2009 Ninth International Conference on Intelligent Systems Design and Applications

LS-LAB: a Framework for Comparing Curriculum Sequencing Algorithms

Carla Limongelli, Filippo Sciarrone, Giulia Vaste
Department of Computer Science and Automation - AI Lab
Roma Tre University
Rome, Italy
{limongel,sciarro,vaste}@dia.uniroma3.it

***Abstract*—Curriculum Sequencing is one of the most appealing challenges in Web-based learning environments: the success of a course mainly depends on the system capability to automatically adapt the learning material to the student's educational needs. Here we address the problem of how to compare and to test different Curriculum Sequencing algorithms in order to reason about them in a self-contained and homogeneous environment. We propose LS-LAB, a framework especially designed for comparing and testing different Curriculum Sequencing algorithms. LS-LAB has been designed to run different algorithms, each of them provided with its own Student Model representation: a Super Student Model is able to incrementally include all of them. In this framework, the Learning Node has to be compliant to the IEEE LOM specifications, while, through a suitable GUI, one can insert new algorithms or run already available ones. We are carrying out the implementation by using a 3-tier Java application technology, in order to make this environment available on the Internet. Finally we show an application example.**

***Index Terms*—curriculum sequencing; student modeling;**

I. MOTIVATIONS AND GOALS

The rapid growth of the Internet is enabling a more widespread use of distance learning based on Web-oriented systems such as Web-based Educational Hypermedia and Learning Management Systems. In this context, the pedagogical strategy behind a course is crucial, such as for example the capability of a system to tailor the course to the needs and interests of each individual student. In fact, Personalization and Adaptation, as opposed to the traditional *one-size-fits-all* approach, are more and more sought in educational systems [6].

Curriculum Sequencing is one of the most interesting challenges in educational research area: research in this field aims to automatically produce a personalized sequence of didactic materials or activities, on the basis of each student's needs, by dynamically selecting the most appropriate didactic materials at any moment [7]. Several approaches addressing this issue have been proposed in the literature: rule-based sequencing, as in the AHA! system [4]; planning-based sequencing, as in the LS-PLAN system [11] and in the work of Baldoni et al. [1]; graph-based sequencing as in the IWT system [14]; KBS-Hyperbook system [9], Lecomps system [15] and in DCG system [10].

In this paper we present the LS-LAB framework, an integrated environment, with the aim to give researchers an instrument for quickly comparing and testing different Curriculum Sequencing algorithms. The motivation behind this effort arise from the fact that, while from one hand the number of proposals in this field is increasing, on the other hand, presently, there is a lack of such an environment where one can actually test and compare different Curriculum Sequencing algorithms. In fact, to really compare them, they need to start from the same conditions, running on the same learning material. Moreover, our system aims to provide researches with an almost *ready-to-use* environment, allowing for a *low-cost* experimentation. In the literature there are a number of proposals of frameworks designed for evaluating and testing different kinds of algorithms. For example in [19] a framework for evaluating the performance of user modelling servers is presented. Weibelzahl and Lauer presented a framework to evaluate different adaptive Case-Based Reasoning systems [17], while in the work of Baldoni et al. a framework for comparing algorithms in Adaptive Educational Hypermedia is presented [2]. In the LS-LAB system different sequencing algorithms belonging to different adaptive educational environments are involved. These algorithms, through suitable software interfaces, e.g. parsers, run in the same environment.

This paper is structured as follows: in Section II the LS-LAB system, with its components and procedures is shown. In Section III is reported an example of application, based on two sequencing approaches: the first one is based on a topological sorting, while the second one is based on automated planning. In Section IV conclusions are drawn.

