

HYBRID SEARCH AND DELIVERY OF LEARNING OBJECTS SYSTEM

Anthony N. Ilukwe and Yevgen Biletsky

University of New Brunswick, Fredericton, Canada

Received 2013-09-15; Revised 2013-09-22; Accepted 2014-01-25

ABSTRACT

Retrieving learning material from the internet is a tedious process that has begged for a solution to filter out of the cluster of data and irrelevant material on the internet and deliver material that is relevant to a specific user. The Hybrid Search and Delivery of Learning Objects (HSDLO) system, put forward in this study, facilitates the personalized search and delivery of such learning material from the internet. The system combines a number of mechanisms to perform this: Keyword-based search, concept-based search and personalization. The keyword-and concept-based search methods are responsible for establishing the relevance of each learning material retrieved from the web. The system presented in this study builds upon work done in the previous iteration by additional functionality; further decoupling the subsystems to improve modularity; perfection of the personalization subsystem; and a redesign of the user interface to a simpler form with Web2.0 sensibilities. Additionally, the personalization subsystem is substantially extended, allowing for a learner to have a profile active within the system during a session in which he or she is logged in and following a search, for the profile to be adapted and stored in memory for subsequent sessions. This functionality has been tested and successfully evaluated.

Keywords: Retrieving Learning, Hybrid Search, Delivery of Learning Objects

1. INTRODUCTION

The system presented in this study builds upon work done in the previous iteration of HSDLO (Biletskiy *et al.*, 2012) by additional functionality; further decoupling the subsystems to improve modularity; perfection of the personalization subsystem; and a redesign of the user interface to a simpler form with Web2.0 sensibilities

The need for a mechanism to deliver learning material from the internet-material that matches the personal attributes and preferences of a learner-has led to the development of a software system that makes this feasible.

1.1. Overview

The HSDLO system allows a learner (user) to create a profile detailing their personal interests and run a personalized search to find Learning Objects (LO) by comparing values of corresponding attributes in the learner profile and Learning Object Metadata (LOM), taking into consideration the degree of relevance assigned to each attribute in the learner profile.

1.2. Scope of the System

A user (learner) can create a personal profile containing his/her attributes and the weights and values assigned to them.

This profile resides within the Personalization subsystem and stored as an XML file in the "users" directory. Whenever a registered user logs into the system, their respective user profile data is loaded into memory where it remains as subsequent actions are performed on the system during the session.

The system contains a set of rules for comparing profile attributes to LOM attributes. The user must enter desired search keyword (s) triggering the system to search for corresponding learning objects.

After running the search, the system processes the learning objects and returns the most relevant ones to the user. The system must provide access to the learning objects selected by the user and based on which learning objects are selected, make necessary adjustments to the learner profile to reflect the choices, allowing the profile

Corresponding Author: Yevgen Biletsky, University of New Brunswick, Fredericton, Canada

to be updated every time a user runs a search. This is achieved by altering the XML file that corresponds to the profile of the user who is currently logged in.

There should be a simple graphical interface-consisting of a text fields, text areas and button-through which the user can interact with the system.

1.3. Objectives

The objectives of this project are to fully implement the personalization subsystem; allowing for the user to create a profile that is saved in memory; enabling the user profile to be active while the user is logged in; and enabling the user profile to be written to memory after it has been adapted following a search. The user interface will also be designed to maximize simplicity and ease of use.

1.4. Definitions, Acronyms and Abbreviations

- LO = Learning object
- LOM = Learning object metadata
- LOR = Learning object repository
- HSDLO = Hybrid search and delivery of learning objects

2. DESCRIPTION OF PRE-EXISTING SYSTEM

The pre-existing HSDLO system (Biletskiy *et al.*, 2012) combines keyword-based and concept-based search to establish the relevance of each learning object to the query triggered by the user. The personalization subsystem incorporates a set of rules that govern the comparison of the user profile attributes to LOM

attributes. The relevance of each search result is calculated based on a comparison of the learner (user) profile and LO descriptions. This comparison is based on the values assigned to parameters of the learner profile, attributes of the LO descriptions and the priority of these characteristics and attributes assigned by the learner.

3. HSDLO SYSTEM, POST-IMPLEMENTATION

3.1. System Architecture

The top-level architecture HSDLO system is presented in **Fig. 1**. The system is decomposed into the following subsystems: Personalization, Crawler, LOSearch, Comparator and Utilities. The Crawler is dedicated to the task of automatically downloading Los from available LORs, NEEDS.org in this case, in order to populate the LOR with data to be used in subsequent searches. The LOSearch subsystem facilitates the search of Los in the LOR, based on a search query transmitted by the user. The user (Learner) delivers a search query by specifying the keyword(s) for the Los that he or she intends to retrieve. LOSearch uses these keywords to perform keyword-based and concept-based search of the LOs. After the query has been issued by the user, the system retrieves the user profile, which is analyzed for the purpose of a comparison with the retrieved LOMs. The personalization subsystem rearranges the results using the profile of the learner and the set of rules encapsulated within the personalization subsystem.

