DOI: 10.2298/CSIS110615005S

Automatic Generation of E-Courses Based on Explicit Representation of Instructional Design

Goran Savić, Milan Segedinac, and Zora Konjović

Faculty of Technical Sciences, Trg D. Obradovića 6,
21000 Novi Sad, Serbia
{savicg, milansegedinac, ftn_zora}@uns.ac.rs

Abstract. This paper presents the system for automatic generation of IMS LD compliant E-Course from three components: machine readable explicit representation of instructional design, ontology of learning goals, and IMS Content Packaging compliant learning resources. For the explicit representation of instructional design, we have created a new domain-specific language named ELIDL which is aimed primarily at assistance to implementation of various pedagogical approaches in the course. Using ELIDL a teacher defines instructional design template which is one of the input parameters for the course generation. The system is verified by generating examples of the six different instructional designs for the Web Programming e-course via templates written in ELIDL.

Keywords: e-learning, instructional design, DSL, automatic course generation, IMS Learning Design.

1. Introduction

When a learning environment is switched from classical face-to-face to e-learning, an unavoidable task is development of electronic courses which are compliant with e-learning standards and shaped according to appropriate instructional design model(s). There are two globally adopted specifications that provide such course description – SCORM [1] and IMS Learning Design [2]. Creation of a new course or modification of an existing course compliant with any of these two standards is, at least, a time-consuming process. In addition, the realistic scenarios very often are those when existing courses should be modified. Most available tools for e-course creation make this task unduly hard because the sequences of learning activities and learning resources are interwoven and represented as a monolithic unit. While changing the course in some segment, it is necessary to deal with this monolithic representation, although only one of these segments which very often is instructional design, is usually the subject of the change.

Course development efforts can be significantly reduced by using systems for automatic course generation. The focus of this paper is an electronic course description which gathers learning resources and learning activities. It presents a system for automatic course generation with a focus on implementation of different instructional design strategies.

The rest of the paper is organized as follows. Concrete motivation for developing the system is described in the *Motivation* Section. The Section *Related Work* analyzes other systems for automatic course generation and it is particularly concerned about the role and implementation of instructional design in these systems. Our system is in detail presented in Sections *System Components* and *System Architecture and Functioning*. We have used the system to automatically generate six electronic courses in Web Programming differing in instructional strategies. The results are described in *Case Study* Section. At the end, conclusions and future research directions are given.

2. Motivation

The first-hand motivation for the research presented in this paper is demand to switch relatively large number of courses on the Faculty of Technical Sciences in Novi Sad (Serbia) to blended learning environment in a short time. Additionally, every semester brings some changes in the existing courses. Very often these changes are related to the course instructional design. There are two main reasons for applying changes to course's instructional design: gradual improvement of an ongoing teaching process, and studying/creating different instructional techniques. For these reasons it is useful to have a system that enables creating/changing the course instructional design easily. Therefore, we have decided to create a system that automates the course creation process by using a formal, machine readable description of instructional design as a base. By using this system, accompanied with ontology of learning goals and IMS Content Packaging compliant learning resources, we can automatically generate an electronic course in IMS LD format.

3. Related Work

In this section we analyze other systems for automatic generation of e-learning courses. These systems using different inputs generate an e-learning course as an output. For a course creation, two aspects have to be considered: (1) technical aspect which is about system functionalities, input/output formats, standard compliance and other characteristics relevant from the technical point of view, and (2) pedagogical aspect which includes ability to apply different instructional principles in a generated course.

Paper [3] describes a system for automatic course generation. The system generates the sequence of learning objects in IMS Content Packaging format on the basis of learning goals and student's knowledge state. Planning mechanism and PDDL (Planning Domain Definition Language) language are used for the creation of the sequence. Hernandez et al. in [4] also use PDDL plan for a course generation in IMS Learning Design format. A concrete learning activity that will be presented to a learner is chosen in real-time depending on the student's profile and learning goals. An approach to course generation based on learning goals ontology where learning resources are mapped to ontology nodes is presented in [5]. Nodes are hierarchically ordered and mastering one learning goal may be a precondition for other learning goals. A student's model contains her/his learning preferences and cognitive state. Ontology data and the student's model are inputs for generating a learning path. A similar approach is used in [6], but there is no defined learning goals ontology. Relations among learning objects are defined directly in the set of objects. The result is an e-learning course compliant with IMS Learning Design specification. All mentioned systems generate a course in one of globally accepted formats, which is done in our system, too. Additionally, our system has borrowed the idea of presenting learning goals by ontology and mapping learning objects to this ontology. Still, in contrast to our system, there is no explicit representation of the instructional design in these papers. The role of PDDL is to define a sequence of learning activities which is similar to the role of the instructional design in an e-learning course. But, PDDL is intended for the general planning and it is not sufficiently expressive to describe a variety of instructional designs in the e-learning course.

The pedagogical aspect is considered in paper [7], which presents a course generation system based on learning resources, learning goals, learner's profile and instructional design. Instructional design is defined separately from learning resources which is similar to our approach. But, it is defined by concrete learning activities in the course, not as a general template like in our paper. In our paper instructional design is defined abstractly and can be applied to any course. Paper [8] describes a web portal for defining pedagogical scenario and graphical interface of an e-learning course. A teacher defines these two components by answering an online questionnaire. The pedagogical scenario is represented by an XML file in IMS Learning Design format and this file is used for automatic course generation. Although, a pedagogical scenario is formally described in a machine-readable format, there is only a predefined set of pedagogical scenarios available.

Since we have decided to create a new domain-specific language for representing instructional design, it is important to analyze other similar researches focused on the formal representation of instructional design and the development of a domain-specific language for this purpose. Instructional design may be implicitly defined in the formal definition of the e-learning course. IMS LD specification introduces an XML-based language that enables an implicit definition of instructional design in a course by specifying learning activities and its organization. Similarly, LAMS [9] uses a specific language for specifying learning activities used in the system. These two languages show how instructional design may be implicitly defined altogether with all other course components. For creating an explicit representation of instructional design (independently from the concrete learning activities) we have to consider a teaching process more abstractly. It means that it is necessary to identify and model specific patterns in instructional design that are widely used in courses. One of the important researches in this area is *Pedagogical Patterns* project [10]. The project collects standard pedagogical patterns used in the teaching process. By this, teachers may find the most appropriate pedagogical pattern for a concrete teaching situation. Although these patterns are very useful for teachers and pedagogues, they are not intended for computer interpretation. The patterns are represented by textual descriptions, so they are not machine-readable. The Pedagogical Patterns project is mostly focused on the classical face-to-face learning. An idea for creating templates that can be used in an e-learning environment is presented in [11]. A template presents a course structure and it is presented graphically. Such a representation is not machine-readable too. Paper [12] also introduces the concept of templates in an e-learning course. Template is defined as a set of learning resources that contains specific pedagogical principles and these templates are copied in their original sequence into the course web site. Such an approach doesn't make an explicit distinction between learning resources and instructional design. The instructional design is not defined as a distinct component, but rather interwoven with learning resources. In [13], an adaptive e-learning environment SAP LSO is improved by involving templates in the system. Templates define a micro-strategy – a rule for organizing atomic knowledge items into the larger learning objects. Our paper is focused on a macro-strategy – how a sequence of learning objects is organized. There are only two predefined macro strategies in [13]. It is not possible to create a new macro-strategy as in our system. The paper [14] presents an approach that formalizes the representation of instructional design using SMID – the semantic model of instructional design. The model contains data about learner's knowledge state, learning goals, learning resources and an instructional strategy. The instructional strategy is defined using if-then rules. The sequence of learning actions is generated on the basis of learner's knowledge state and metadata about learning goal and learning object. Actions are composed of Gagne's Nine Events of Instruction. Here again, there is only a predefined set of instructional strategies available. The paper [15] presents an approach for converting pedagogical patterns described in [10] from textual format to the machine-readable IMS LD language. The result of this conversion is not a concrete IMS LD course, but a template that represents a set of abstract learning activities. Still, IMS LD is not sufficient to describe a machine-readable instructional design template since the purpose of the IMS LD is to describe concrete learning process, not an abstract template. Another approach that formalizes the representation of pedagogical patterns is described in [16]. Authors present a structured description of a pedagogical pattern. Although the pattern description is structured, it still consists of textual field, so it is not machine-readable. A term "pedagogical pattern" is in their paper used for describing a set of learning activities that are performed by a student or teacher. This definition is quite similar to the definition of the “instructional design template” used in our paper.

By analyzing instructional design description in all these papers, we have derived two conclusions about instructional design treatment: (1) instructional design is not represented as a machine readable format and/or (2) expressiveness and flexibility are not sufficient for describing a variety of instructional techniques that may be used in an e-learning course (in most papers these techniques are predefined or the underlying planning mechanisms are of general type which cumbers defining a new instructional techniques).

The system described in our paper is aimed at an automatic course generation which allows combination of pedagogical and technical aspects. The pedagogical aspect is in our system implemented using a distinct component written in our domain-specific language for describing instructional design. The proposed language enables formal representation of instructional design in a machine-readable format, independently of concrete learning goals and used learning resources and poses expressiveness and flexibility sufficient for the specific domain – instructional design. The language principles and other details are described in Section 4.3.

4. System Components

Our approach is based on a curriculum model presented in [17]. The model is based on Tyler’s taxonomy and defines four components in a course:

– learning goals,
– learning resources,
– instructional design, and
– assessment strategy.