II. THE LS-LAB SYSTEM

As pointed out in [5], to produce adaptive educational courses we need to design the domain model, i.e. the knowledge space covered by the course, and its connection with the learning materials. Moreover it is necessary to sequence these entities in a personalized way, taking into account the Student Model (SM). All these components, i.e., the domain model, the SM, the selection of concepts and didactic materials and their sequencing, are managed in different ways in the literature. Sequencing is generally performed by following two main approaches: sequencing given step-by-step to students, through techniques such as adaptive link annotation and direct guidance [6] (like in the AHA! System [4] and in the ELM-ART system [16]), and sequencing that plans the entire

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.146

learning path at the beginning, then modifying it, when the study does not succeed as it should (like in the work of Baldoni et al. [1], [3] and in the IWT system [14]). Hence, the main issue concerning the design of such an environment is the heterogeneity of the different sequencing techniques, each of them with their own particular SM representation, together with their different Domain Knowledge (DK) that have to be uniformly parsed and represented in order to have a homogeneous environment. Other problems concern the building of all those software interfaces between algorithms and the technical environment, necessary to correctly run them.

A. The Learning Environment Representation

In the following we show the architecture of the common learning environment where researchers can run their algorithms. First we give some definitions and after we make some working assumptions.

Definition 1: Knowledge Item (KI). A KI is an atomic element of knowledge about a given learning topic.

Definition 2: Knowledge Domain (KD). It is the set of all the KIs related to a particular course: $KD = \{KI_1, KI_2, \ldots, KI_n\}$.

Definition 3: Learning Node (LN). A LN is a 3-tuple:

$$LN = \langle LM, AK, RK \rangle \quad \text{where}$$

LM is the Learning Material, i.e., any instructional digital resource.

AK Acquired Knowledge. It is a KI that represents the concept that has to be acquired after having taken the contents related to the given LN.

RK Required Knowledge. It is the set of KI necessary for studying the material of the node, i.e., the cognitive prerequisites required by the contents of the associated LN.

Definition 4: Course. A Course is a particular set of Learning Nodes (LN), created by the teacher about a particular topic.

Definition 5: Starting Knowledge (SK). The SK for a given course is a subset of KIs representing the knowledge that the student is supposed to have, before starting the course.

Definition 6: Target Knowledge (TK). The TK for a given course is a subset of KIs representing the knowledge to be acquired by the student after having taken the course.

Definition 7: Learning Object Sequence (LOS). It is the sequence of LN selected by a given sequencing algorithm.

All algorithms to be inserted into the system, should satisfy the following minimal requirements about their working domain representation to be mapped into the LS-LAB learning environment:

- to be *goal driven*, i.e., to allow the learner for acquiring a TK,
- to be able to manage KIs,
- to be able to manage IEEE-LOM compliant LNs,
- to be able to manage, for each LN, its RK and the AK.

Finally, we assume that each LN in a course should be compliant to the IEEE-LOM specifications, in order to make, in the future, the system compliant with Learning Management Systems. To this purpose, here, we briefly show how a LN is mapped into some suitable IEEE-LOM metadata. For example the contents of the fields $<RK>$ and $<AK>$ are represented by means of the tag <purpose> of the last IEEE-LOM category, <classification>; each element of the <taxonPath>, <taxon>, is composed by an identification tag <id> and by a string <string>, that represents the name of a prerequisite, or of an AK. In any case, researchers that want to compare their sequencing algorithms to other ones in LS-LAB, have to build LNs compliant to the IEEE-LOM specifications and they have to provide or insert the executable files or source code. Moreover, a set of parsers should be built for data mapping. We made the aforesaid assumptions just to start with an environment that is as standard as possible and, therefore, compliant with most common learning platforms.

B. The Super Student Model

Each sequencing algorithm has its own SM representation. In such a general system, it would be a very hard problem to build in advance a general SM, i.e., a SM that contains all possible SM representations. Hence, in LS-LAB we introduce the *Super Student Model* (SSM) that is a general SMs container gradually filled by researchers as they insert a new algorithm, together with the related SM, into the system. The increasing law is the following:

$$SSM_i = SSM_{i-1} \cup SM_i \tag{1}$$

being SM_i the SM related to the new sequencing algorithm A_i to be inserted into the system, while SSM_{i-1} and SSM_i are the SSM before and after the new A_i insertion, respectively. In Section III we show, in a real application, the SSM increasing mechanism in detail for the insertion of two new algorithms into the system. With this type of representation, the SSM starts as an empty set: $SSM_0 = \emptyset$.