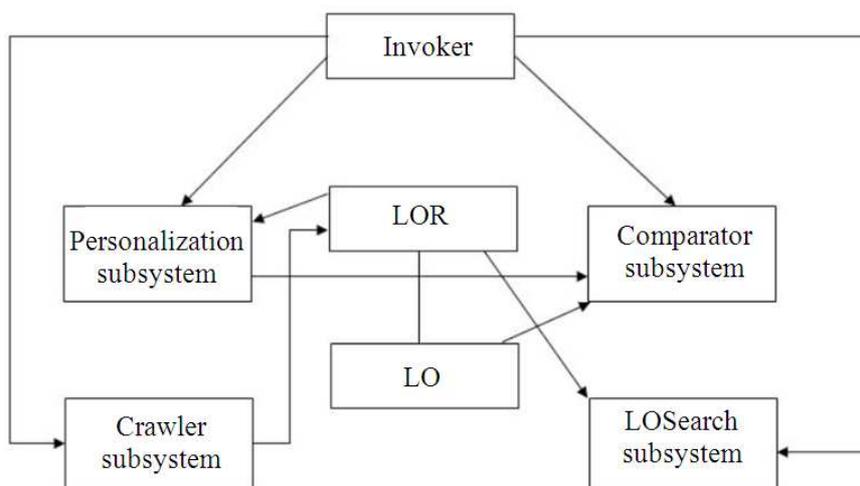


Fig. 1. HSDLO high level architecture

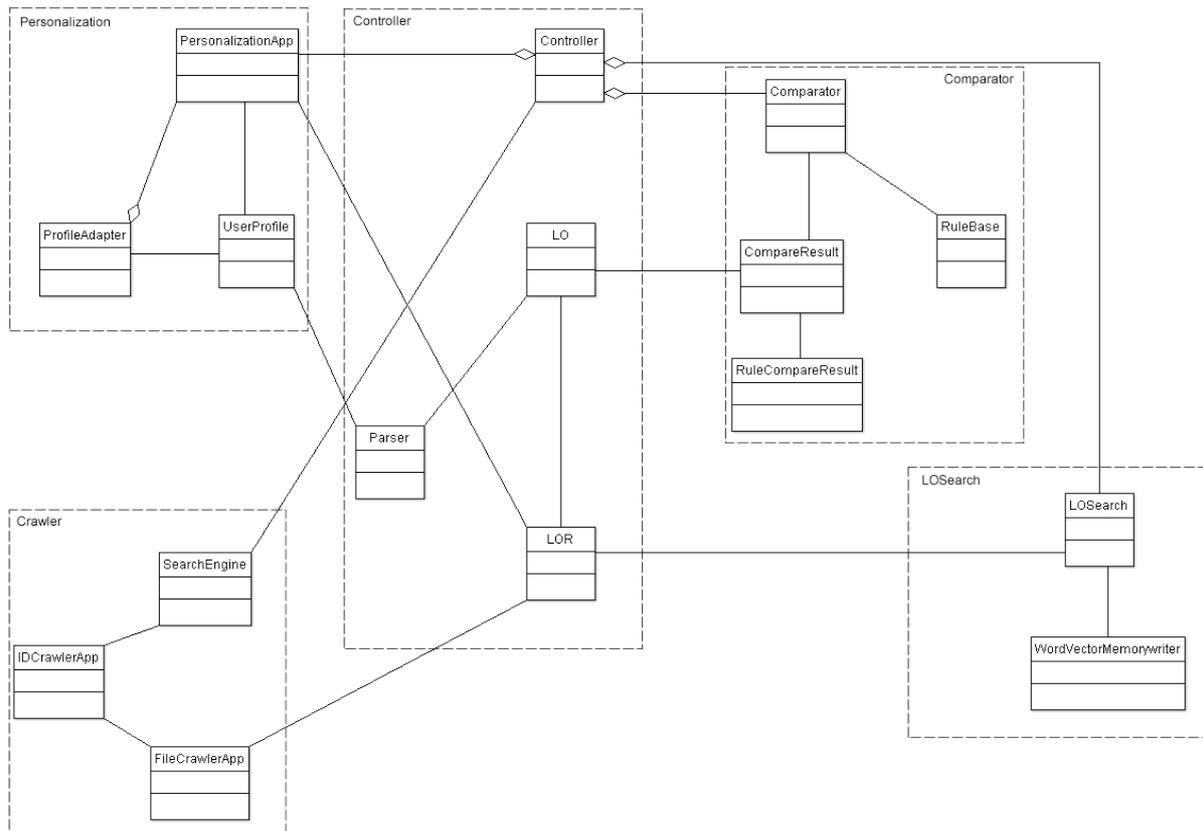


Fig. 2. HSDLO subsystem decomposition

3.2. Subsystem Decomposition

The system is decomposed into the following subsystems: Personalization, Crawler, LOSearch, Comparator and Utilities (Fig. 2).

Personalization the personalization adapts a profile to reflect the feedback from the results given after a search by using the profile adapter.

Crawler the web crawler stores the data collected from NEEDS in XML format.

LOSearch the search engine sorts the learning objects retrieved from a search query into a learning object repository. After a keyword search is performed, LOSearch uses the WordNet API to find synonyms and then searches each learning objects meta-data.

Comparator the comparator interacts with the Parser to produce a Parsed LOM object. It controls the comparison of the Parsed Profile with Parsed LOM then gives a score defined by the rules in the Rulebase.

Utilities this subsystem merely houses the LOs, LORs and XML parsers. The rationale is to decouple the search data from the aforementioned processing subsystems.

The next subsections describe in more detail the LOSearch and Personalization subsystems (Fig. 3).

3.3. LOSearch: Keyword and Concept-Based Search

Biletskiy *et al.* (2012) outlined the process of keyword and concept-based search in HSDLO as follows:

Many methods of searching text documents are based on looking for specific words or different forms of these words in the text. This is based on the scenario in which the user enters a query that contains a number of words. One of the common methods is to create an index of words to the documents that contain them in order to speed up the search (Wirth, 1976). First, all the documents are processed and their words are separated. Then a list of all words is created. A list of documents that contain each word is attached to it. When the user enters some keywords to search, the list of the documents connected to each word is returned.