Within the context of learning process automation, first three components are of interest. Learning goals are defined by course curriculum and they are not subject of a frequent change. For learning resources, a need for change is slightly bigger; commonly, a teacher changes some learning resources, adds new tests, etc. Finally, instructional design is the most dynamic course component in practice. In a teaching process, there is a continuous need to apply different instructional strategies for the same learning goals and the learning resources as an attempt of improving the learning process.

For the sake of flexibility in a course modification, where a teacher commonly changes only some of course’s components, we chose to define learning goals, learning resources and instructional design as three distinct components. The system organized in this way enables application of different instructional strategies for achieving defined learning goals where each strategy is consist of a particular set of learning activities. Likewise, since learning resources are defined as a separate component, the system enables simple alteration of learning resources. The result is an automatically generated IMS Learning Design compliant course which applies defined pedagogical approach. Hereby, a course creation process is easier and faster and a teacher gets a flexible environment for a course preparation. Next to it, detachment of a learning goals/content from an instructional design provides for easier testing of various instructional strategies to find the best one, as well as for creation/analysis of instructional design strategies.

4.1. Learning Goals

Learning goals can be defined as learning outcomes that a learner has to achieve. While considering a formal description of learning goals we set two basic requirements:

– the structure must be sufficiently expressive to describe learning goals and appropriate relations among them, and
– description must be machine-readable to allow automatic processing of learning goals

Following these requirements we decided to use ontology for representing learning goals. Similar approach is used in [3], [5] and [18]. Each ontology node represents one learning goal. We have defined two types of relations among our ontology nodes:

– `is-part-of`. Since learning goals are hierarchically organized, a learning goal may contain subgoals, which are defined by this relation type.
– `is-precondition`. This relation type defines a condition between learning goals: a goal c1 is a precondition for a goal c2 if, from knowing that a student has achieved c2, we can infer that he/she has achieved c1.

An ontology node can be annotated with the `hierarchical level` annotation. Predefined hierarchical levels are: `course`, `chapter`, `lesson` and `topic`. This is an optional annotation. A goal may be defined only for the purpose of grouping other learning goals. The goal doesn't necessarily represent a lesson, topic or any other course item. Furthermore, nodes can be annotated with the `orderNumber` attribute. This attribute is also not mandatory and specifies the suggested order of learning goals when the goals share the same hierarchical level and there is no relation `is-precondition` among them. Figure 2 shows the part of the learning goals ontology for the Web programming course.

4.2. Learning Resources

In this paper we use the term *learning resource* to refer any textual, graphical or multimedia digital content that a learner consumes during a learning process. For learning resources, it is necessary to specify formats of digital content, metadata and packaging. Regarding the format of digital content, although there are numerous globally used electronic formats (PDF and Office documents, images, video files), recent trends are to create learning resources in a browser readable format. The reason is that most e-learning systems are web applications, so a learner uses the internet browser to interact with the system. There are different formal specifications for defining learning resources. One widely used is IMS Content Packaging (IMS CP) [19] specification. IMS CP organizes learning resources into packages. A package consists of learning resources and a manifest file. The manifest file in XML format specifies resources, their description, organization and other elements defined by IMS CP standard. Learning resources are described by metadata and IMS consortium suggests IEEE LOM [20] specification for this purpose. Considering global acceptance of IMS CP specification, we have decided to use it to define learning resources in our system.

Learning resource is always related to one or more learning goals. A student uses a learning resource to achieve a specific learning goal. If learning resource is a test, then it is used to evaluate student's knowledge. Since learning resources and learning goals are separated in our system, it is necessary to define a component that links learning resources with learning goals. Our system contains an intermediate component that maps learning resources to the ontology of learning goals. Mapping is defined as an XML document. The XML schema of this XML document is shown in Figure 1.

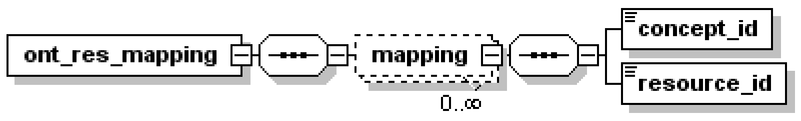


Fig. 1. The XML schema of the mapping document

Listing 1 shows a part of the mapping XML document for the Web programming course. One can notice that the learning resource "HTML tables" is related to the learning goal "Tables".

```
<ont_res_mapping  ...>
...
   <mapping>
      <concept_id>Tables</concept_id>
      <resource_id>HTML_tables</resource_id>
   </mapping>
...
</ont_res_mapping>
```

Listing 1. XML document for mapping learning goals to learning resources in the Web programming course

An example of the learning goals ontology for the Web programming course and links among learning resources and learning goals are shown in Figure 2. Within the text some names are translated to English for the sake of clarity. This applies to all figures and listings in the paper.

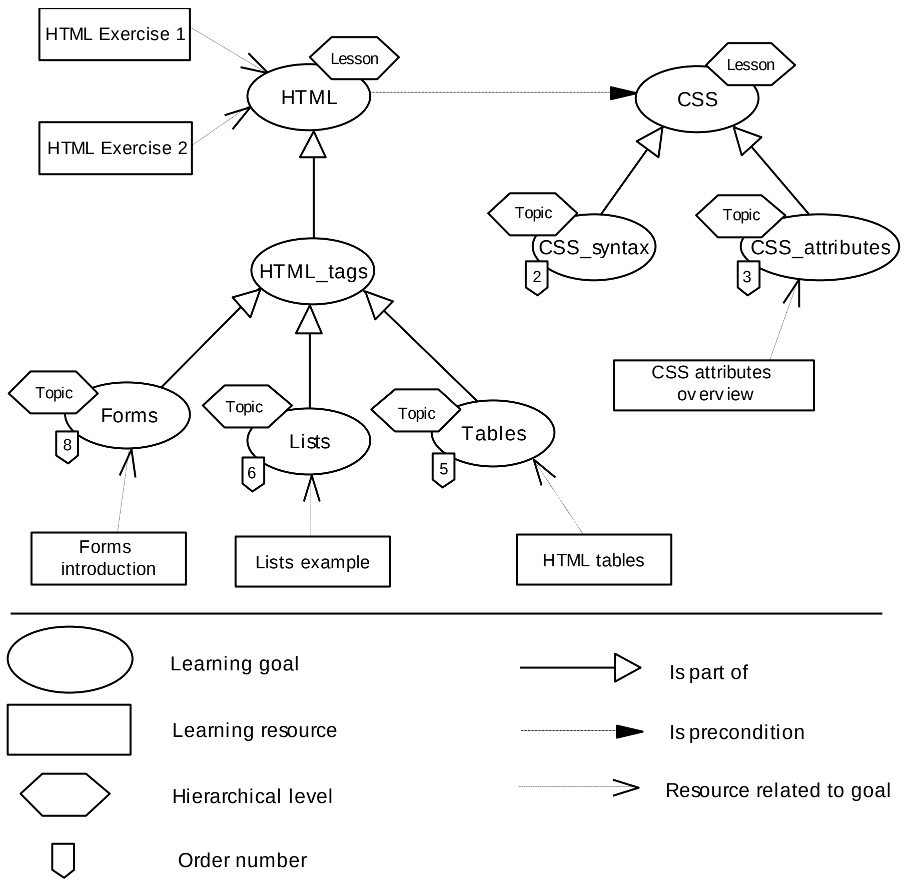


Fig. 2. Mapping of learning resources to the learning goals ontology

Using this structure, the ontology becomes an instrument that defines relations among learning resources. Thus, the relation is not defined as a part of the resource, but it is implicitly defined through the ontology. Using this approach it is easier to change the relation between two resources. In order to do that, a teacher should only map learning resource to another learning goal. Likewise, when adding a new resource, its relationships with other resources will be defined by simply mapping it to a specific learning goal(s).

4.3. E-Learning Instructional Design Language (ELIDL)

In our system, the idea is to represent instructional design detached from concrete learning goals and learning resources used for achieving these goals since the instructional design, in general, is not learning goals/resources specific [21]. Therefore, an instructional design applied to a course is represented by a detached component. The formal description of instructional design is implemented using templates that may be applied to any learning goals and learning resources. The templates are written in ELIDL (E-Learning Instructional Design Language) - our domain-specific language for describing instructional design.

ELIDL is developed following to the main principle that it should enable teacher to completely specify all components of instructional design in the e-learning environment. Thus, it is of interest to analyze how a teacher specifies instructional design in an e-learning course. Learning experience in an e-learning environment is always connected with consuming a specific digital learning resource using the computer. Learning resources can be different (a web page which explains a lesson, an internet forum for discussion with other students, an online test for knowledge evaluation, etc.), but from the technical point of view learning experience has always to do with a student's usage of some electronic resources. Therefore, instructional design in an e-learning environment is defined by the selection of appropriate learning resources and by ordering these resources. In other words, instructional design is specified by learning activities and paths through them. Given that learning activity in e-learning corresponds to the usage of some learning resource, in this paper we use the terms “learning resource” and “learning activity” interchangeably.

Willey in [22] examines instructional design for learning objects and he proposes a new instructional design theory – LODAS (Learning Object Design and Sequencing Theory). The author states that instructional design for learning objects is defined by two components:

– the scope and design of learning objects,
– the sequencing or combination of learning objects

Our research is not concerned with the scope and design of learning objects. In our work created resources are assumed as input parameters. Therefore, our language is aimed at defining the second component: the sequencing or combination of learning objects. Using ELIDL a teacher is able to formally specify instructional design that defines the order of learning activities and a criterion for selecting learning resources (number of resources, their type and priority). By this, instructional design in an e-learning course is explicitly and formally represented. Instructional design defined in ELIDL syntax is machine-readable, which is utilized here for the automatic generation of an e-learning course based on instructional design template written in ELIDL. After analyzing other languages [2, 7, 9, 15] from this domain we have decided that our language should also be XML-based.

