

LS-Plan: an Effective Combination of Dynamic Courseware Generation and Learning Styles in Web-Based Education

Carla Limongelli¹, Filippo Sciarrone², and Giulia Vaste¹

¹ “Roma Tre” University - Department of Computer Science and Automation
Via della Vasca Navale, 79 - 00146 Rome, Italy

² Open Informatica srl - Adaptive Technology Lab.
Via dei Castelli Romani 12a - 00040 Pomezia, Italy
{limongel,sciarro,vaste}@dia.uniroma3.it

Abstract. This paper presents LS-PLAN, a system capable of providing Educational Hypermedia with adaptation and personalization. The architecture of LS-PLAN is based on three main components: the Adaptation Engine, the Planner and the Teacher Assistant. Dynamic course generation is driven by an adaptation algorithm, based on Learning Styles, as defined by Felder-Silverman’s model. The Planner, based on Linear Temporal Logic, produces a first Learning Objects Sequence, starting from the student’s Cognitive State and Learning Styles, as assessed through pre-navigation tests. During the student’s navigation, and on the basis of learning assessments, the adaptation algorithm can propose a new Learning Objects Sequence. In particular, the algorithm can suggest different learning materials either trying to fill possible cognitive gaps or by re-planning a newly adapted Learning Objects Sequence. A first experimental evaluation, performed on a prototype version of the system, has shown encouraging results.

1 Introduction

Personalization and adaptation in learning environments are two very important requirements for providing an effective educational service on the Internet. In this context, Dynamic Courseware Generation [6] and Instructional Planning [16] are two of the most important research areas.

In this work we address the problem of helping the student during his learning activity by means of a synergy based on his cognitive state, his learning styles and the teacher’s didactic strategy. The main contribution of our work is given by the adaptation algorithm, capable to modify the student’s model and to guide the student step by step, especially in recovery activity. At the same time the system lets the student free to navigate in the learning hyperspace in accordance with the constructivist pedagogical theory [17]. Here we propose LS-PLAN, a Web-based system, capable of providing Educational Hypermedia with adaptation and personalization. The system is based on the synergy between classical planning techniques and Learning Styles refinement procedures.

The architecture of LS-PLAN includes three main modules: the Adaptation Engine, the Planner and the Teacher Assistant. The Adaptation Engine manages the adaptivity mechanism and the user model, from its initialization to its update. The Planner produces a Learning Objects Sequence (*LOS*), on the basis of the current Student Model and of the learning strategies previously set by the teacher. The Teacher Assistant allows the teacher to modify the teaching strategies related to the learning material. The pedagogical background of the Student Model is based on the student's Cognitive State (*CS*) and Learning Styles (*LS*). The student's *CS* is defined as a set of Knowledge Items, i.e., atomic elements of knowledge concerning the learning domain, according to the Knowledge Space Theory [10]. *LS* are the student's learning preferences as defined by Felder-Silverman's (*FS*) Learning Styles Model [11]. Moreover, the system models the student's knowledge by an Overlay Model [5], based on three of the five levels of Bloom's Taxonomy [2]. Our system is based on the idea that *LS* are tendencies and may change through educational experiences [12]. In fact, the system takes into account the information gathered from the student's self-assessments and navigation, in order to evaluate the effectiveness of the current teaching strategy, modifying it, if necessary.

In the literature, different systems have been proposed on the basis of the *FS* Model and for generating *LOS*. In the system proposed in [1], an adaptive interface has been presented, while the CS383 system [8] and the Intelligent Web Teacher system [7] propose an adaptive presentation based on learning material typologies. Our system generates *LOS* by means of planning techniques, similarly to the Dynamic Course Generation (DCG) system [6] following the style of AHA! [3] and ELM-ART [18]: while AHA! does not exploit assessment for adaptivity, ELM-ART and DCG do not make use of *LS*. Our adaptation mechanism provides both features and it is very fine grained: a specific learning material has associated its own *LS*, thus providing the teacher with the possibility to implement suitable didactic strategies for different learners.

The rest of the paper is organized as follows. Section 2 illustrates the architecture of LS-PLAN together with its main components. Section 3 shows a first experimentation of the system to a real instructional environment. In Section 4 our conclusions are drawn.