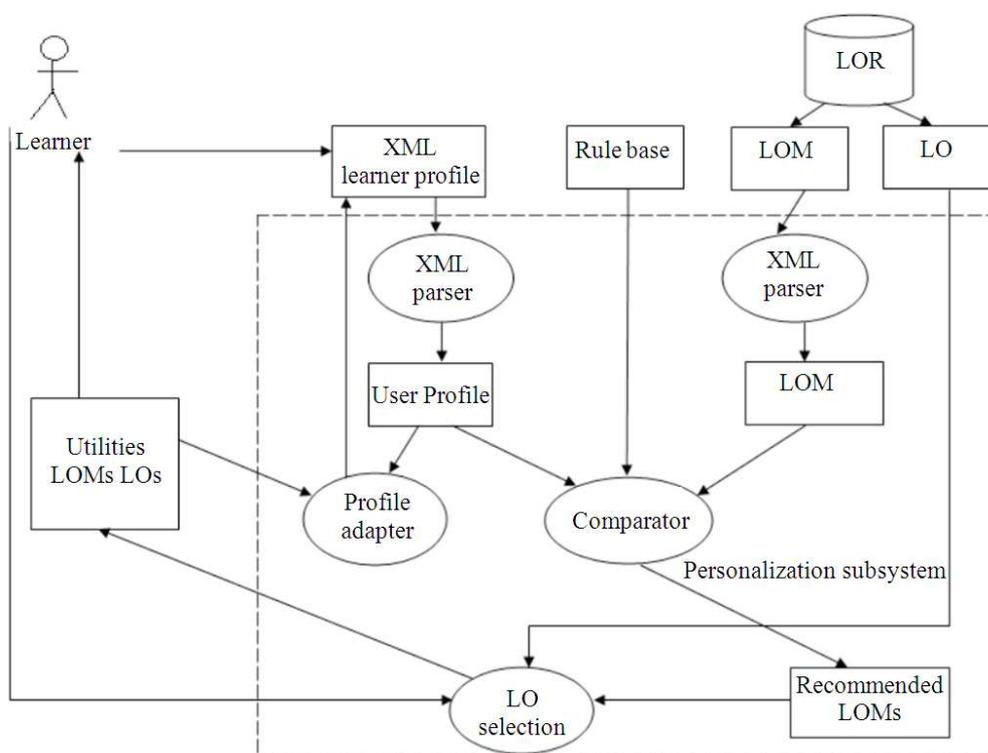


Fig. 3. Personalization subsystem

Boolean operations can be implemented in this method. Usually a search algorithm ranks the text documents based on their relevance to the user query. In this scenario, the document that is most similar to the user query, based on some criteria, appears first in the list. For comparison and defining the criteria, the query and the documents should be defined in a model. Most of the models consider a document as a set of unordered words, namely bag of words Chwodhury, 2004. Then each document is represented by a vector with length of the number of all the words in the document collection. Calculating the elements of this vector can be done in different ways.

Term Frequency-Inverse Document Frequency (TFIDF) is the approach employed for analysis of the text documents by the HSDLO system. Biletskyi *et al.* (2012) elucidates on this method as follows:

Term Frequency-Inverse Document Frequency (TFIDF) is one of the commonly used methods

for modeling and comparing text documents. Term Frequency is defined as the number of times a word occurred in a document. Document Frequency is defined as the number of documents in which the word occurs at least once over all documents. The documents are presented as vectors ($\vec{d} = (d^{(1)}, d^{(F)})$) that consist of keywords with their term frequencies so that the documents with similar content have similar vectors. The new document is presented as a vector \vec{d} as well. The attribute values in the vectors are presented as $TF(w,d)$. The keywords are distinct words or word parts, which distinguish the existing categories, or words that are prominent in the new document which could be selected from the documents by pruning infrequent words to remove most spelling errors and pruning high frequency words to eliminate non-content words like “the”, “and”, or “for”. The inverse document frequency for each keyword $IDF(w)$ is calculated based on the following formula:

$$IDF(w) = \log = \left(\frac{|D|}{DF(w)} \right)$$

where, |D| is the total number of documents. DF(w) represents the document frequency of the keyword.

The inverse document frequency of a keyword is high if it occurs in only one document, which makes it more significant for distinguishing that particular document. The weight value $d^{(i)}$ in a document d for each keyword is calculated by multiplying its term frequency in the document and its inverse document frequency:

$$d^{(i)} = TF(w_i, d) \times IDF(w_i)$$

The similarity between documents can be calculated by computing the cosine value between their representative vectors. The query can also be modeled by a vector in the same way and be compared with each document to find the most similar document to it. The cosine value of two vectors is calculated based on the following formula:

$$\cos(\vec{d}_i, \vec{q}) = \frac{\vec{d}_i \cdot \vec{q}}{\|\vec{d}_i\| \cdot \|\vec{q}\|}$$

The documents are sorted based on their cosine value in decreasing order. There are other ways to create vectors for documents. For example, only term frequency can be considered as the attribute value. In the simplest model the vectors can be binary, which shows whether a particular document contains the keyword or not.