C. The System Architecture

In this Subsection we show the LS-LAB system in all its logical and functional components, shown in the Fig. 1 where dashed arrows represent the input given by the researcher. The system is composed of the following functional modules:

- *GUI*. It is the graphical environment composed of five components. Through the *insert your data* component the researcher can input the personal data of the student, if any. Through the *Course Information* component one can select the course of interest, while through the *Starting Knowledge Selection* and the *Target Knowledge Selection* components the researcher can input respectively the Student Starting Knowledge and the Course Target Knowledge. Finally, through the *Algorithm Selection* component

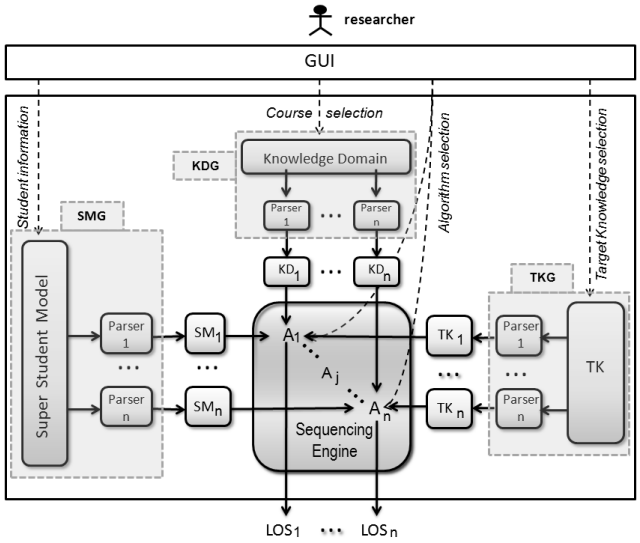

Fig. 1. The Functional Schema of the LS-LAB System.

the algorithm currently present in the environment can be selected.

- *Student Model Generator* (SMG). This module generates the particular SM_i related to the particular algorithm A_i to be run. Firstly, the SMG takes in input the researcher's SM selection coming from the GUI, among all the SMs contained in the current *Super Student Model* (SSM). Secondly, it runs the $Parser_i$ producing as output the SM_i, i.e., that particular SM representation needed by the algorithm A_i.
- *Knowledge Domain Generator* (KDG). This module takes as input the $Course_i$ selected by the researcher giving as output the KD_i related to that course, by launching the right parser in order to obtain the right KD representation adapted to be managed by the algorithm A_i.
- *Target Knowledge Generator* (TKG). This module takes as input the TK selected by the researcher, and consequently, through a suitable parser, produces as output the right TK_i representation ready to be managed by the A_i algorithm.
- *Sequencing Engine* (SE). This module is the core of the system. After having taken as input the sequencing algorithm selection, for example A_i, among all those algorithms currently present in the system, it runs A_i on the 3-ple $I =< TK_i, KD_i, SM_i >$.

D. The System Workflow

LS-LAB can be used in two different scenarios. In the first one a researcher can run two or more algorithms currently present in the system, while in the second one she can insert a new algorithm in the system comparing it with other available algorithms. The insertion of a new algorithm needs the effort to make uniform its associated SM, its KD, and its TK representations to the LS-LAB standard. This effort means that we need to receive information about the algorithm input data, to provide the algorithm executable file, and to develop the necessary parsers and adapters for SM, KD, and TK. In this way a researcher, who wants to insert her algorithm into the LS-LAB system, has to provide the LS-LAB developers with:

- the algorithm executable file;
- the description of the algorithm input data;
- an *xml* file, describing the SM: it will be used by LS-LAB developers to extend the SSM and to provide a suitable adapter between the SSM and the actual SM, that can be given as input to the algorithm;
- a domain description compliant to IEEE-LOM standard and, if required, the semantics of the used tags: it will be used by LS-LAB developers to provide a suitable parser, that will translate the IEEE-LOM metadata of the learning materials of a given domain into the correct input for that particular sequencing algorithm;
- a TK description to allow LS-LAB developers for providing a suitable parser to give the correct input to the algorithm.

A researcher who wants to run two or more algorithms already present in the system, has to fill-in the GUI of the system, giving the information about a simulated student, about the domain to be used and about the TK. Moreover she has to select the algorithms she wants to compare. The SM, DK and TK information are consequently translated by the parsers associated to the selected algorithms and sent as input to the algorithms themselves. Their outputs are then shown to the researcher.