The XML schema for ELIDL is shown in Fig. 3 and 4.

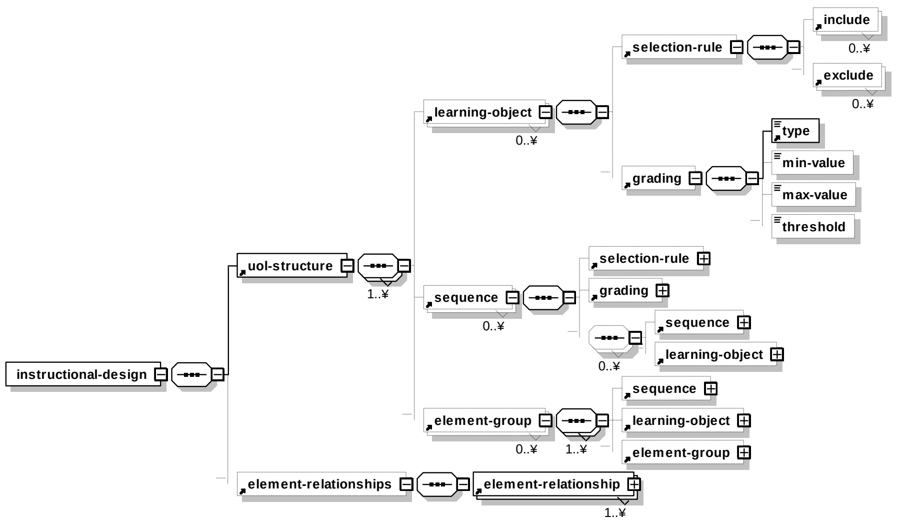


Fig. 3. The XML schema for an instructional design template – part 1

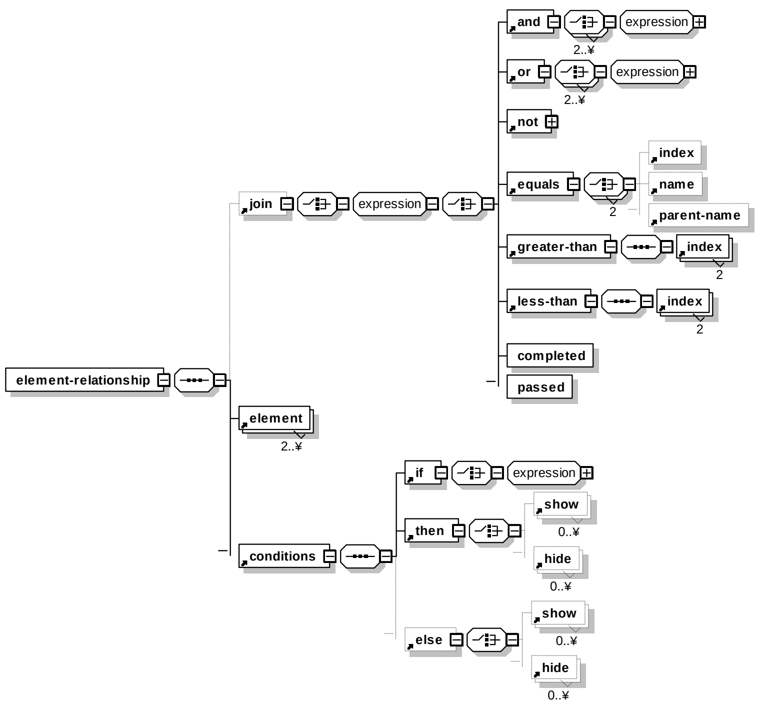


Fig. 4. The XML schema for an instructional design template – part 2

The most important schema elements are described in the following. Root element instructional-design contains two subelements:

– uol-structure - for defining learning activities in the course
– element-relationships – for defining relations among learning activities

Learning activities are defined by sequence or learning-object elements. The element sequence represents a sequence of other elements. The role of this element is similar to the role of "loop" statements in programming languages. The element sequence directs the system to pass through all learning goals annotated with hierarchical level defined in attribute element. On this way one can, for example, define the sequence of lessons, topics or learning objects. The element learning-object represents a concrete learning object in the course. Both learning-object and sequence elements contains the selection-rule element. This element specifies which learning resources and in which order will be selected from the set of learning resources mapped to a specific learning goal. The selection and priority of learning resources is defined by elements include and exclude, respectively. These elements are subelements of the selection-rule element. For the selection of a learning resource, it is necessary to define its metadata name and value. In this way our system provides the usage of any set of metadata for describing learning resources. The element grading specifies grading strategy for the learning object (the element is used for tests, projects, etc.). In order to facilitate managing learning elements, we can group them using ELIDL element element-group. The purpose of this element is to be a container for other elements.

The element element-relationships is a container for elements that define relations among learning activities in a course, e.g. there may be a strong relationship between a theory test and a project task. The project task is not available for students who didn't pass the theory test. ELIDL defines such relation using element-relationship element. Its subelements named element specifies course elements that are parts of the relation. From all specified elements, relations will be formed only for the elements that satisfy a condition defined by the join element. The element conditions has a set of if-then-else elements which specify concrete actions that will be applied to the learning elements. Currently ELIDL supports actions for displaying and hiding learning resources (ELIDL elements show and hide).

The element expression groups elements that represent a logical expression. The elements and, not and or represent logical operators: conjunction, negation and disjunction, respectively. Logical operators for comparison are defined in the elements equals, greater-than and less-than. We can define the comparison of order numbers of the learning goals (ELIDL element index). Also, the equals element can compare string values. A string value that can be used as a comparison operand is the name of the learning goal to which a learning object is connected (ELIDL element `name`). Similarly, the element `parent-name` can be used to specify the name of the parent learning goal. Completing a learning activity or passing a test is evaluated using the ELIDL elements `completed` and `passed`, respectively.

The description of all schema elements is publicly available at www.informatika.ftn.uns.ac.rs/GoranSavic/IDTemplates.

The usage of our domain-specific language is illustrated by an example presented in Listing 2.

Listing 2 shows the part of an instructional design template for describing instructional strategy called "competency assessment". This strategy is described in [23] as follows: "*The learner is first presented with the introduction (lesson overview). He is then presented with an assessment that internally evaluates the learner's mastery of each of the module objectives. The learner is presented with the instructional material (modules) related to unsatisfied objectives. After the learner has completed all the required instructional materials, an exam is presented that re-tests the objectives the learner has not satisfied.*"

The XML document shown in Listing 2 has a root element `instructional-design-template` whose attribute `root` specifies the initial node in the ontology of learning goals. Thus, learning resources that will be presented to a learner are chosen from the learning objects mapped to the subgoals of the root goal. Concretely in this example, the initial goal is the one at the highest hierarchical level annotated with "course". The first `sequence` element defines the sequence of learning goals on the hierarchical level "lesson". For each goal (lesson), we choose a learning object annotated with the label "pre-test". Each test is graded with a passed/failed mark (boolean type of grading). The second `sequence` element specifies doing the lessons once again. This time, the sequence of lesson topics is represented by the learning goals annotated with the hierarchical level "topic". For each topic, the sequence of learning objects is created. The element `selection-rule` first chooses theoretical content, then examples and all other learning resources at the end. Finally, resources annotated with the "post-test" label are added to the course. This is defined by the last `sequence` element.

As noted, this instructional strategy specifies that a learner should learn only lessons related to the learning goals unsatisfied on the pre-test or the post-test. So, we have to define the relation between the lesson, pre-test and post-test using the `element-relationship` element. The element `join` specifies that relations are created only among lessons, pre-tests and post-tests which belongs to the same learning goal. Element `if` defines the condition that a learner has to satisfy to pass the pre-test or the post-test. If they have passed one of the tests, the `then` element specifies that the corresponding lesson shouldn't be available to the learner. Otherwise, the lesson will be available which is defined in the `else` element.

```
<instructional-design root = "course">
 <uol-structure>
  <sequence element="lesson" element-id="PRETEST_LESSON_SEQUENCE">
   <learning-object element-id="PRE_TEST">
    <selection-rule>
     <include att-name="label" att-value="pre-test"/>
    </selection-rule>
    <grading>
     <type>boolean</type>
    </grading>
   </learning-object>
  </sequence>
  <sequence element="lesson" element-id="L_SEQ"
           sequence-element-id="LESSON">
   <sequence element = "topic" element-id="T_SEQ">
    <sequence element = "learning-object">
     <selection-rule>
      <include att-name="type" att-value="explanation-content"
               priority="1"/>
      <include att-name="type" att-value="example" priority="2"/>
      <include att-name="type" att-value="*" priority="3"/>
     </selection-rule>
    </sequence>
   </sequence>
  </sequence>
  <sequence element = "lesson" element-id="PT_L_SEQ">
   <learning-object element-id="POST_TEST">
    <selection-rule>
     <include att-name="label" att-value="post-test"/>
    </selection-rule>
    <grading>
     <type>boolean</type>
    </grading>
   </learning-object>
  </sequence>
 </uol-structure>
<element-relationships>
  <element-relationship>
   <element element-ref="LESSON" alias="lesson"/>
   <element element-ref="PRE_TEST" alias="pretest"/>
   <element element-ref="POST_TEST" alias="posttest"/>
   <join>
    <and>
     <equals>
      <name element="lesson"/>
      <parent-name element="pretest"/>
     </equals>
     <equals>
      <name element="lesson"/>
      <parent-name element="posttest"/>
     </equals>
    </and>
   </join>
```

```
    <conditions>
     <if>
      <or>
       <passed element="pretest"/>
       <passed element="posttest"/>
      </or>
     </if>
     <then>
      <hide element="lesson"/>
     </then>
     <else>
      <show element="lesson"/>
     </else>
    </conditions>
   </element-relationship>
  </element-relationships>
 </instructional-design>
```

Listing. 2. Instructional design template for “competency assessment” instructional strategy

5. System Architecture and Functioning

This section presents system architecture and its functioning. System components and their relations are shown in the Fig. 5.