Biletsky *et al.* (2012) further expatiates on the need for semantic expansion of search queries:

The use of keywords for comparing documents and searching in documents gives good results, but it does not consider the semantics of the text. If a user wants to search for something which she has in mind, she must know the exact keywords. This is helpful in many situations, but in many other scenarios the user does not know exactly which keywords she should use. There may be many different keywords to express a single concept and there may be words that can represent different concepts. To be more effective, search engines may include the semantic relations between the words and document. In other words, concept-based information retrieval is search for documents based on their meaning rather than on the presence of keywords in the document.

The relevance of LOs, alluded to earlier, to a learner's (user's) query can be described with the following expression:

$$R_D = w_k R_k + w_c R_c$$

where, R_k and R_c are relevancy measures for keyword-based and concept-based search, respectively and w_k and w_c are their corresponding weights.

The discussed search subsystem is itself composed of two major subsystems: Keyword-and concept-based document search subsystems. The proposed system uses an ontology to represent the hierarchy of concepts- WordNet an open access lexical database of the English language, developed by researchers at Princeton University. WordNet groups nouns, verbs, adjectives and adverbs into sets of cognitive synonyms, known as synsets, with each of these synsets representing a precise concept. These Synsets are interlinked by means of conceptual-semantic and lexical relations that are defined in the WordNet database.

Biletsky *et al.* (2012) describes the comparison of the term and concept vectors, following a new query, as follows:

When the learner issues a query, the query engine creates a term vector and a concept vector based on the query. Then, these two vectors are compared for similarity to the term and concept vectors of each document. This gives two measures for sorting the document based on relevance: Keyword-based similarity and concept-based similarity.

3.4. Personalization

The Personalization subsystem is an extension of the main HSDLO system that allows users to create profiles describing their personal attributes and to run a personalized search in the LOR. It then ranks the retrieved Los after performing a comparison of the values of learner profile attributes and LOM, taking into consideration the level of significance assigned to each learner profile attribute. This comparison is implemented as a set of rules written in JavaScript and stored in XML. The Personalization subsystem finally adjusts the learner's profile based on implicit feedback from the learner on the Los retrieved from the personalized search.

Biletsky *et al.* (2012) articulates the context in which the personalization subsystem interlocks the HSDLO system:

The HSDLO system allows a learner to create a profile, which consists of a set of attributes with their values and importance. The learner can access the profile through the system at any time to make and save changes. Once a learner has a profile, she is able to run searches of learning objects in learning object repositories. Before running the search, the learners must select which repository they wish to search from the list of available repositories maintained by the system. The default repository is the most recent one searched. When the search is run, the personalization subsystem scans through the designated learning object repository, parsing the metadata of the learning objects and comparing it to the learner's profile, assigning a score to each learning object in the repository. Once the search is complete, the subsystem returns the top ranked learning objects to the learner. The learner can then select which learning objects to download and the system acquires them and saves the learner's selections. Using the aggregate set of selections made by the learner, the subsystem makes modifications to the learner profile to better represent his or her preferences.

The following are the aforementioned attributes that make up the learner profile of the HSDLO system:

- Qualifications
- keywords_of_interests
- proficiency_writing
- proficiency_reading
- proficiency_speaking
- input_output_technology
- priority_of_goal
- goal
- proficiency_listening
- skill_level
- physical_preferences
- age
- activity_status
- role
- language
- activity_type
- cognitive_preferences

HSDLO system data is stored in XML format due to its efficacy and simplicity in the storage and access of data. It is also platform independent.

- **User profile**

The user profile data is be stored as an XML file allowing for quick access as well as make it easy to edit. An example is provided below:

```
<!--Profile information goes here-->
<Attribute>
  <Name>xxxxxxxxxxxxxxxx</Name>
  <Value>xxxxxxxxxxxxxxxx</Value>
  <Weight>xxxxxxxxxxxxxxxx</Weight>
</Attribute>
```

- **Learning Objects IDs**

Learning objects Ids will be stored in an XML file which provides easy access and is more usable. An example is provided below:

```
<!--LO information goes here-->
<List>
  <Name>xxxxxxxxxxxxxxxx</Name>
  <URL>xxxxxxxxxxxxxxxx</URL>
  <ID>xxxxxxxxxxxxxxxx</ID>
</List>
```

- **Learning objects files**

Learning objects will be stored in an XML file which provides easy access and is more usable. One file will contain information about a particular web site. An example is provided below:

```
<List>
<Name>xxxxxxxxxxxxxxxx</Name>
  <URL>xxxxxxxxxxxxxxxx</URL>
  <LOMURL>xxxxxxxxxxxxxxxx</LOMURL>
</List>
```

- **Rule base**

The rules that are used by the comparator to calculate the score of Los is stored as a file in XML format. An example is provided below:

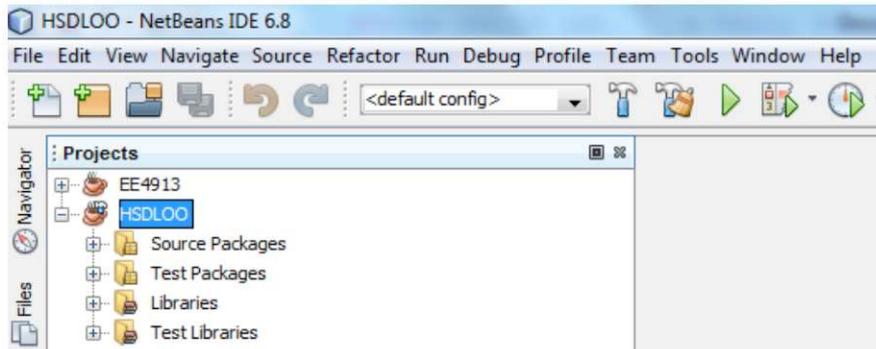
```
<!-- Rule Base information goes here-->
<Rule>
  <ProfileAttribute>xxxxxxxxxxxxxxxx</ProfileAttri
  bute>
  <MetaAttribute>xxxxxxxxxxxxxxxx</MetaAttribute>
</Rule>
```

Appendix 1 and 2 contains more examples of all the currently used attributes of LOM and learner profile. The full set of comparison rules is presented in Appendix 3.