2 The Adaptive System

The overall system, that is LS-PLAN together with the Adaptive Educational Hypermedia (AEH), is shown in Figure 1 where the main components are highlighted with grey blocks.

The teacher, through a suitable framework, the *Teacher Assistant*, arranges a pool of learning objects, i.e., learning nodes, building the Domain Knowledge, stored in a special repository inside the AEH system. The teacher also prepares the initial *Cognitive State Questionnaire* for evaluating the starting knowledge of the student, that is the knowledge already possessed by the student with respect to the topic to be learned. Moreover, the teacher provides each student

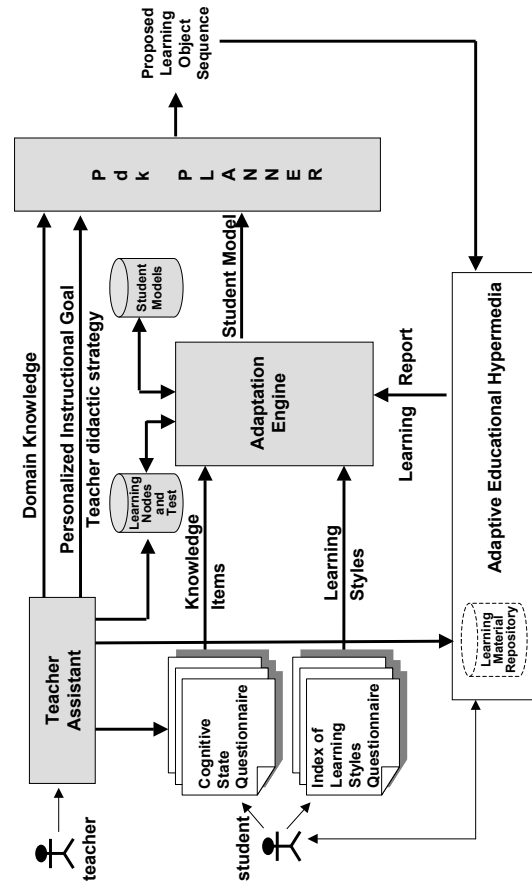


Fig. 1. The Functional Schema of the Adaptive System. Grey blocks form LS-PLAN.

with his own instructional goal, and specifies the didactic strategies. The *Index of Learning Styles (ILS) Questionnaire*, developed by Felder and Soloman ³, is also submitted to the student with the aim of mapping his learning preferences to the four dimensions of the *FS Model*: active-reflective, sensing-intuitive, visual-verbal, sequential-global [11]. The information gathered through the two questionnaires, is processed by the *Adaptation Engine* module, that builds the initial Student Model and updates the *Student Models Database* that collects all of them.

The *Pdk Planner*, described in Section 2.3, takes in input the Domain Knowledge, the current Student Model, the Personalized Instructional Goal, and the teacher didactic strategies, giving in output to the AEH a new *LOS* tailored to that particular student. According to the constructivist theory, the student is not forced to follow the *LOS* generated by the planner.

The *Adaptation Engine* builds a Learning Report on the basis of the student's navigation in the AEH. Consequently, it builds a new Student Model starting from the current one. If significant variations are detected, the adaptation algorithm provides an alternative *LOS*, as it will be discussed in Section 2.1.

Before entering in details of the Student Model, its updating procedures and the planner, we introduce some definitions about the elements we are going to work with.

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

$$KI = \{KI_K, KI_A, KI_E\}$$

where KI_ℓ , with $\ell \in \{K, A, E\}$, represents a cognitive level taken from Bloom's Taxonomy: Knowledge, Application and Evaluation.

Let us note that it is possible to give different meanings to the cognitive levels, by using other classifications like, for example, the one proposed in [9].

Definition 2. (LEARNING STYLE). *A Learning Style LS is a 4-tuple:*

$$LS = \langle D_1, D_2, D_3, D_4 \rangle, \quad \text{with } D_i \in [-11, \dots, +11], \quad i = 1, \dots, 4$$

where each D_i is a FS Learning Style Dimension, i.e., D_1 : active-reflective, D_2 : sensing-intuitive, D_3 : visual-verbal, D_4 : sequential-global.