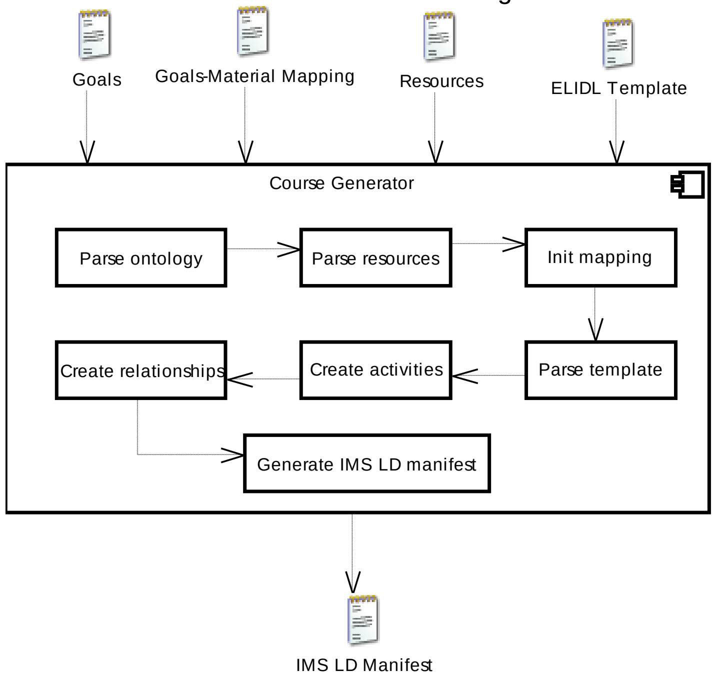


Fig. 5. System architecture

The files on the top of the figure are the input parameters to our system. The system is neutral with respect to how these files are created. In the Section 6 we have specified the software we used to create the input files. Ontology of learning goals can be created using some ontology editor, such as Protege. The goals-resources mapping is an XML file and can be created using any text or XML editor. Learning resources should be defined in the IMS CP format. There are some specialized applications, named IMS CP editors, which can facilitate this task. ELIDL template still has to be created manually using some text or XML editor, but our plan is to develop a graphical editor for creating ELIDL templates.

The system functioning is presented through the process of the course generation. Firstly, the system parses the OWL file that represents the ontology of learning goals. Then, the IMS CP manifest file which specifies learning resources is parsed. The system links resources with learning goals by parsing the XML file which defines the mapping of learning resources to the learning goals ontology. Then, the system parses ELIDL instructional design template. On the basis of the template, the system traverses the learning goals. For each learning goal, specified learning resources are taken. Then, the hierarchy of learning activities with relations among them is created. Each relation contains `if`, `then` and `else` section created according to the content of the `element-relationship` element in the template. Finally, the IMS LD compliant unit of learning is generated.

IMS LD unit of learning is generated by converting learning activities and their relations to the corresponding tags of the IMS LD manifest file. IMS LD specifies users and their roles in the manifest element `roles`. Our system creates two types of roles - `learner` for learners and `staff` for teachers within the `roles` element in the manifest file. The tag `play` is in IMS LD the root element when interpreting the learning design and it represents the flow of activities during the learning process. So, during the course generation, learning activity on the highest hierarchy level is mapped to the tag `play` in the manifest file. In IMS LD, play consists of `act` elements. In our system, tags `act` are generated on the basis of the activities on the second hierarchy level. On the third hierarchy level in IMS LD are `role-part` elements. Our system generates `activity-structure` tags for learning activities on the third hierarchy level. For each of these activities the system generates the `role-part` tag which references an appropriate `activity-structure` tag. Finally, `learning-activity` tags are created from the activities on the lowest hierarchical level, because in IMS LD `learning-activity` elements are on the lowest hierarchical level.

The flow of a learning process depends on relations among learning activities. So, relations among activities are converted to the `conditions` section of the IMS LD manifest file. Logical operators *and*, *or* and *not* are mapped to the tags with the same name in the IMS LD manifest file. ELIDL operator `completed` defines a certain action that will be performed after the completion of a learning activity. In IMS LD an activity contains the tag `on-completion` for this purpose. In this tag we can change the value of some property using the tag `change-property-value`. Properties in IMS LD are defined using tags `locpers-property`. Depending on the property value, IMS LD allows performing specific action in the `conditions` section of the manifest. So, ELIDL operator `completed` causes adding a new `locpers-property` tag within the `properties` element of the manifest. Likewise, sub tags `on-completion` and `change-property-value` are added to the `learning-activity` tags. The purpose of these elements is to define that the value of the `locpers-property` attribute will be changed if the learning activity is completed. At the end, for the defined property, the tag `greater-than` is added to the `conditions` section. Listing 3 shows a segment of the IMS LD manifest file that is generated on the basis of the template operator `completed`.

```
<imsld:locpers-property identifier="HTML_and_Java-Is-Done">
 <imsld:datatype datatype="integer"/>
 <imsld:initial-value>0</imsld:initial-value>
</imsld:locpers-property>
...
<imsld:complete-activity>
 <imsld:user-choice/>
</imsld:complete-activity>
<imsld:on-completion>
 <imsld:change-property-value>
  <imsld:property-ref ref="HTML_and_Java-Is-Done"/>
  <imsld:property-value>6</imsld:property-value>
 </imsld:change-property-value>
</imsld:on-completion>
...
<imsld:if>
 <imsld:greater-than>
  <imsld:property-ref ref="HTML_and_Java-Is-Done"/>
  <imsld:property-value>5</imsld:property-value>
 </imsld:greater-than>
</imsld:if>
```

Listing. 3. Generated segment of the manifest file for the template operator `completed`

`HTML_and_Java-is-Done` represents the property whose value will be changed on the completion of a learning activity. The initial value is 0 (defined in the `initial-value` tag). After the completion of the activity, the value is changed to 6 (in the `change-property-value` tag). A specific action will be performed if the value is greater than 5, which is defined in the `greater-than` tag. Hence, the action will be executed after the completion of the activity.

We are now going to consider the ELIDL operator `passed`. A student passes the test if he/she gets a passing grade. Therefore, it is necessary to define the grade for each operator `passed`. A new tag `locpers-property` that represents the grade is created in the `properties` section of the manifest file. In order to grade a learning activity, it is necessary to add a new web page into the unit of learning. The page has an input field for entering the grade. This page is automatically created and added to the list of learning resources. The page is created on the basis of the template grading page, part of which is shown in the Listing 4.

```
<ld:set-property ref="${PROPERTY_NAME}" property-of="self"
                view="value"/>
```

Listing. 4. Template input field for entering the grade in a grading web page

Before the grading page is added, the template field `${PROPERTY_NAME}` is substituted with the name of a concrete attribute that represents the grade. A new `greater-than` tag is added to the `conditions` sections indicating whether a student has passed the learning activitiy. Listing 5 shows the segment of the manifest file generated for the template operator `passed`.

```
<imsld:locpers-property identifier="Html_1_grading-Grade">
 <imsld:datatype datatype="boolean"/>
 <imsld:initial-value>false</imsld:initial-value>
  </imsld:locpers-property>
...
<imsld:if>
...
  <imsld:is>
   <imsld:property-ref ref="Html_1_grading-Grade"/>
   <imsld:property-value>true</imsld:property-value>
  </imsld:is>
...
</imsld:if>
```

Listing. 5. Generated segment of the manifest file for the template operator `passed`

The `locpers-property` named `Html_1_grading-Grade` represents the grade. The shown activity is graded using a boolean value – passed/not passed. In the `if` tag we evaluate this value in order to perform an action. Hence grading is done by the teacher, grading activity is defined by the tag `support-activity` in the manifest.

Regarding `if` and `then` tags in an instructional design template, the system converts them to `if` and `then` manifest tags, respectively. ELIDL actions `show` and `hide` are mapped to the manifest tags with the same name and they are added to the `conditions` section. These two tags represent actions for displaying and hiding a learning activity, respectively.

Finally, generated manifest file with all learning resources is packed into a ZIP file and this file represents the unit of learning compliant with IMS LD standard.

The system is implemented in Java programming language. Given that all input parameters are XML files, we have used Java DOM parser for parsing these files. Other technical details about the system can be found in [24].

6. Case Study

Case study shows automatic creation of the Web Programming course using our system. According to defined course curriculum, we created the ontology of learning goals in OWL [25] language using ontology editor Protege 3.4.1. Learning resources for the course are converted to HTML web pages, packed into the IMS CP package and each resource is mapped to the corresponding learning goal. In order to improve the quality of the selection of learning resources, the resources are annotated with metadata. Firstly, we describe the type of a learning resource with metadata `learningResourceType` defined by IEEE LOM specification. Values for this element are chosen from extended vocabulary defined in CLEO [26] specification which is created as an extension to the IEEE LOM. Furthermore, resources are annotated with the IEEE LOM metadata `description` that closely describes the resource.