Appendix 1: Usage

Instructions 1. Open the project in NetBeans as follows: File >> Open Project. Browse to the project directory and select.

When the project is open, click run project to start the system.



2. After the system launches, a window like one below appears. If a learner profile already exists, login with the appropriate user name and password, otherwise, click on the "create new user" tab.

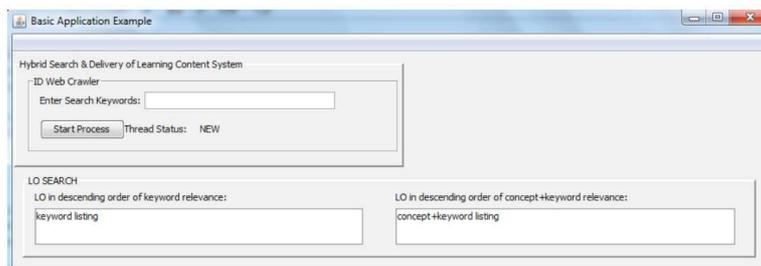


3. Fill in the new user form and register a user name and password that you will remember. This data is stored in the HSDLO/src/users directory. Use the slider to adjust the weight of the attributes as appropriate.



4. After logging in, the main application window seen below is shown on the screen. Enter keywords into the text area and click “start process” to begin the search.

The system will take a few seconds to run the search and the personalization process and a list of retrieved learning objects will appear in the larger text areas. The weights assigned to the learner profile attributes can be observed to have changed due to the profile adaptation that takes place during the personalization process.



Appendix 2: A Sample Learner Profile in the HSDLO System

Note. Reprinted from A Rule-based System for Hybrid Search and Delivery of Learning Objects to Learners, by Y. Biletskiy, H. Baghi, J. Steele, and R. Vovk, Appendix 1. 2010. Reprinted with permission.

```

<Attributes>
<Attribute id="1">
  <Name>age</Name>
  <Value>22</Value>
  <Weight>0.5</Weight>
</Attribute>
<Attribute id="2">
  <Name>language</Name>
  <Value>en-US</Value>
  <Weight>0.5</Weight>
</Attribute>
<Attribute id="3">
  <Name>proficiency_writing</Name>
  <Value>1</Value>
  <Weight>0.5</Weight>
</Attribute>
<Attribute id="4">
  <Name>proficiency_reading</Name>
  <Value>1</Value>
  <Weight>0.5</Weight>
</Attribute>
<Attribute id="5">
  <Name>proficiency_speaking</Name>
  <Value>1</Value>
  <Weight>0.5</Weight>
</Attribute>
<Attribute id="6">
  <Name>proficiency_listening</Name>
  <Value>1</Value>
  <Weight>0.5</Weight>
</Attribute>
<Attribute id="7">
  <Name>priority_of_goal</Name>
  <Value>3</Value>
  <Weight>0.5</Weight>
</Attribute>
<Attribute id="8">
  <Name>qualifications</Name>
  <Value>programming, java, C++</Value>
  <Weight>0.5</Weight>
</Attribute>

```

```
<Attribute id="9">
  <Name>goal</Name>
  <Value />
  <Weight>0.5</Weight>
</Attribute>
<Attribute id="10">
  <Name>activity_type</Name>
  <Value>study</Value>
  <Weight>0.5</Weight>
</Attribute>
<Attribute id="11">
  <Name>activity_status</Name>
  <Value>active</Value>
  <Weight>0.5</Weight>
</Attribute>
<Attribute id="12">
  <Name>cognitive_preferences</Name>
  <Value>4</Value>
  <Weight>0.5</Weight>
</Attribute>
<Attribute id="13">
  <Name>physical_preferences</Name>
  <Value />
  <Weight>0.5</Weight>
</Attribute>
</Attribute>
<Attribute id="14">
  <Name>skill_level</Name>
  <Value>4</Value>
  <Weight>0.5</Weight>
</Attribute>
<Attribute id="15">
  <Name>role</Name>
  <Value>learner</Value>
  <Weight>0.5</Weight>
</Attribute>
<Attribute id="16">
  <Name>keywords_of_interests</Name>
  <Value>java</Value>
  <Weight>0.5</Weight>
</Attribute>
<Attribute id="17">
  <Name>input_output_technology</Name>
  <Value>java</Value>
  <Weight>0.5</Weight>
</Attribute>
</Attributes>
```



```

<Value>high</Value>
</Attribute>
<Attribute>
  <Name>context</Name>
  <Value />
</Attribute>
<Attribute>
  <Name>typical_age_range_max</Name>
  <Value />
</Attribute>
<Attribute>
  <Name>difficulty</Name>
  <Value>easy</Value>
</Attribute>
<Attribute>
  <Name>typical_learning_time</Name>
  <Value />
</Attribute>
<Attribute>
  <Name>taxon_path_concept_and_discipline</Name>
  <Value />
</Attribute>
  Name>
  <Value />
  </Attribute>
  <Attribute>
    <Name>taxon_path_idea_discipline</Name>
    <Value />
    </Attribute>
    <Attribute>
      <Name>intended_end_user_role</Name>
      <Value>teacher</Value>
      </Attribute>
    </Attributes>
  </Attributes>

```

Reprinted from A Rule-based System for Hybrid Search and Delivery of Learning Objects for Learners, by Y. Biletskiy, H. Baghi, J. Steele, and R. Vovk, Appendix 3. 2010. Reprinted with permission.