We used the range $[-11, \dots, +11]$ according to the Felder-Soloman *ILS* scale.

Definition 3. (LEARNING NODE). *A Learning Node LN is a 5-tuple:*

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

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

³ Available at <http://www.engr.ncsu.edu/learningstyles/ilsweb.html>

AK *Acquired Knowledge*. It is a KI_ℓ , with an associated success threshold σ_{KI_ℓ} defined in Definition 4, that represents the knowledge that the student acquires at a given level as specified in Definition 1, after having passed the assessment test related to the KI_ℓ of the node. If such a test is not present in the node the AK is considered acquired anyway.

RK *Required Knowledge*. It is the set of KI_ℓ necessary for studying the material of the node, i.e., the cognitive prerequisites required by the AK associated to the node.

LS is given in Definition 2.

T is a pair of reals $T = (t_{min}, t_{max})$ which represents the estimated time interval for studying the material of the node, as prefixed by the teacher. A fruition time, t_f , less than t_{min} is not a realistic time to learn that material; for a fruition time, t_f , greater than t_{max} we have the so-called "coffee break" effect, i.e., the student is supposed to have done something else.

Definition 4. (THRESHOLD σ_{KI_ℓ}). A threshold value σ_{KI_ℓ} is a real number associated to KI_ℓ , defined as:

$$\sigma_{KI_\ell} = \frac{S_T}{S_{max}}, \quad 0 < \sigma_{KI_\ell} \leq 1$$

being S_T the lowest score of an assessment test, as fixed by the teacher, in order to consider the KI_ℓ acquired; S_{max} is the highest possible score for that test⁴.

Definition 5. (POOL). A pool is the particular set of LN, selected or created by the teacher in order to arrange a course about a particular topic.

Definition 6. (DOMAIN KNOWLEDGE). The Domain Knowledge DK is the set of all the KI present in a pool.

Definition 7. (COGNITIVE STATE). The Cognitive State CS is the set of all the KI_ℓ possessed by the student with respect to the given topic: $CS \subseteq DK$.

Definition 8. (STUDENT MODEL). The Student Model SM is a pair:

$$SM = (CS, LS)$$

where, CS is given in Definition 7 and LS is given in Definition 2.

Definition 9. (TEST). A Test is a set of k items, i.e., questions, with $k \in \mathcal{N}$. To each item is associated a weight $Q_j \in \mathcal{R}$. Each item has m answers, with $m \in \mathcal{N} - \{0, 1\}$ and to each answer is associated a weight $w_i \in \mathcal{R}$.

Let S_{KI_ℓ} be the score associated to a test; it assesses the student knowledge of the single KI_ℓ :

$$S_{KI_\ell} = \sum_{j=1}^k (Q_j \cdot \sum_{i=1}^m w_i)$$

where $w_i = 0$ for the answers the student does not select.

⁴ Here and in the following we suppose that the teacher gives to S_T and S_{max} positive values.

Definition 10. (ACQUISITION OF A KI_ℓ). A KI_ℓ is supposed to be acquired by the student if:

$$S_{KI_\ell} = \sum_{j=1}^k (Q_j \cdot \sum_{i=1}^m w_i) \geq \sigma_{KI_\ell}$$

where σ_{KI_ℓ} is given in Definition 4 and k , m , Q_j , w_i and S_{KI_ℓ} are given in Definition 9.

2.1 The Adaptation Engine

In this section we show the mechanisms of the SM management, i.e., the initialization and the updating processes, based on adaptation strategies.

Student Model Initialization. At the first access to the system the student fills in the *Cognitive State Questionnaire* composed by some tests (see Definition 9), related to the KI of the Domain. The acquisition of a KI_ℓ is described in Definition 10. All the acquired KI_ℓ initialize the CS , which can also be an empty set if the student does not know anything about the domain. The student also fills in the *ILS Questionnaire* whose result is used to initialize the LS .

