge base: b3
---  The knowledge base b3 was selected for processing.
>>> Select the object to investigate: s1
--- The object s1 was selected.
>>> Select the attribute under investigation: h
--- Your choice was the attribute h.
Answer: The value of attribute h of object s1\\
 is 30 and the certitude factor is fc=20.

VXML code:
<prompt>Welcome. Ready for you!</prompt>
<prompt count="1">
Select your knowledge base: <value expr="KnowledgeBase"/>?
</prompt>
// Javascript Code
<prompt> The knowledge base </prompt>
// Code to print --KnowledgeBase--
<prompt> was selected for processing. </promt>
<prompt count="2">
```

```
Select the object to investigate: <value expr="Object"/>?
</prompt>
// Javascript Code
<prompt> The object </prompt>
// Code to print -- Object Name--
<prompt> was selected. </promt>
<prompt count="3">
Select the attribute under investigation: <value expr="Attribute"/>?
</prompt>
// Javascript Code
<prompt> Your choice was the attribute </prompt>
// Code to print --Attribute--
<prompt>. </promt>
```

4 Markup technologies

From the point of view of the process of XML tying - JAVA implemented by JAXB, it is noticed the existence of two major components [24]: a generator of diagrams and compiler of diagrams and the process actually involves tying seven actions: the generation of classes; the compilation of classes; unmarshal (XML documents which satisfy the restrictions in the diagram source are processed by the JAXB. Also, JAXB lets you transfer XML data from sources other than the files and XML documents such as the nodes DOM, paintings rows of characters, SAX sources and so on and so forth); the generation of the shaft into which describes the contents of an XML document; validation (Unmarshal process involves the validation of the source before generating shaft into which describes the contents. Where there is a change in the shaft in the next step can be used the operation JAXB validation to confirm the changes before to transform the contents into a document XML); the client application may change the XML data represented by JAXB shaft using the interfaces generated by the compiler JAXB; marshal (the shaft that describes the contents is converted into the XML document. The content can be validated before the conversion. A process called "Marshalling" offers a client applications the ability to convert a Java derived from JAXB in data XML.)

With a force greater than the programming, can be used JAXP technology (Java API for Processing XML) based on SAX (Simple API for Parsing XML) and DOM (Document Object Model). During the operation of the "parsing" based on SAX it generates events that notify the components identified, and Java application must deal with the events of the callback methods (for the construction of the structure of the database). The operation of parsing DOM build in the memory a representation tree diagram of the data from the XML document. JAXP technology allows the transformation of XML documents using XSLT technology (Extensible Stylesheet Language Transformation).

XMLBeans technology is used to compile the XML layout with obtaining in memory, the classes, and has been developed in the period 2003-2014 by the Apache Software Foundation to enable the processing of large structures.

Therefore, for the processing of databases structured knowledge which comply with a diagram and are stored in XML files may be used JAXB technologies facing on the diagram XML and JAXP facing on the direct rendering of documents XML. JAXP is a good choice for large knowledge database to be processed in terms of low computational capacity.

The following example shows how a XSLT stylesheet is used to transform a sample data set into VoiceXML 2.0 format.

```
<?xml version="1.0"?>
<phdstud>
   <stud>
      <pid>06</pid>
      <uni>University of Pitesti</uni>
      <phds>Ali Amer Mohameed Saeed</phds>
      <year>2017</year>
   </stud>
   <stud>
      <pid>107</pid>
      <uni>University of Bucharest</uni>
      <phds>Radu Mihai</phds>
      <year>2017</year>
   </stud>
</phdstud>