We have created 6 instructional design templates that formally describe different instructional strategies. The templates are created for the following instructional strategies: *No sequencing*, *Linear*, *Knowledge Paced*, *Remediation*, *Competency Assessment* (strategies description adopted from [23]) and *Project-based learning* (description adopted from [27]). All templates written in ELIDL and accompanied with brief descriptions are publicly available at www.informatika.ftn.uns.ac.rs/ GoranSavic/IDTemplates/ Examples. Using these templates as inputs to our system, 6 courses of Web Programming in IMS LD format are automatically generated. These units of learning are also publicly available at www.informatika.ftn.uns.ac.rs / GoranSavic/IDTemplates/ Examples.

As an illustration, we are going to present two generated courses: for linear and knowledge paced instructional strategy. Linear strategy is described in [23] as follows "... *the learner must progress through the contents in a pre-determined order. The learner will start with the introduction first, then do all the modules and lessons in a linear order, directed by the LMS. The learner cannot proceed forward with the lessons until he has completed the current lesson. Each module is complete when he has finished all lessons in the module. The student will be presented with a comprehensive exam or quiz after he has completed all the modules*." Linear strategy for the Web programming course is presented graphically in Fig. 6.

One can notice that a student has to learn the lesson HTML firstly. After completing this lesson, he/she can learn CSS. After it, the lesson JavaScript becomes available and so on.

Using ELIDL we have written an instructional design template that formally describes this instructional strategy. Listing 6 presents a fragment of the template.

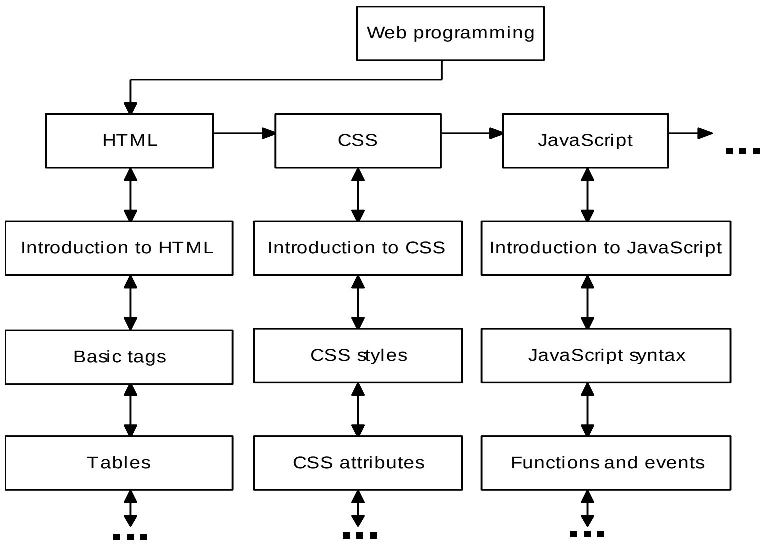


Fig. 6. Linear instructional strategy for the Web programming course

```
<uol-structure>
 <sequence element = "lesson" element-id="L_SEQ"
   sequence-element-id="LESSON">
  <sequence element = "topic" element-id = "T_SEQ"
    sequence-element-id="TOPIC">
    <sequence element = "learning-object">
     <selection-rule>
...
      <include att-name="type" att-value="*" priority="3"/>
     </selection-rule>
    </sequence>
...
<element-relationships>
   <element-relationship>
      <element element-ref="LESSON" alias="l1"/>
      <element element-ref="LESSON" alias="l2"/>
      <join>
         <less-than>
            <index element="l1"/>
            <index element="l2"/>
         </less-than>
      </join>
      <conditions>
          <if>
             <not> <completed element="l1"/> </not>
          </if>
          <then>
             <hide element="l2"/>
          </then>
          <else>
             <show element="l2"/>
          </else>
...
```

Listing. 6. ELIDL template for “Linear” instructional strategy

A student gets all learning resources in linear order. The tag `include` defines the selection of all types of learning resources. This is defined using the attributes `att-name` and `att-value`. The character * in the att-value attribute indicates the selection of all learning resources.

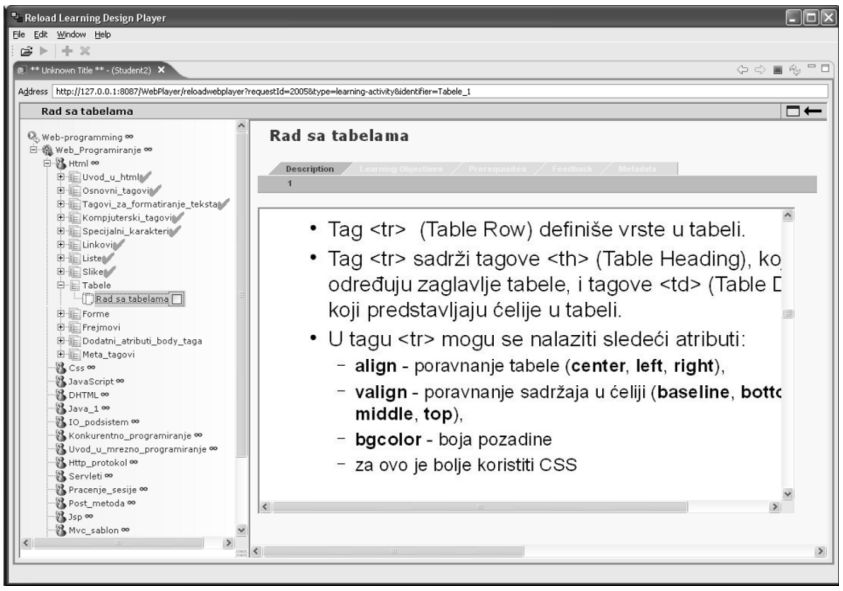
The tag `conditions` in the template document defines when the learning resource should be shown to the student. The lesson is available only when the previous lesson has been completed. Using this ELIDL template, our system has generated an IMS LD manifest file that is shown in Listing 7.

```
<imsld:learning-activity identifier="Tables_1">
 <imsld:title>Working with tables</imsld:title>
 <imsld:activity-description>
  <imsld:item identifierref="tables_res">
   <imsld:title>Working with tables</imsld:title>
  </imsld:item>
 </imsld:activity-description>
 <imsld:complete-activity>
  <imsld:user-choice/>
 </imsld:complete-activity>
 <imsld:on-completion>
  <imsld:change-property-value>
   <imsld:property-ref ref="Tables_1-Is-Done"/>
   <imsld:property-value>6</imsld:property-value>
  </imsld:change-property-value>
 </imsld:on-completion>
</imsld:learning-activity>
...
<imsld:if>
...
 <imsld:not>
  <imsld:greater-than>
   <imsld:property-ref ref="Tables_1-Is-Done"/>
   <imsld:property-value>5</imsld:property-value>
  </imsld:greater-than>
 </imsld:not>
...
</imsld:if>
<imsld:then>
 <imsld:hide>
  <imsld:activity-structure-ref ref="as_CSS_attributes"/>
 </imsld:hide>
</imsld:then>
```

Listing. 7. IMS LD manifest file generated for “Linear” instructional strategy

Listing shows a generated learning activity for learning HTML tables (learning activity with the identifier `Tables_1`). Tag `if` defines that the topic “CSS attributes” (activity structure `as_CSS_attributes`) is not available if a student hasn’t completed learning of HTML tables. The property `Tables_1-Is-Done` indicates whether the learning of the HTML tables is completed.

Figures 7 and 8 present screenshots of the Web programming course generated using ELIDL template for “linear” instructional strategy. The screenshots are taken from the Reload LD Player [28].



http://127.0.0.1:8087/WebPlayer/reloadwebplayer?requestId=2005&type=learning-activity&identifier=Tabele_1

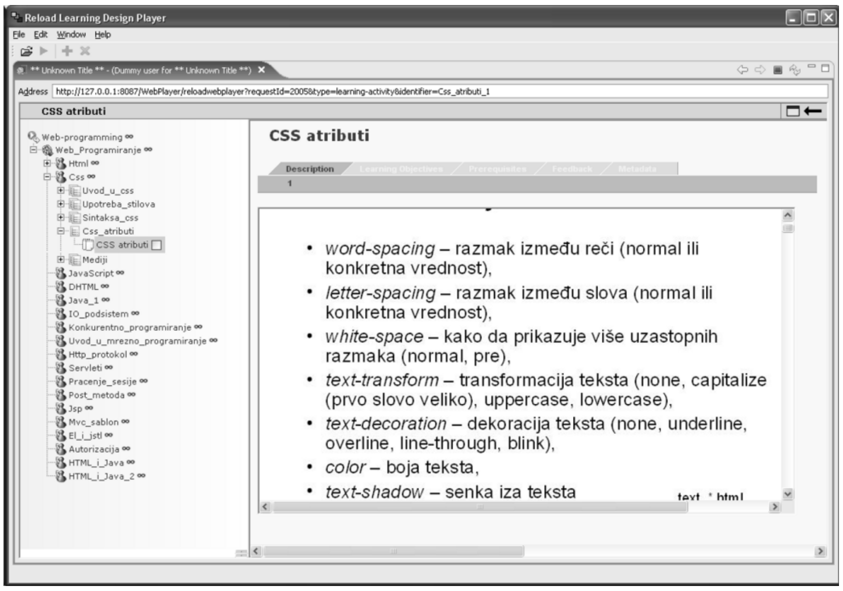
Rad sa tabelama

Web-programming
Web_Programiranje
Html
Uvod_u_html
Osnovni_tagovi
Tagovi_za_formatiranje_teksta
Kompjuterski_tagovi
Specijalni_karakteri
Linkovi
Liste
Slike
Tabele
Rad sa tabelama
Forme
Frejmovi
Dodatni_atributi_body_taga
Meta_tagovi
Css
JavaScript
DHTML
Java_1
IO_podsistem
Konkurentno_programiranje
Uvod_u_mrezno_programiranje
Http_protokol
Servleti
Pracenje_sesije
Post_metoda
Jsp
Mvc_sablon