Student Model Updating and Adaptive Methodology. In order to update the SM and guide the student to fill his cognitive gaps, we propose the algorithm presented in Figure 2, where the function UPDATEANDLEARN returns the SM

UPDATEANDLEARN (LearningNode LN , StudentModel SM)

```

 $t_f \leftarrow$  TIME SPENT ON THE NODE( $LN$ )
 $S_{KI_\ell} \leftarrow$  COMPUTE SCORE POST TEST( $KI_\ell$ )
 $SM \leftarrow$  ( $CS$ , UPDATE LS( $LS$ ,  $LN$ ,  $t_f$ ,  $S_{KI_\ell}$ ))
if ((not post-test) or ( $KI$  acquired)) then
     $SM \leftarrow$  ( $CS \cup AK$ ,  $LS$ )
    if ( $\exists D_i$  that changed sign) then REPLAN( $SM$ ); return ( $LN^{first}$ ,  $SM$ )
    else return ( $LN^{next}$ ,  $SM$ )
else if ((time-out( $t_f$ )) and (not read)) then return ( $LN$ ,  $SM$ )
    else  $LN' \leftarrow$  CHECK CLOSEST NODE( $LN$ ,  $SM$ )
        if ( $LN' \neq NIL$ ) then return ( $LN'$ ,  $SM$ )
        else  $L \leftarrow$  ORDERED PREDECESSORS LIST( $LN$ )
            if ( $L \neq NIL$ ) then
                 $\forall LN_i \in L$ , if ( $AK \in CS$ ) then  $CS \leftarrow CS - AK$ 
                ADD_L_TO_PLAN( $L$ ); return ( $LN^{first}$ ,  $SM$ )
            else REPLAN( $SM$ ); return ( $LN^{first}$ ,  $SM$ )

```

Fig. 2. UPDATEANDLEARN: $LearningNode \times StudentModel \rightarrow (LearningNode \times StudentModel)$.

after studying the node LN , and proposes the next node to be learned. It takes

in input a LN and SM . When the student study a LN , this function is activated with the information about LN and with the current SM .

The function $TIMESPENTONTHENODE(LN)$ computes and returns the time t_f spent by the student on the node. The function $COMPUTESCOREPOSTTEST(KI_\ell)$ computes and returns the score taken by the student in the post-test related to the KI_ℓ of the node, that is the AK related to LN . If the post test does not exist we assume a score equal to 0.

The student LS is updated by means of the function $UPDATELS$, according to the LS associated to the chosen node, to the time spent for studying the instructional material, to the score got with the post-test assessment and to the knowledge level of the node.

If the node does not provide any post-test assessment, i.e., the boolean variable “post-test” is “false”, or if the student passes the test, the SM acquires the KI_ℓ related to that node.

If a D_i (as given in Definition 2) changes sign, we consider that a significant variation in the student LS is present, and it is necessary to re-plan the LOS : The algorithm suggests the first LN of the new LOS computed by the planner.

If the student does not pass the test, the time t_f , i.e., the fruition time, is examined: the boolean function “time-out” checks whether t_f is out of range and if it is the first time that the LN has been studied. In positive case, the system proposes once again the same node to the student. After the second unsuccessful trial, the system applies the function $CHECKCLOSESTNODE$, that looks for the “closest” node similar to the student LS , with the same RK and AK of that LN . If such a node does not exist, the algorithm by means of the function $ORDEREDPREDECESSORSLIST$, computes the list L of the LN predecessors, i.e., the nodes connected to LN by an incoming link, in order to verify the acquisition of prerequisites, RK , related to LN . The AK of the prerequisite nodes, if present, are removed from the CS , because we are in presence of a sort of “loss” of knowledge. Then the algorithm puts L on the top of the LOS and suggests the first LN , of such a new LOS . If both the attempts to explain the concepts with different learning material and the prerequisite checks fail, the algorithm replans a new LOS and proposes the first node of such a sequence.

The functions $UPDATELS$, $CHECKCLOSESTNODE$ and $ORDEREDPREDECESSORSLIST$ are explained in the following.