<vxml version="2.0">
<form id="start">
 <audio>Some PhD students </audio>
 <xsl:for-each select="phd">
       <audio>PhD student id is <xsl:value-of select="pid" /></audio>
         <break time="100ms"/>
         <audio>Comes from PhDsch.<xsl:value-of select="uni"/></audio>
         <break time="100ms"/>
         <audio>The PhD name is <xsl:value-of select="phds" /></audio>
         <break time="100ms"/>
     <audio>Year of defence is <xsl:value-of select="year"/></audio>
         <break time="100ms"/>
  </xsl:for-each>
  </form>
</vxml>
```

5 KRP systems in education

According to [16], a KRP system for Artificial Education (AE) should take into consideration four elements. In AE, the first element, "knowledge would include knowledge of pedagogy (teaching practices and beliefs), curriculum, and knowledge regarding the individual student's needs, assessments, evaluating, and more". The second element is connected to problem solving. In this context, the KRP system should "look at past successful and unsuccessful pedagogies used with individual student, and it would be able to present instructional material to that specific student in a way the benefited him or her individually". The last two elements are connected to developers and administrators, but the mentioned authors did not conclude on smart interfaces for KRP educational systems. However, they emphasize on Intelligent Tutoring Systems (ITS), but ITS are "emphasizing those aspects which have relevance to user support, rather than detailed consideration of the merits of pedagogical or student knowledge modelling strategies" as shown by Hefley & Murray (1993) in [8].

Following Horvitz(1999), an intelligent user interface should consider imprecision and uncertainty aspects during run-time [9]. This is more important in AE, due to the nature of queries

formulated by learners. As Salih(2014) mentioned [15], the Natural User Interfaces (NUI) will be the next generation of user interfaces to improve user experiences. Our proposal is based both on Artificial Intelligence Techniques to deal with imprecision/uncertainty and natural language aspects with speech understanding and knowledge restructuring for fast answering systems.

Therefore, any KRP system for education should consider preliminary requirements to understand the learner's behaviour, markup models and technologies to implement solutions to queries given in "approximate" natural language by learners. One KRP system for education should be able to represent not only pedagogical aspects, but also, different variants of content, and appropriate behaviour according to the learn initiatives.

Smart interfaces of KRP systems for education are based on Voice-enabled applications to support e-learning in many ways, making possible the usage of e-learning systems by visually impaired users. In Web-based e-Learning systems, the output is generated in HTML format. In order to support Voice type output, one step more is required to translate HTML to VXML. In the following, only a short example for translating a table is given. Only the VXML code is shown.