Appendix 4: Sample of HSDLO Rule Base

```

=
<rules>
= <rule>
  <param source="userprofile" name="age" type="int" />
  <param source="lom" name="typical_age_range_min" type="int" />
  <param source="lom" name="typical_age_range_max" type="int" />

```

```

<rulescript lang="JavaScript">
  <![CDATA[
      function test(){
        if ( age >= typical_age_range_min * 10 && age <=
          typical_age_range_max * 10 )
          return 1;
        else
          return 0;
      }
  ]]>
</rulescript>
</rule>

<rule>
  <param source="userprofile" name="language" type="string" />
  <param source="lom" name="content_language" type="string" />
  <rulescript lang="JavaScript">
    <![CDATA[
        function test(){
          if (language == content_language )
            return 1;
          else
            return 0;
        }
    ]]>
  </rulescript>
</rule>

<rule>
  <param source="userprofile" name="proficiency_writing" type="int" />
  <param source="lom" name="interactivity_type" type="string" />
  <param source="lom" name="interactivity_level" type="string" />
  <rulescript lang="JavaScript">
    <![CDATA[
        function test(){
          if ( proficiency_writing >= 2 && (interactivity_type
            == "active" || interactivity_type == "exposive" ) )
            return 1;

          if ( proficiency_writing == 1 && (interactivity_type == "active" ||
            interactivity_type == "exposive" ) && (interactivity_level == "medium" ||
            interactivity_level == "low" ) )
            return 1;

          if ( proficiency_writing == 0 && (interactivity_type == "active" ||
            interactivity_type == "exposive" ) && (interactivity_level == "low" ) )
            return 1;

          return 0;
        }
    ]]>
  </rulescript>
</rule>

<rule>
  <param source="userprofile" name="proficiency_reading" type="int" />

```

```

<param source="lom" name="interactivity_type" type="string"/>
<param source="lom" name="interactivity_level" type="string"/>
_ <rulescript lang="JavaScript">
  _ <![CDATA[
      function test(){
        == "exposive" )          if ( proficiency_reading >= 2 && (interactivity_type
                                return 1;
                                if ( proficiency_reading == 1 && (interactivity_type == "exposive") &&
                                (interactivity_level == "medium" || interactivity_level == "low" )
                                return 1;
                                if ( proficiency_reading == 0 && (interactivity_type == "exposive") &&
                                (interactivity_level == "low" )
                                return 1;
                                return 0;
                                }
                                function test(){
        == "exposive" || interactivity_type == "active" )
                                if ( (proficiency_reading >= 2) && (interactivity_type
                                return 1;
                                return 0;
                                }
      ]>
</rulescript>
</rule>
_ <rule>
  <param source="userprofile" name="proficiency_listening" type="int"/>
  <param source="lom" name="format" type="string"/>
  <param source="lom" name="interactivity_type" type="string"/>
  <param source="lom" name="interactivity_level" type="string"/>
  _ <rulescript lang="JavaScript">
    _ <![CDATA[
        function test(){
          || format == "video" )
            if ( proficiency_listening >= 2 && (format == "audio"
            return 1;
            return 0;
            }
    ]>
  </rulescript>
</rule>
_ <rule>
  <param source="userprofile" name="proficiency_speaking" type="int" />
  <param source="lom" name="format" type="string"/>
  <param source="lom" name="interactivity_type" type="string"/>
  <param source="lom" name="interactivity_level" type="string"/>

```

```

=> <rulescript lang="JavaScript">
=> <![CDATA[
=>         function test(){
=>             if ( proficiency_speaking >= 2 && (format == "audio"
=> || format == "video") && interactivity_type != "exposive" && interactivity_level >
=> 2)
=>                 return 1;
=>             return 0;
=>         }
=>     ]>
=> </rulescript>
=> </rule>

=> <rule>
=> <param source="userprofile" name="priority_of_goal" type="int"/>
=> <param source="lom" name="duration" type="int"/>
=> <rulescript lang="JavaScript">
=> <![CDATA[
=>         function test(){
=>             if ( priority_of_goal <= 2 )
=>             {
=>                 if (duration < 200 && typical_learning_time
=> < 300) return 1;
=>                 else
=>                     return 0;
=>             }
=>             return 1;
=>         }
=>     ]>
=> </rulescript>
=> </rule>

=> <rule>
=> <param source="userprofile" name="qualifications" type="string"/>
=> <param source="lom" name="taxon_path_concept_and_discipline" type="string"/>
=> <param source="lom" name="taxon_path_skilllevel_prerequisite" type="int"/>
=> <rulescript lang="JavaScript">
=> <![CDATA[
=>         function test(){
=>             if ( taxon_path_skilllevel_prerequisite > 2 )
=>             {
=>                 var quals = qualifications.split(",");
=>                 for ( var i = 0 ; i < quals.length ; ++i
=>                 {
=>                     if ( quals[i] ==
=> taxon_path_concept_and_discipline)
=>                         return 1;
=>                 }
=>                 return 0;
=>             }
=>             else
=>                 return 1;
=>         }
=>     ]>
=> </rulescript>
=> </rule>

=> <rule>
=> <param source="userprofile" name="goal" type="string"/>
=> <param source="lom" name="taxon_path_skilllevel_prerequisite" type="int"/>