$UPDATELS$. Each D_i related to a student LS is updated on the basis of the value of the previous D_i , and on the basis of the difference between the D_i of the node and the D_i of the student (ΔD_i). The updating shown in equation (1) is a function that takes into account some of the possible feedbacks that the student gives to the system when he deals with the contents of a given node: the fruition time, the score obtained with the test and the difficulty level.

$$D_{i_{new}} = D_{i_{old}} + (\alpha(t_f) + \beta(S_{KI_\ell}, \ell)) | \Delta D_i | \quad (1)$$

The second addendum of the sum in the equation (1) ranges between -1 and 1 . Moreover $\alpha(t_f)$ is a function of the fruition time t_f , i.e., the time spent by the student on the node. If KI is acquired, the student LS is reinforced towards the

node LS ; the less is the fruition time t_f , the more is the reinforcement. β is a function of the score S_{KI_ℓ} , obtained by the student in the post-test, and of the knowledge level ℓ associated to the KI . Let us note that the updating of the LS is computed apart from the acquisition of a KI .

CHECKCLOSESTNODE. The closest node is computed by selecting an alternative node, LN_{alt} with the same RK and the same AK of the current LN , that is the closest to the student LS , on the basis of the following Euclidean distance metric:

$$d(LS_{LN_{alt}}, LS_{student}) = \sqrt{\sum_{i=1}^4 (D_i^{LN_{alt}} - D_i^{student})^2} \quad (2)$$

ORDEREDPREDECESSORSLIST. The list is composed by all the nodes that are predecessors of LN , ordered according to the following priorities: (i) the predecessor nodes that have not been visited by the student. In fact it is possible that the student got the AK related to that node, by giving a correct answer to the initial test, but he lacks that concept indeed; (ii) the nodes that do not provide tests are proposed on the basis of the difficulty levels: K , A , E ; (iii) the nodes that provide test whose LS are closest to the student LS , by following the equation (2).

2.2 The Teacher Assistant

The *Teacher Assistant* is responsible for the management of the functionalities provided for the teacher, i.e., for the management of the pool. The teacher also selects the items and the threshold for the *Cognitive State Questionnaire* and manages the students' registration to the course. In particular he decides the student's instructional goal and specifies his didactic strategies, such as the desired level of the course, or the particular way he prefers to explain a given concept.

2.3 The Pdk Planner

In automated planning, *planning languages* are used to specify problems in a uniform and simple way. In the context of course configuration, planning problems are described by "actions" (LN), specifying action preconditions (RK) and action effects (AK), as well as the initial state (initial SM) and the goal.

Besides all these basic elements a teacher would be allowed to express his didactic strategy, e.g., preferences related to a concept explanation. Unfortunately standard planning language, such as PDDL [14], and classical planners are not suitable to describe such kind of learning problems. To this aim we use the planning language PDDL-K and the Pdk planner (Planning with Domain Knowledge) [15]⁵. It conforms to the "planning as satisfiability" paradigm: the logic used to encode planning problems is propositional Linear Time Logic (LTL). The related planning language PDDL-K[15], conforming to standard PDDL,

⁵ Available at <http://pdk.dia.uniroma3.it/>

guides the teacher, through the Teacher Assistant, in the specification of heuristic knowledge, providing a set of control schemata, that is a simple way of expressing control knowledge. The language is given an executable semantics by means of its translation into LTL.

In the following section, in the framework of empirical evaluation, we show how the planner can configure different courses.

3 A First Evaluation

In this Section we show a first evaluation of our pilot system with the aim to show a first assessment of the system capability in configuring different *LOS*, starting from different student models and from different pedagogical strategies. The experiment has been performed starting from a pool of 20 *LN* related to the *Recursion* topic and arranged in a such a way that the teacher could decide to explain Recursion either with the induction principle or activation records. We followed the experimental methods and procedures suggested in [4] and in [13].

Experimental Setup. We consider the two following case studies.

First Student Model: $SM_1 = (CS_1, LS_1) = (\emptyset, \langle 3, 3, 7, 7 \rangle)$, that is, the student is supposed to know nothing about the *DK*, while his *LS*₁ are: reflective, intuitive, verbal, global.

First Pedagogical Strategy: The teacher desires to configure the course at the Evaluation Level and decides to explain recursion through activation records.