III. LS-LAB AT WORK

In this Section we present an example of the use of the LS-LAB system for comparing two sequencing algorithms. The first algorithm is the one used by the LS-PLAN system [11], [12], based on the Pdk (Planning with Domain Knowledge) planner [13]. Pdk conforms to the *planning as satisfiability* paradigm, and the logic used to encode planning problems is the propositional Linear Time Logic [18]. The second algorithm is a classical Topological Sort Algorithm (TSA), used in the literature by a number of adaptive educational systems such as the IWT system [14], the KBS-Hyperbook system [9], and the Lecomps system [15]. The TSA uses student's previous knowledge for building a personalized sequence performing a depth first traversal, while LS-PLAN uses both student's previous knowledge and *Learning Styles* (LS) according to the Felder and Silverman's model (FS) [8].

The Domain used in this experiment is the *Italian Neorealist Cinema*, as shown in Fig. 3, where edges represent prerequisite relations among concepts. Some of these concepts are linked to more than one learning material. In particular, since the LS-PLAN sequencing algorithm exploits LS, we used a KD where each LN is equipped with four weights, one for each FS dimension, representing the suitability of the LN for a given

student. In the following we show the insertion and running process managed by LS-LAB .

A. Learning Node Binding

As explained in the previous Section, the LN description already provides RK and AK specifications, given through suitable metadata compliant to the IEEE-LOM specifications. In the case of LS-PLAN, we must represent the LSs also, both for the student and for the LN. In the case of the LNs, the binding of LS specification to IEEE-LOM metadata is obtained by exploiting the tag <purpose> that belongs to the $9-th$ category of the IEEE-LOM <classification>. In the <taxonPath> field, the tag <taxon> contains an <id> and a <string> tag where the last one describes the LS associated to the LN as shown in Fig. 2.

```
<!-- Learning Styles definition -->
<classification>
  <purpose>
    <source>LOMv1.0</source>
    <value>accessibility restriction</value>
  </purpose>
  <taxonPath>
    <source>
      <string language="it-IT">Rossellini</string>
    </source>
    <taxon>
      <id>LearningStyle</id>
      <entry>
        <string language="it-IT">vis_vrb=-2|sen_int=+7</string>
      </entry>
    </taxon>
  </taxonPath>
</classification>
```

Fig. 2. IEEE-LOM specifications for LS.

The domain parser associated to the LS-PLAN sequencing algorithm, acquires both LS weights and prerequisite relations from the LOM metadata of each LN translating them to the correct input for the sequencing algorithm. In particular, the parser translates each node in an *action* with RK as preconditions and AK as post-conditions. The parsing of all the set of LNs represents the *domain* description and it is written in PDDL language, that is the input language for the Pdk planner. Moreover, in a post-processing phase the LNs with their LS most similar to the student's ones will be assigned. Instead, the domain parser, associated to the TSA will create a graph starting from the set of LNs which are already defined and compliant to TSA. It will order them, and in a post-processing phase will throw away the LNs that correspond to concept already known by the student (SK).

B. New Student Model Insertion

As stated in Section II-B, the SSM starts from scratch, i.e., $SSM_0 \equiv \emptyset$, increasing as new SMs are loaded into the system together with their related sequencing algorithms. TSA requires KIs that represent the student's SK, related to the topics she has to work with. In this case the SSM becomes:

$$SSM_1 = SSM_0 \cup SK$$

LS-PLAN requires, in addition, also the LS that describes the learning preferences of the student as defined in the following:

Definition 8: Learning Style (LS). A LS is a 4-tuple: $LS = \langle D_1, D_2, D_3, D_4 \rangle, \quad with\ D_i \in [-11, \ldots, +11]$
with $i = 1, \ldots, 4$ being each D_i a Felder-Silverman Learning Style Dimension [8], i.e., D_1: active-reflective, D_2: sensing-intuitive, D_3: visual-verbal, D_4: sequential-global.