```
<?xml version=''1.0''?>
<vxml version=''2.0''>
..
<block> The next structure is table 1 </block>
<block> The table name is </block>
// Code ...
<block> Row 1 </block>
<block> Column 1 </block>
<block> E[1][1] </block>
<block> Column 2 </block>
<block> E[1][2] </block>
<block> Row Ending 1 </block>
<block> Row 2 </block>
<block> Column 1 </block>
<block> E[2][1] </block>
<block> Column 2 </block>
<block> E[2][2] </block>
<block> Row Ending 2 </block>
<block> This the ending of table 1 </block>
</vxml>
```

6 Conclusions

This work has analyzed the detailed rules for the description of the information structured used in context of KRP systems, using markup languages. Some markup technologies based on Java are considered.

The best choice is a model of the XML, and from the point of view of the java technologies for the processing of XML documents, it is found that for practical application JAXB (object interrogation, processing in memory) and JAXP (linear, facing processing on the fragments identifying and dealing with events) are more appropriate to be used. The effort of JAXB programming is less and object processing is promoted.

In addition, by Voice XML can be describes the smart interfaces of the KRP systems based on voice.

Bibliography

[1] Angles, R.; Gutierrez, C. (2005); Querying RDF Data from a Graph Database Perspective, In: *Gómez-Pérez A., Euzenat J. (eds) The Semantic Web: Research and Applications. ESWC 2005, Lecture Notes in Computer Science, vol 3532*, Springer, Berlin, Heidelberg, 2005.

[2] Berners-Lee, T. (1989); *Information Management: A Proposal*, CERN, [Online] Available: https://www.w3.org/History/199/proposal.html, Accessed on December 20, 2016.

[3] Buraga, S. (2004); *Semantic Web. Fundamente şi aplicaţii*, Matrix Rom, Bucureşti, 2004.

[4] Coleman, J.; Willis, D. (1997); *SGML as a Framework for Digital Preservation and Access, Commission on Preservation and Access*, Washington DC, 1997.

[5] Dănciulescu, D. (2015); Formal Languages Generation in Systems of Knowledge Representation Based on Stratified Graphs, *Informatica*, 26(3), 407-417, 2015.

[6] Dănciulescu, D.; Colhon, M.; Grigoraş, G. (2017); Right-Linear Languages Generated in Systems of Knowledge Representation based on LSG, *BRAIN. Broad Research in Artificial Intelligence and Neuroscience*, 8(1), 42-51, 2017.

[7] Fensel, D.; van Harmelen, F.; Horrocks, I.; McGuinness, D.L.; Patel-Schneider, P.F. (2001); OIL: An Ontology Infrastructure for the Semantic Web, http://www.cs.man.ac.uk/~horrocks/Publications/download/2001/IEEE-IS01.pdf (Last visited on 9/09/2017).

[8] Hefley, W.E.; Murray, D. (1993); Intelligent User Interfaces, In Proceedings of IUI'93, Orlando, Florida, ACM Press, NY, 3-10, 1993.

[9] Horvitz, E. (1999); Principles of Mixed-Initiative User Interfaces, In Proc. of CHI, 159-166.

[10] Kent, R.E. (2000); Conceptual Knowledge Markup Language (CKML): An introduction, Netnomics 2, 139-169.

[11] Kistner, G.; Nuernberger, Ch. (2014); Developing User Interfaces using SCXML State charts, *NVIDIA*, Publication Rights Licensed to ACM, http://phrogz.net, 1-7, 2014.

[12] Meenakshi, A.; Aghila, R.; Suganthi, P.; Kavya, S.(2016); A Knowledge Representation Technique for Intelligent Storage and Efficient Retrieval using Knowledge based Markup Language, *Indian Journal of Science and Technology*, 9(8), 1-8, 2016.

[13] Mohameed Saeed, A.A. (2017); Intelligent Interfaces for Knowledge Representation and Processing Systems, *Proceedings of ICVL 2017*, 370-375. 2017.

[14] Negru, V.; Grigoraş, G/; Dănciulescu, D. (2015); Natural Language Agreement in the Generation Mechanism based on Stratified Graphs, *Proceedings of the 7th Balkan Conference on Informatics Conference (BCI '15)*, ACM, New York, NY, USA, Article 36, 1-8, 2015.

[15] Salih, D. (2014); Natural User Interfaces, LM Research Topics in HC, http://www.cs.bham.ac.uk/, 2014.

[16] Sora, J.C.; Sora, S.A. (2012); Artificial Education: Expert systems used to assist and support 21st century education, *GSTF Journal on Computing (JoC)*, 2(3), 2012.

[17] Sowa, J.F. (2008); Conceptual Graphs, In *F. van Harmelen, V. Lifschitz, and B. Porter (Eds.): Handbook of Knowledge Representation*, Elsevier, 213-237, 2008.

[18] Ţăndăreanu, N. (2000); Knowledge Bases with Output, Knowl. Inf. Syst., 2(4), 438-460.

[19] Tăndăreanu, N.(2007); Communication by Voice to Interrogate an Inheritance Based Knowledge System. Research Notes in Artificial Intelligence and Digital Communications, *7th International Conference on Artificial Intelligence and Digital Communications*, 107, 1-15, 2007.

[20] Vohra, A.; Vohra, D. (2006); Pro XML Development with Java Technology, Apress.

[21] [Online] DAML tools; Available: http://www.daml.org/tools/, Accessed on December 15, 2016.

[22] [Online] DLML; Available: http://co4.inrialpes.fr/xml/dlml/, Accessed on December 15, 2016.

[23] [Online] HTML; Available: https://en.wikipedia.org/wiki/HTML, Accessed on September 15, 2016.

[24] [Online] JAVA XML parsers; Available: http://docs.oracle.com/javase/8/docs/api/index.html, Accessed on December 15, 2016.

[25] [Online] KIF; Available: http://www-ksl.stanford.edu/knowledge-sharing/kif/, Accessed on September 15, 2016.

[26] [Online] Ontology Markup Language; Available: http://www.ontologos.org/OML/OML%200.3.htm, Accessed on December 15, 2016.

[27] [Online] RDF/XML; Available: https://www.w3.org/TR/rdf-syntax-grammar/, Accessed on December 15, 2016.

[28] [Online] SGML - ISO 8879:1986; Available: https://www.iso.org/obp/ui/#iso:std:iso:8879:ed-1:v1:en, Accessed on December 15, 2016.

[29] [Online] VXML 2.0; Available: https://www.w3.org/tR/voicexml20/#dml1.4, Accessed on December 15, 2016.

[30] [Online] W3C (2015); State Chart XML (SCXML): State Machine Notation for Control Abstraction, Available: https://www.w3.org/TR/scxml/, Accessed on December 15, 2016.

[31] [Online] XML COVER PAGES; Available: http://xml.coverpages.org, Accessed on December 15, 2016.