```

```

<rulescript lang="JavaScript">
  <![CDATA[
        function test(){
            return 1;
        }
    ]>
</rulescript>
</rule>

<rule>
<param source="userprofile" name="activity_type" type="string"/>
<param source="lom" name="context" type="string"/>
<rulescript lang="JavaScript">
  <![CDATA[
        function test(){
            if ( ( context == "school" || context ==
"higher_education" ) && activity_type == "study" )
                return 1;
            if ( context == "training" && activity_type == "work" )
                return 1;
            if ( context == "research" && activity_type ==
"research" )
                return 1;
            return 0;
        }
    ]>
</rulescript>
</rule>

<rule>
<param source="userprofile" name="activity_status" type="string"/>
<param source="lom" name="context" type="string"/>
<rulescript lang="JavaScript">
  <![CDATA[
        function test(){
            return 1;
        }
    ]>
</rulescript>
</rule>

<rule>
<param source="userprofile" name="cognitive_preferences" type="int"/>
<param source="lom" name="difficulty" type="string"/>
<rulescript lang="JavaScript">
  <![CDATA[
        function test(){
            if ( difficulty == "easy" ){
                if ( cognitive_preferences >= 1 ) {
                    return 1;
                }
            }
        }
    ]>
</rulescript>
</rule>

```

```

    }
  }
  if ( difficulty == "medium" ){
    if ( cognitive_preferences >= 2 ) {
      return 1;
    }
  }
  if ( difficulty == "difficult" ){
    if ( cognitive_preferences >= 3 ) {
      return 1;
    }
  }
  return 0;
}
}
]]>
</rulescript>
</rule>
<rule>
<param source="userprofile" name="physical_preferences" type="string"/>
<param source="lom" name="difficulty" type="string"/>
<rulescript lang="JavaScript">
  <![CDATA[
        function test(){
          return 1;
        }
    ]]>
</rulescript>
</rule>
<rule>
<param source="userprofile" name="skill_level" type="int"/>
<param source="lom" name="taxon_path_skilllevel_prerequisite" type="int"/>
<rulescript lang="JavaScript">
  <![CDATA[
        function test(){
          if ( skill_level > taxon_path_skilllevel_prerequisite )
            return 1;
          else
            return 0;
        }
    ]]>
</rulescript>
</rule>
<rule>
<param source="userprofile" name="role" type="string"/>
<param source="lom" name="intended_end_user_role" type="string"/>
<rulescript lang="JavaScript">
  <![CDATA[
        function test(){
          if ( role == intended_end_user_role ||
            intended_end_user_role == "any" )
            return 1;
        }
    ]]>
</rulescript>
</rule>

```

```

        else
            return 0;
    }
}
</rulescript>
</rule>
<rule>
<param source="userprofile" name="keywords_of_interests" type="string" />
<param source="lom" name="title" type="string" />
<param source="lom" name="description" type="string" />
<rulescript lang="JavaScript">
<![CDATA[
function test(){
return 1;
return sim(keywords_of_interest,title + description );
}
}
</rulescript>
</rule>
<rule>
<param source="userprofile" name="input_output_technology" type="string" />
<param source="lom" name="requirement" type="string" />
<param source="lom" name="other_platform_requirement" type="string" />
<rulescript lang="JavaScript">
<![CDATA[
function test(){
if ( requirement == input_output_technology ) return
1;
if ( other_platform_requirement ==
input_output_technology ) return 1;
return 0;
}
}
</rulescript>
</rule>
</rules>

```

Table 1 shows the rules that the HSDLO system uses to perform comparison and the calculation of LO scores.

The comparison and calculation of the scores of Los is governed by a set of criteria. Biletskiy *et al.* (2012) states the following:

To facilitate the personalized search and delivery of learning objects to learners the scoring criterion $LOScore_j$ which estimates the suitability of the j -th learning object to the learner's personal profile, is defined as a function of the learner's preference to select interesting materials from learning objects delivered to the learner:

$$LOScore_j = \sum_{i=1}^n w_i \times RR_{ij}$$

Where:

RR_{ij} = Respond of the i -th comparison rule (**Table 1**) on the comparison of the j -th learning object with the learner profile

w_i = Coefficient of importance, which defines a level of influence of the i -th attribute of the learner profile (corresponding to the i -th comparison rule) on the selection of learning objects; the range of values of this coefficient is [0,1]

n = Total number of selection criteria (same as the number of attributes of learner profile or the number of comparison rules)

After estimating the suitability of the learning objects, they are sorted in order of decreasing values of $LOScore_j$ and the top scored learning objects are delivered to the learner. The learner explores the delivered documents' LOM and content. After the learning object is explored, the learner defines a degree of utility (usefulness) of the learning object $LOU_j = (1: \text{Useful}, 0.5: \text{Auxiliary}, 0: \text{Not useful})$. In reality, the learner's preference may change, or the learner may explicitly declare her preferences incorrectly. Therefore, the dynamic adjustment of the coefficients of importance will be necessary. The degree of utility serves as a criterion for adjustment as follows. If the learner has defined a learning object as useful or auxiliary, then a comparative analysis of the corresponding learner's profile and LOM attributes is

conducted. This analysis is conducted for all the learning objects delivered to the learner and is based on the principle that if the learning object is useful or auxiliary for the learner and the learner's profile and LOM attributes match ($RR_i = 1$) then the weight w_i is increased, but if $RR_i = 1$ then w_i is decreased.