So, the SK, as well as the TK related to a given course, are common to both the SMs, while the LSs are significant for the LS-PLAN algorithm only. The SSM at this stage will be the following:

$$SSM_2 = SSM_1 \cup \{LS\}$$

In general, with the uploading of a new algorithm A_i into the system, the SSM_i can stay unchanged, if the information present in the SM_i is already contained in the SSM_i, or can be increased with new information that are not yet provided by the SSM_{i-1}. As we can see in Fig. 1, also the data for SM_i will be filtered and parsed, to be taken as input by the algorithm A_i.

C. Target Knowledge description

Since TK, as given in Def. 6 is a set of KIs, in this case the KIs given by the TK will be used by the first algorithm, to filter the LOSs produced by TSA, while in the second algorithm the set TK will be translated in a way that the elements will contribute to define the *problem* specification, written in PDDL language. Finally the two algorithms can be uploaded in the Sequencing Engine.

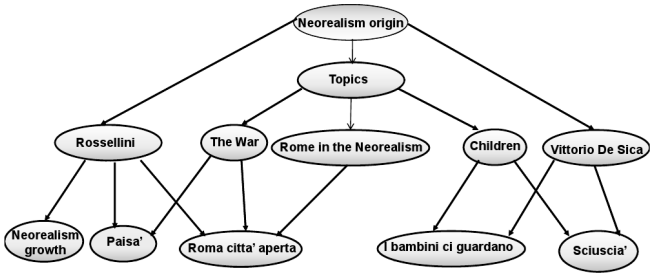

Fig. 3. Italian Neorealist Cinema Domain.

D. Results

We run the two algorithms on two different SM. The first, SM_1, is a learner that knows nothing about the domain. The second, SM_2, is a learner that knows the following concepts: *Neorealism origin*, *Rossellini*, *I bambini ci guardano*, *Rome in the Neorealism*, *De Sica*, *Topics*, *The War*, *Children*. The LOSs produced by the two algorithms for the two SM are shown in Fig. 4. As we can see, the TSA did not manage the LS associated to the SM and to the LN: its LOS was built taking into account the *default* learning material only, i.e., the learning material marked as suitable for every SM, shown in

SM_1 - LOS	
TSA	**LS-Plan**
Neorealism origin	Neorealism origin
Vittorio De Sica	Topics_2
Topics_1	Children
The War	The War
Rome in the Neorelaism	Rossellini_2
Children	Paisa'_2
Sciuscia'	Neorealism growth_2
I bambini ci guardano_1	Rome in the Neorealism
Rossellini_1	Roma citta' aperta_2
Roma citta' aperta_1	Vittorio De Sica
Paisa'_1	I bambini ci guardano_2
Neorealism growth_1	Sciuscia'

SM_2-LOS	
TSA	**LS-Plan**
Sciuscia'	Neorealism growth_2
Roma citta' aperta_1	Paisa'_1
Paisa'_1	Roma citta' aperta_1
Neorelism growth_1	Sciuscia'

Fig. 4. Produced LOSs.

Fig. 4 with "_1". LS-PLAN selected, instead, different learning materials on the basis of the student's LS. LS-LAB does not allow for a validation of the produced LOSs: this is left to teachers and instructional designers. These experts can evaluate the suitability of the sequences for a given SM and their opinions can be stored, allowing for a database of useful past evaluations.

IV. Conclusions and Future Work

In this paper we presented the LS-LAB system, a framework, now at its early stage of development, that provides researchers with a suitable environment where to compare and test their Curriculum Sequencing algorithms. We decided to develop such a kind of system because of a lack in the literature of the Curriculum Sequencing research area of such an environment: while many sequencing algorithms have been presented during years, there is not a suitable environment where people can select a particular sequencing algorithm for a particular KD or to compare different ones for research purposes.

The final phase of comparison among output sequences is not the central issue of this contribution, although it is the final goal. In this respect two main approaches should be considered: a "subjective" comparison, where a system expert judges the suitability of the produced sequences, and an "objective" comparison, where some suitable heuristics decide the goodness of the sequences, while some suitable metrics measure their similarity.

As a future work we plan to implement other sequencing algorithms together with the possibility to communicate, through suitable communication protocols (e.g. SOAP) with remote algorithms, a better GUI and a *xml* binding in order to automatically input many semantic information such as the SM.