Rad sa tabelama

- Tag <tr> (Table Row) definiše vrste u tabeli.
- Tag <tr> sadrži tagove <th> (Table Heading), ko određuju zaglavlje tabele, i tagove <td> (Table D koji predstavljaju ćelije u tabeli.
- U tagu <tr> mogu se nalaziti sledeći atributi:
  - **align** - poravnanje tabele (**center**, **left**, **right**),
  - **valign** - poravnanje sadržaja u ćeliji (**baseline**, **bott** **middle**, **top**),
  - **bgcolor** - boja pozadine
  - za ovo je bolje koristiti CSS

Fig. 7. HTML lesson in the Web programming course generated for linear instructional strategy



http://127.0.0.1:8087/WebPlayer/reloadwebplayer?requestId=2005&type=learning-activity&identifier=Css_atributi_1

CSS atributi

Web-programming
Web_Programiranje
Html
Css
Uvod_u_css
Upotreba_stilova
Sintaksa_css
Css_atributi
CSS atributi
Mediji
JavaScript
DHTML
Java_1
IO_podsistem
Konkurentno_programiranje
Uvod_u_mrezno_programiranje
Http_protokol
Servleti
Pracenje_sesije
Post_metoda
Jsp
Mvc_sablon
El_i_jstl
Autorizacija
HTML_i_Java
HTML_i_Java_2

CSS atributi

- *word-spacing* – razmak između reči (normal ili konkretna vrednost),
- *letter-spacing* – razmak između slova (normal ili konkretna vrednost),
- *white-space* – kako da prikazuje više uzastopnih razmaka (normal, pre),
- *text-transform* – transformacija teksta (none, capitalize (prvo slovo veliko), uppercase, lowercase),
- *text-decoration* – dekoracija teksta (none, underline, overline, line-through, blink),
- *color* – boja teksta,
- *text-shadow* – senka iza teksta

text . html

Fig. 8. CSS lesson in the Web programming course generated for linear instructional strategy

The screenshots show the course before and after the learning of HTML is completed. We can notice from the Figure 7 that the item "CSS" cannot be expanded and its subitems are not displayed to the user. The reason is that the lesson “CSS” is not available if the lesson “HTML” hasn’t completed. The lesson HTML is completed when all its topics are completed (as shown in the Figure 7, completed topics are checked). The topics can be completed in any order. When the user completes all topics in the lesson HTML, the lesson CSS becomes available. As we can see from the Figure 8, CSS’s topics are now displayed to the user.

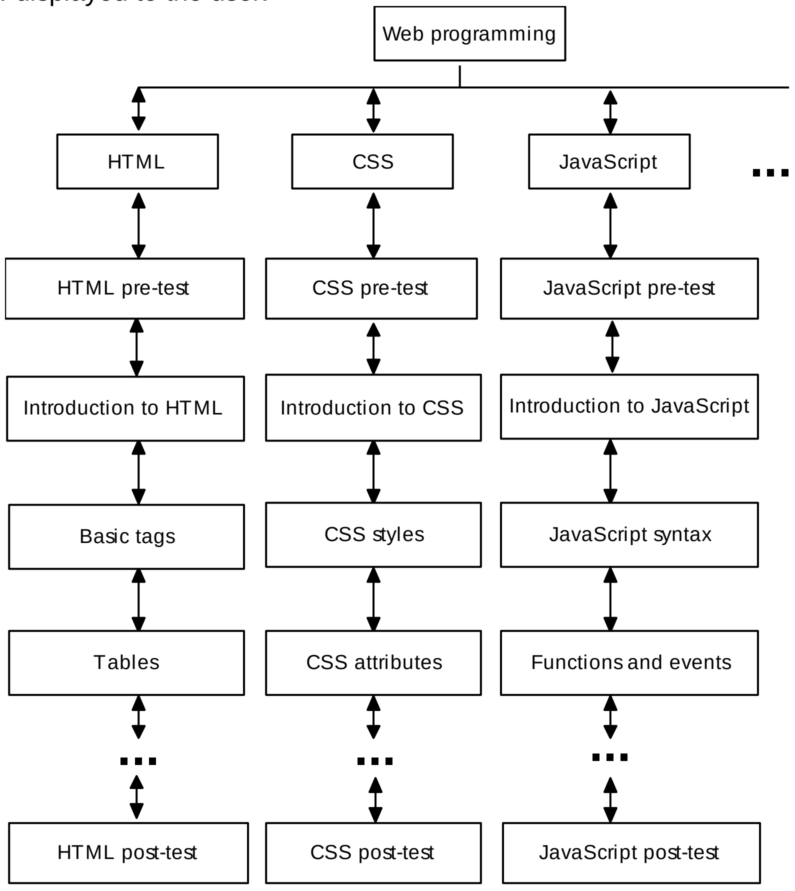


Fig. 9. “Knowledge Paced” instructional strategy for the Web programming course

The second course that will be presented here is the one generated for “Knowledge Paced” instructional strategy. This strategy is described in [23] as follows: “*In this mode, the learner must go through and complete the introduction first. After that he may proceed to the module 1 pre-test, select another module pre-test, or select a lesson. The learner may "jump" between modules, selecting pre-tests or lessons in any order. The learner cannot select the Module post-tests. These are only encountered after the learner "flows" through the modules lessons. If the learner passes an exam (pre- or post-), the module’s learning objective has been satisfied and the module’s* *post-test becomes disabled. The learner may continue to select individual lessons for the duration of the course, even after a module's objective has been satisfied. If the learner does not pass the exam, the learner is directed to that module's instructional content, and once completed, must retake the module exam (post-test).*" Figure 9 shows "Knowledge Paced" instructional strategy for the Web programming course.

In contrast to "linear" instructional strategy, in "knowledge paced" strategy a student is not limited only to linear path. He/she may arbitrary choose next learning unit. A learning unit is satisfied if a student passes the unit pre-test or post-test. Listing 8 shows ELIDL template created for this instructional strategy.

```
<uol-structure>
...
 <sequence element = "lesson" element-id="L_SEQ">
  <learning-object element-id="PRE_TEST">
   <selection-rule>
    <include att-name="label" att-value="pre-test"/>
   </selection-rule>
...
  <sequence element = "topic" element-id="T_SEQ">
   <sequence element = "learning-object">
    <include att-name="type" att-value="*" priority="3"/>
...
  <learning-object element-id = "POST_TEST">
   <selection-rule>
    <include att-name="label" att-value="post-test"/>
   </selection-rule>
...
<element-relationships>
 <element-relationship>
  <element element-ref="PRE_TEST" alias="pre"/>
  <element element-ref="POST_TEST" alias="post"/>
  <join>
   <equals>
    <parent-name element="pre"/>
    <parent-name element="post"/>
   </equals>
  </join>
  <conditions>
   <if>
    <or>
     <passed element="pre"/>
     <passed element="post"/>
    </or>
   </if>
   <then>
    <hide element="pre"/>
    <hide element="post"/>
   </then>
   <else>
    <show element="pre"/>
    <show element="post"/>
   </else>
  </conditions>
 </element-relationship>
</element-relationships>
```

Listing 8. ELIDL template for "Knowledge Paced" instructional strategy

As we can see, the template iterates through all lessons (ELIDL element `sequence`). For each lesson it firstly selects a pre-test using the tag `include` and its attribute `att-value` whose value is “`pre-test`”. After the pre-test, in the second `sequence` element, the template iterates through lesson topics and their learning resources. It selects all learning resources in the lesson (the attribute `att-value` in the tag `include` has the value “*”). At the end, a user gets a post-test in a similar way as for the pre-test.