Defining the operations of increase and decrease of w_i as operations with saturation, some coefficients of importance converge to 1 (meaning that the attribute is important), some converge to 0.1 (meaning that the attribute is not important) and some remain in between 0.1 and 1 (meaning that importance cannot be definite). The minimum coefficient is selected as to keep a chance to increase it if it again becomes important. In addition, the presented approach assumes that learning objects are searched within the same topic or theme that is interesting for the learner, because otherwise a learning object can be evaluated by the learner as not useful based on the theme, rather than on personal preferences.

Table 1. HSDLO system rules Reprinted from A rule-based system for hybrid search and delivery of learning objects to learners, by (Biletskiy *et al.*, 2012) Reprinted with permission

Rule#	LIP attribute involved	LOM attributes involved	Rule respond (0: Otherwise)
1	Birth date	Typical age range	1: if age fits in the typical age range
2	Language	Content language	1: if language = content language
3	Proficiency of writing of the language	Interactivity type Interactivity level	1: if proficiency satisfies the interactivity level and interactivity type is "active"
4	Proficiency of reading of the Language	Interactivity type Interactivity level	1: if proficiency satisfies the interactivity level and interactivity type is "active" or "exposive"
5	Proficiency of listening of the language	Format interactivity type interactivity level	1: if proficiency satisfies the interactivity level, interactivity type is "active" or "exposive" and format is "audio" or "video"
6	Proficiency speaking of the language	Format interactivity type interactivity level	1: if proficiency satisfies the interactivity level, interactivity type is "active" and format is "audio" or "video"
7	Priority of goal	Duration typical learning time	1: if priority of goal is high; or, if low, the duration and learning time must be short
8	Qualification description	Taxon path type Taxon path value	1: if qualification corresponds to the classification of the learning object
9	Activity type	Context	1: if learner's activity corresponds to the context of learning
10	Cognitive preferences	Difficulty	1: if the level of cognitive preferences of the learner is sufficient for the difficulty level of the learning object
11	Skill level	Taxon path type Taxon path value	1: if skill level is sufficient for the learning object
12	Role	Intended end user role	1: if the learner's role is desirable
13	Interests	Title description	1: if the learner's interests correspond to the topic of learning object
14	I/O technology	Requirement other platform requirements	1: if the learner has sufficient access to technology, e.g., multimedia

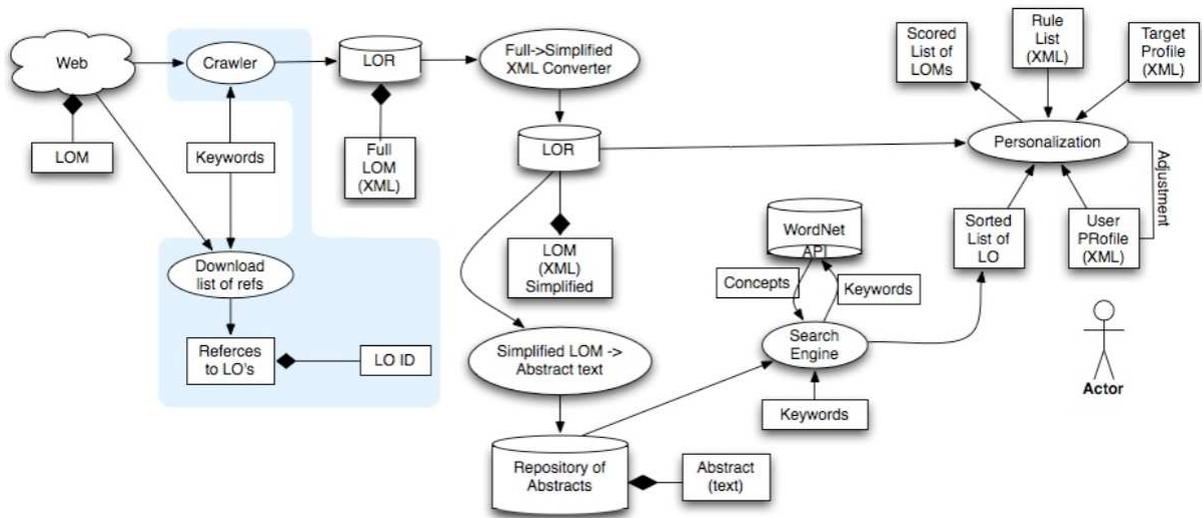


Fig. 4. HSDLO conceptual model

4. CONCLUSION

The conceptual model of the HSDLO systems is shown in Fig. 4. It illustrates the extraction of LOs, which are stored in the LOR, by the crawler. It also shows the extraction of LOM attributes and its conversion to a format that can be used by the personalization subsystem. This subsystem performs the appropriate adjustment to the user profile, following the processing of Los and user feedback. The adapted profile of each unique learner is saved in memory and used for subsequent searches by each unique learner.

5. REFERENCES

Biletskyi, Y., H. Baghi, J. Steele and R. Vovk, 2012. A rule-based system for hybrid search and delivery of learning objects to learners. *Interact. Techn. Smart Edu.*, 9: 263-279. DOI: 10.1108/17415651211284048

Wirth, N., 1976. *Algorithms + Data Structures*. 1st Edn., Pearson Education Canada, Englewood Cliffs, ISBN-10: 0130224189, pp: 366.