References

[1] M. Baldoni, C. Baroglio, I. Brunkhorst, E. Marengo, and V. Patti. A service-oriented approach for curriculum planning and validation. In *Proceedings of International Workshop on Agents, Web-Services, and Ontologies - Integrated Methodologies, Durham, UK (6th–7th September 2007)*, pages 108–123, 2007.

[2] M. Baldoni, C. Baroglio, N. Henze, and V. Patti. Setting up a framework for comparing adaptive educational hypermedia: First steps and application on curriculum sequencing. In *Proc. of ABIS-Workshop 2002: Personalization for the mobile World, Workshop on Adaptivity and User Modeling in Interative Software Systems*, pages 43–50, Hannover, Germany, October 2002.

[3] M. Baldoni, C. Baroglio, V. Patti, and L. Torasso. Reasoning about learning object metadata for adapting SCORM courseware. In *Int. Workshop on Engineering the Adaptive Web, EAW*, volume 4, pages 4–13, 2004.

[4] P. D. Bra, D. Smits, and N. Stash. Creating and delivering adaptive courses with AHA! In *EC-TEL*, pages 21–33, 2006.

[5] P. Brusilovsky. Developing adaptive educational hypermedia systems: From design models to authoring tools. *Authoring Tools for Advanced Technology Learning Environment. Dordrecht: Kluwer Academic Publishers*, pages 377–409, 2003.

[6] P. Brusilowsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11:87–110, 2001.

[7] P. Brusilowsky and J. Vassileva. Course sequencing techniques for large-scale web-based education. *International Journal of Continuing Engineering Education and Life-long Learning*, 13:75–94, 2003.

[8] Felder and Silverman. Learning and teaching styles in engineering education. *Engr. Education*, 1988.

[9] N. Henze and W. Nejdl. Adaptation in open corpus hypermedia. *International Journal of Artificial Intelligence in Education*, 12(4):325–350, 2001.

[10] J.Vassileva. *Dynamic CAL-courseware generation within an ITS-shell architecture*, volume 602 of *Lecture Notes in Computer Sciance*, pages 581–591. Springer-Verlag, i.tomek edition, 1992.

[11] C. Limongelli, F. Sciarrone, and G. Vaste. Ls-plan: An effective combination of dynamic courseware generation and learning styles in web-based education. In W. Nejdl, J. Kay, P. Pu, and E. Herder, editors, *Proc. of 5-th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2008)*, number 5149 in Lecture Notes in Computer Science, pages 133–142. Springer, 2008.

[12] C. Limongelli, F. Sciarrone, and G. Vaste. An application of the ls-plan system to an educational hypermedia. *Int. J. of Web-Based Learning and Teaching Technologies*, 4(1):16–34, Jan-March 2009.

[13] M. C. Mayer, C. Limongelli, A. Orlandini, and V. Poggioni. Linear temporal logic as an executable semantics for planning languages. *J. of Logic, Lang. and Inf.*, 1(16):63–89, Jan 2007.

[14] E. Sangineto, N. Capuano, M. Gaeta, and A. Micarelli. Adaptive course generation through learning styles representation. *Universal Access in the Information Society*, 7(1):1–23, 2008.

[15] A. Sterbini and M. Temperini. A logical framework for course configuration in elearning. In *Proc. of ITHET03*, July 2003.

[16] G. Weber and P. Brusilovsky. ELM-ART: An adaptive versatile system for Web-based instruction. *International Journal of Artificial Intelligence in Education*, 12(4):351–384, 2001.

[17] S. Weibelzahl and C. U. Lauer. Framework for the evaluation of adaptive CBR-systems. In I. Vollrath, S. Schmitt, and U. Reimer, editors, *Experience Management as Reuse of Knowledge. Proceedings of the 9th German Workshop on Case Based Reasoning, GWCBR2001*, pages 254–263, Baden-Baden, Germany, 2001.

[18] P. Wolper. The tableau method for temporal logic: an overview. *Journal of Logique et Analyse*, 28:119–152, 1985.

[19] V. Zadorozhny, M. Yudelson, and P. Brusilovsky. A framework for performance evaluation of user modeling servers for web applications. *Web Intelli. and Agent Sys.*, 6(2):175–191, 2008.