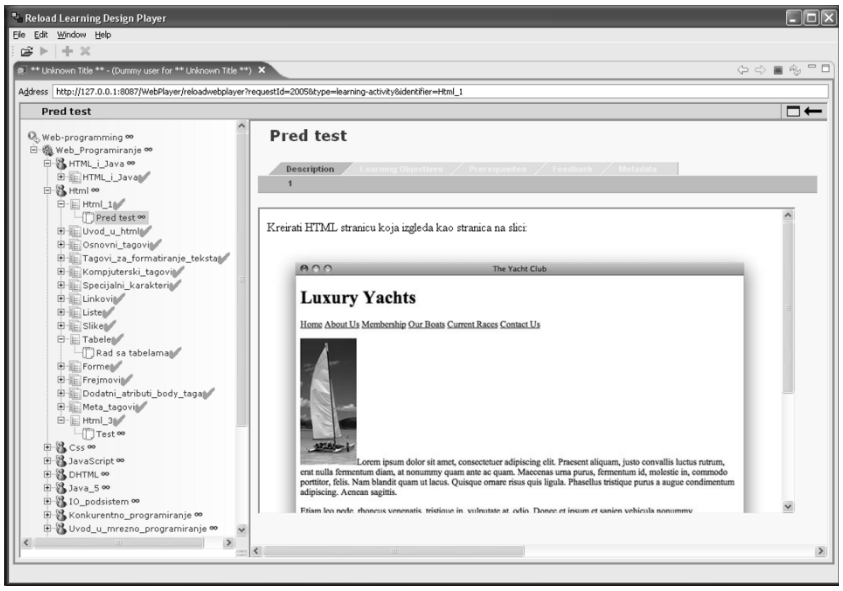
The tag `element-relationship` defines a relation between pre-test and post-test. If a student passes pre-test or post-test (defined in `passed` ELIDL tags), both tests become unavailable (ELIDL elements `hide`). Otherwise, the tests are available which is defined in the ELIDL elements `show`. Listing 9 shows a part of the generated IMS LD manifest file for “Knowledge Paced” instructional strategy.

```
<imsld:learning-activity identifier="Html_3">
 <imsld:title>Post test</imsld:title>
 <imsld:activity-description>
  <imsld:item identifierref="html_posttest_res">
   <imsld:title>Post test</imsld:title>
  </imsld:item>
 </imsld:activity-description>
</imsld:learning-activity>
...
<imsld:activity-structure identifier="as_Html_3"
                         structure-type="sequence">
 <imsld:title>Html_3</imsld:title>
 <imsld:learning-activity-ref ref="Html_3"/>
</imsld:activity-structure>
...
<imsld:if>
...
 <imsld:or>
  <imsld:is>
   <imsld:property-ref ref=" Html_1_grading-Grade"/>
   <imsld:property-value>true</imsld:property-value>
  </imsld:is>
  <imsld:is>
   <imsld:property-ref ref=" Html_3_grading-Grade"/>
   <imsld:property-value>true</imsld:property-value>
  </imsld:is>
 </imsld:or>
...
</imsld:if>
<imsld:then>
 <imsld:hide>
  <imsld:activity-structure-ref ref=" as_Html_3"/>
 </imsld:hide>
</imsld:then>
```

Listing. 9. IMS LD manifest file generated for “Knowledge Paced” strategy

A post-test for lesson HTML is represented with a learning activity “Html_3” and a corresponding `activity-structure` element. The post-test should be hidden if a student has passed the pre-test or post-test. The attributes “Html_1_grading-Grade” and “Html_3_grading-Grade” represent grades on the pre-test and post-test, respectively. The test grade is a logical value –

true/false (passed/failed). When the grades values are `true` (defined in the IMS LD tag `if`), the post-test is hidden (defined using IMSL LD element `hide`). Figure 10 shows the screenshot of the Web Programming course generated from “Knowledge Paced” ELIDL template. The screenshot is taken before the pre-test for the lesson “HTML” is passed. We observe appearance of the items Html_1 and Html_3 inside the lesson HTML. These items represent pre-test and post-test, respectively. Fig. 11 shows the screenshot of the same course after the pre-test is passed. One can notice that the Html_1 and Html_3 items do not appear in the lesson, because the pre-test and post-test are hidden now.



Address http://127.0.0.1:8087/WebPlayer/reloadwebplayer?requestId=2005&type=learning-activity&identifier=Html_1

Kreirati HTML stranicu koja izgleda kao stranica na slici:

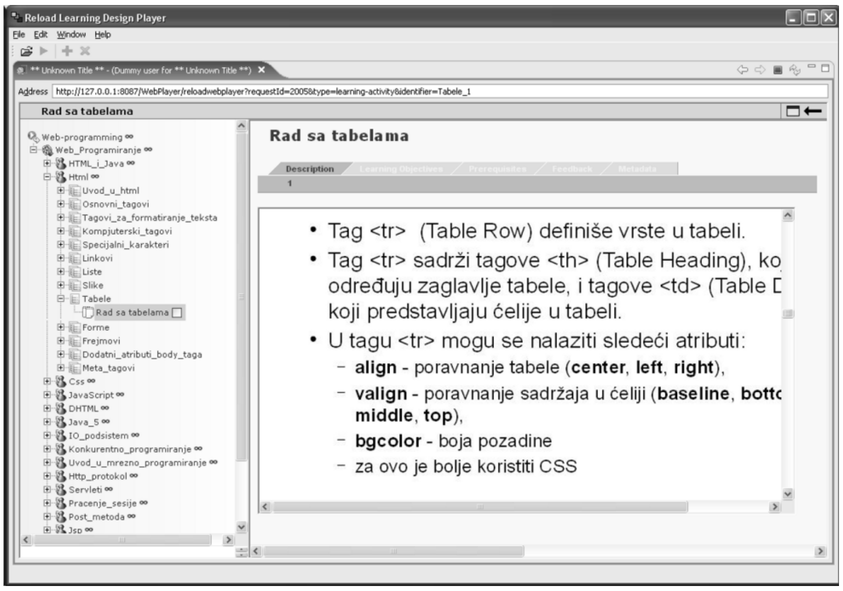
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Praesent aliquam, justo convallis luctus rutrum, erat nulla fermentum diam, at nonummy quam ante ac quam. Maecenas urna purus, fermentum id, molestie in, commodo porttitor, felis. Nam blandit quam ut lacus. Quisque ornare risus quis ligula. Phasellus tristique purus a augue condimentum adipiscing. Aenean sagittis.

Etiam leo pede, rhoncus venenatis, tristique in, vulputate at, odio. Donec et ipsum et sapien vehicula nonummy

Fig. 10. HTML lesson in the Web programming course generated for “Knowledge Paced” instructional strategy (before the pre-test is passed)

All 6 generated units of learning for Web Programming course are analyzed and the results are presented in Table 1. For each unit of learning, the table shows applied instructional strategy and the following characteristics of the manifest file: number of learning activities, relations, `locpers-property` tags and lines of XML code.

Three of the six instructional strategies are described earlier in the text: *Competency Assessment, Linear and Knowledge Paced*. Before the analysis of the generated courses, we are going to describe briefly three other strategies. *No-sequencing* is the simplest strategy and it allows unrestricted access to all learning resources in any order. *Remediation* strategy is the combination of the *Linear and Knowledge Paced strategy.* A learner has to learn lessons in linear order. For each lesson there is a post-test (there is no pre-tests as in case of *Knowledge Paced* strategy). If a student passes the post-test, the lesson is completed and he/she doesn't need to learn the lesson any more. *Project-based learning* strategy is directed to create a specific product as the result of the learning process. A student is free to explore all learning content in any order without restrictions. During the course there are milestones for evaluating the progress of the project. Each milestone defines a project-part which is evaluated.



Rad sa tabelama

- Tag <tr> (Table Row) definiše vrste u tabeli.
- Tag <tr> sadrži tagove <th> (Table Heading), ko određuju zaglavlje tabele, i tagove <td> (Table D koji predstavljaju ćelije u tabeli.
- U tagu <tr> mogu se nalaziti sledeći atributi:
  - **align** - poravnanje tabele (**center**, **left**, **right**),
  - **valign** - poravnanje sadržaja u ćeliji (**baseline**, **bottc** **middle**, **top**),
  - **bgcolor** - boja pozadine
  - za ovo je bolje koristiti CSS

Fig. 11. HTML lesson in the Web programming course generated for "Knowledge Paced" instructional strategy (after the pre-test is passed)

Table 1. Comparison of the generated units of learning

The strategy applied	# of				Processing time
	Activi ties	Relations	locpers-property	Manifest lines	
No-sequencing	85	0	2	1942	2 s
Linear	84	138	83	47109	219 s
Knowledge Paced	131	210	95	7663	6 s
Remediation	107	186	94	58332	456 s
Competency Assessment	131	210	104	19673	22 s
Project-based learning	88	170	7	5305	4 s

The strategies *No-sequencing* and *Linear* are characterized by a small number of learning activities because they contain only learning the lessons and work on a project. Since the students have to take pre-tests and post-tests, *Knowledge-paced* and *Competency Assessment* strategies have the largest number of activities. Regarding the number of relations, the *No-sequencing* strategy has no relation at all, as it means a completely free learning path. The largest numbers of relations have *Knowledge Paced* and *Competency Assessment* strategies since they have the largest number of activities and these activities are conditioned by each other (availability of some activities depends on the result of completed and passed actions of other activities). Elements `locpers-property` comply with the number of relations among learning activities. *Competency Assessment* and *Knowledge Paced* strategies have the same number of relations, but for *Competency Assessment* strategy three activities participate in some relations, while for *Knowledge Paced* strategy all relations are defined between only two activities. Therefore, the generated manifest file has more `locpers-property` elements for the *Competency Assessment* instructional strategy. The *No-sequencing* strategy doesn't have any relations, so it has only two `locpers-property` elements. These two elements represent the student's grades on the projects and they are the consequences of the grading element in the instructional design template. The least lines of XML code is generated for *No-sequencing* strategy, since this strategy has a small number of activities and there are no relations in it. For this strategy, the instructional design template doesn't define any condition element at all. Manifests for the *Linear* and *Remediation* strategies have most lines of code, since these strategies require linear path through lessons. This means that a lesson is not available until the previous one is completed which causes many `if-then` elements in the manifest. In addition, the *Remediation* strategy contains tests, which produces a manifest file containing most lines of code. Regarding the processing time, it is proportional to the number of the generated lines in the manifest. *No-sequencing* strategy requires least time, while the most time the system spends to generate a course based on the *Remediation* strategy.

An automatically created course can be imported into some IMS LD Player. For our purposes, we have used Reload LD Player [28] and CopperCore IMS Learning Design Engine [29].

7. Conclusion

The paper presents the system for automatic generation of IMS LD compliant courses based on explicitly expressed machine-readable instructional design templates. The courses are generated from three inputs: the ontology of learning goals, learning resources and instructional design template. The output result is an IMS LD unit of learning that applies defined instructional design.

Instructional design templates provide the means for explicit and formal specification of instructional design, which most of existing systems for the automatic course generation do not support. Detaching instructional design eases change of an applied instructional strategy in the course. For the purpose of formal templates' specification and representation, we have developed a new domain-specific language named ELIDL.

It should be mentioned that the generated course is a static one – it represents the sequence of predefined learning activities. The quality of the teaching process and utilization of an e-learning environment could be improved if a learning activity is chosen in real-time. These would enable that a student gets a learning activity dynamically depending on various parameters (instructional design, student's knowledge state, personal preferences etc.). However, given that most current LMS's support only static courses, in this phase of the research we decided to generate only a static sequence of learning activities.

The system is verified on the example of generating Web Programming course at the Faculty of Technical Sciences in Novi Sad. For the defined learning goals and learning resources, we created 6 instructional design templates that describe following instructional strategies: *No sequencing*, *Linear*, *Knowledge Paced*, *Remediation*, *Competency Assessment* and *Project-based learning*. As a result, the system generated 6 Web Programming courses compliant with IMS LD specification. All templates and generated courses are publicly available.

Our future work is mainly concerned with using the generated courses in a teaching process. Currently we are measuring students' achievements and motivation in order to evaluate different instructional strategies and to determine which one is the best for the given course.

Another direction of the further research is oriented towards development of the software tool aimed at assisting teachers in instructional design specification. This tool will enable simple and intuitive specification of instructional design templates. There are ongoing activities aimed at creating a graphical editor for ELIDL that would enable creating templates without knowing ELIDL syntax. This editor should be only a part of the integrated GUI application that we are planning to develop. This application will provide a graphical interface for our system. Currently, our system is used from the command line.

Although in our course some activities are done by students and some by teachers, our system doesn't have a component for an explicit specification of users and their roles. Our long-term goal is to create such a component. It would enable collaborative learning, which is not currently supported.

Acknowledgements. Results presented in this paper are part of the research conducted within the Grant No. III-47003 financed by the Ministry of Education and Science of the Republic of Serbia.

References

1. Advanced Distributed Learning (ADL): SCORM 2004 4th edition - Sequencing and Navigation Version 1.1 (2009). [Online]. Available: www.adlnet.gov (current June 2011)
2. IMS Global Learning Consortium: IMS Learning Design Information Model (2003). [Online]. Available: www.imsglobal.org/learningdesign/ldv1p0 /imsld_infov1p0.html (current June 2011)
3. Kontopoulos, E., Vrakas, D., Kokkoras, F., Bassiliades, N., Vlahavas, I.: An Ontology-based Planning System for e-Course Generation, Expert Systems with Applications, Vol. 35, No. 1-2, 398-406. (2008)
4. Hernandez, J., Baldiris, S., Santos, O. C., Fabregat, R., Boticario, J. G.: Conditional IMS Learning Design Generation using User Modeling and Planning Techniques. In Proceedings of the 9th IEEE International Conference on Advanced Learning Technologies, IEEE Computer Society, Riga, Latvia, 228-232. (2009)
5. Capuano, N., Gaeta, M., Micarelli, A., Sangineto, E.: An Integrated Architecture for Automatic Course Generation. In Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT 02), Kazan, Russia, 322-326. (2002)
6. Morales, L., Castillo, L., Fernandez-Olivares, J., Gonzalez-Ferrer, A.: Automatic Generation of User Adapted Learning Designs: An AI-Planning Proposal. In Proceedings of the 5th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Springer-Verlag, Hannover, Germany, 324-328. (2008)
7. Arapi, P., Moumoutzis, N., Mylonakis, M., Theodorakis, G., Christodoulakis, S.: A Pedagogy-Driven Personalization Framework to Support Adaptive Learning Experiences. In Proceedings of the 7th IEEE International Conference on Advanced Learning Technologies (ICALT 2007), Nigata, Japan, 96-97. (2007)
8. Pacurar, E. G., Trigano, P., Alupoaie, S.: Knowledge base for automatic generation of online IMS LD compliant course structures. Educational technology & Society, Vol. 9, No. 1, 158-175. (2006)
9. LAMS: Learning Activity Management System. [Online]. Available: www.lamsinternational.com (current June 2011)
10. The Pedagogical Patterns Project. [Online]. Available: www.pedagogicalpatterns. org (current June 2011)
11. McAlpine, I., Allen, B.: Designing for active learning online with learning design templates. In ICT: Providing choices for learners and learning, Proceedings ascilite Singapore 2007, Centre for Educational Development, 639-651. (2007)
12. Heathcote, E. A.: Learning design templates – a pedagogical just-in-time support tool. In: Minshull, Geoff & Mole, Judith (Eds.) Designing for Learning, JISC Development Group, Bristol, UK, 19-26. (2006)
13. Abbing, J., Koidl, K.: Template Approach for Adaptive Learning Strategies. In Proceedings of Adaptive Hypermedia, Dablin, Ireland. (2006)
14. Guangzuo, C., Xinqi, R., Haitao, Z., Ronghuai, H.: SMID: A Semantic Model of Instructional Design. In Proceedings of the 2009 First International Workshop on Education Technology and Computer Science, Vol. 3, IEEE Computer Society, 130-134. (2009)
15. de Moura Filho, C.O., Derycke, A., Pedagogical Patterns and Learning Design: When Two Worlds Cooperate. In Proceedings of the UNFOLD-PROLEARN Joint Workshop, Valkenburg, The Netherlands. (2008)

16. Ljubojevic, D., Laurillard, D., A theoretical approach to distillation of pedagogical patterns from practice to enable transfer and reuse of good teaching. In Proceedings of the 2010 European LAMS & Learning Design Conference, Oxford, UK. [Online]. Available: http://lams2010.lamsfoundation.org/papers.htm (current June 2011)
17. Segedinac, M., Savić, G., Konjović, Z.: Knowledge representation framework for curriculum development. In Proceedings of the International Conference on Knowledge Engineering and Ontology Development, Valencia, Spain. (2010)
18. Segedinac, M. T., Konjović, Z., Segedinac, M. D., Savić, G.: A formal approach to organization of educational objectives. Psihologija, In press. (2011)
19. IMS Global Learning Consortium: IMS Content Packaging Specification v1.2 (2007). [Online]. Available: www.imsglobal.org/content/packaging (current June 2011)
20. IEEE: IEEE 1484.12.1-2002 Learning Object Metadata Standard (2002). [Online]. Available: http://ltsc.ieee.org/wg12 (current June 2011)
21. Hlebowitsh, P. S.: Designing the School Curriculum. Pearson Education, Inc., USA. (2005)
22. Wiley, D. A., Learning object design and sequencing theory, PhD dissertation, Department of Instructional Psychology and Technology, Brigham Young University (2000). [Online]. Available: www.opencontent.org/docs/dissertation .pdf (current June 2011)
23. Chew, L. K., Hua, T. G., Instructional Strategies and Limitations of the SCORM 2004, In Proceedings of the 16th International Conference on Computers in Education (ICCE 2008), Taipei, Taiwan, 153-160. (2008)
24. Savić, G., Segedinac, M., Konjović, Z.: The Implementation of the IMS LD E-course Generator. In Proceedings of the 1st International Conference on Information Society, Technology and Management (ICIST 2011), Kopaonik, Serbia (2011)
25. W3C: OWL Web Ontology Language Overview (2004). [Online]. Available: www.w3.org/TR/owl-features (current June 2011)
26. CLEO Lab: CLEO Extensions to the IEEE Learning Object Metadata (2003). [Online]. Available: www.oasis-open.org/committees/ download.php/20490 /CLEO_ LOM_Ext_v1d1a.pdf (current June 2011)
27. Derntl, M., Motschnig-Pitrik, R., Patterns for Blended, Person-Centered Learning: Strategy, Concepts, Experiences, and Evaluation. In Proceedings of the 2004 ACM symposium on Applied computing, ACM, Nicosia, Cyprus, 916-923. (2004)
28. Reload: Learning design player v. 2.1.3 (2010). [Online]. Available: www.reload.ac.uk/ldeditor.html (current June 2011)
29. CopperCore: The IMS learning design engine v 3.3 (2008). [Online]. Available: http://coppercore.sourceforge.net (current June 2011)

Goran Savić graduated from the University of Novi Sad, Faculty of Sciences, in 2006. He is currently on PhD studies in Computer Science, from the University of Novi Sad, Faculty of Technical Sciences where he is a teaching assistant. He has authored papers in international and national journals and conferences. His research interest is e-learning.

Milan Segedinac was born in Novi Sad, on May 28, 1984. He received his M.Sc. degree from the Faculty of Technical Sciences at the University of

Novi Sad in 2008, where he entered the Ph.D. studies the same year. He has been employed at the Faculty of Technical Sciences since 2011 as a teaching assistant. His research interests are in the area of computer-enhanced education. He has authored papers in international and national journals and conferences.

Zora Konjović received her Bachelor degree in Mathematics from the Faculty of Natural Science Novi Sad (in 1973), the Master degree and PhD (both in Robotics, in 1985. and 1992 respectively) from the Faculty of Technical Sciences Novi Sad. She is a full professor at the Faculty of Technical Sciences, Novi Sad, Serbia since 2003. Prof Konjović participated in 30 research projects (as the project leader in 18). She published more than 180 scientific and professional papers. Her current research interests include artificial intelligence, web programming, digital libraries and archives, and geo-informatics.

Received: June 15, 2011; Accepted: February 21, 2012.