Second Student Model: $SM_2 = (CS_2, LS_2) = (\{recursive_programs, rec_fun_K\}, \langle -3, -5, -7, -9 \rangle)$, that is, the student is supposed to know something about recursive programs and the functional approach to recursion at the Knowledge level, while his *LS*₂ are: active, sensing, visual, sequential.

Second Pedagogical Strategy: the teacher desires to configure the course at the Application level and decides to explain recursion in a functional manner.

The instructional goal is in both cases learning *Recursion*.

Experimental Results. The planner produced the following two *LOS*, for the first and second *SM* with their related pedagogical strategies, respectively.

LOS 1	LOS 2
1) Unit_Description	1) Rec_Fun_StringReverse
2) Recursive_Programs	2) Rec_Exercises
3) Rec_RunTimeStack_Intro	3) Rec_List_VI
4) Rec_RunTimeStack_Factorial	4) Rec_List_Examples
5) Rec_RunTimeStack_Use_Examples	5) Rec_List_Exercises
6) Rec_List_VE	6) Complements_GroupWorking
7) Rec_List_Examples	
8) Rec_Exercises	
9) Rec_List_Exercises	
10) Complements_Reflection_Proposals	

Discussion. The first planned learning path included nodes related to activation records at all the difficulty levels, i.e., *K*, *A* and *E*, to obtain the Evaluation

level is necessary to know the previous ones: K and A . The learning path is also suitable for the student because it reflects his LS and starting knowledge: all the contents are proposed because the student has an empty starting knowledge; list recursion is explained in a *verbal* way; suggested complements are proposals for thinking about the learning material. Moreover the proposed LOS presents all the theoretical components before proposing exercises. The LOS includes a node that provides an overall picture of the topics of the unit. These are suitable features for *global* learners.

The second LOS is also suitable for the student: it does not include the *Recursive_Programs* and the *Rec_Fun_Intro* nodes because the student knows these concepts; it proposes *visual* didactic material. It suggests complements for group working, this is motivating for an *active* learner. Moreover, theoretical material is immediately followed by exercises, because the learner prefers studying in a *sequential* manner. Finally the LOS reflects the teacher didactic strategies, i.e.: *functional* nodes.

These two learning sequences were assessed by a small sample of 14 teachers who were required to assess the instructional validity of the two proposed didactic plans compared to their related SMs. To this aim, we submitted to our experimental group the following question both for LOS_1 and LOS_2 : *This Learning Object Sequence is a valid Learning Object Sequence on the basis of the starting student model SM?*, with a 5-points Likert scale (strongly disagree, disagree, neither agree nor disagree, agree, strongly agree).

The experimental results have shown the following results: 7.1% disagree, 7.1% neither agree nor disagree, 71.4% agree and 14.4% strongly agree with the first didactic plan; 78.6% agree and 21.4% strongly agree for the second one.

4 Conclusion and Future Work

In this work we proposed a Web-based system for personalizing and adapting sequences of learning objects. The main contribution of this work consists of combining course generation with an adaptation algorithm, based on Learning Styles. During the learning activity, the student navigates through learning nodes and its model is constantly updated. If the student fails a post-test assessment, the adaptation algorithm proposes alternative learning strategies.

Let us note that the use of control knowledge in planning domain description languages, such as the PDDL-K, enriches the expressivity of relations among concepts to be taught and helps the teacher both in configuring optimized courses and managing the pool of learning nodes. We plan to extend the PDDL-K with new syntactic elements that help the teacher in building the pool of learning nodes.

References

1. N. Bajraktarevic, W. Hall, and P. Fullick. Incorporating learning styles in hypermedia environment: Empirical evaluation. In *Proceedings of the Fourteenth Conference on Hypertext and Hypermedia*, pages 41–52, 2003.

2. B.S. Bloom. *Taxonomy of Educational Objectives*. David McKay Comp. Inc., 1964.
3. P. De Bra, D. Smits, and N. Stash. Creating and delivering adaptive courses with AHA! In *EC-TEL*, pages 21–33, 2006.
4. P. Brusilovsky, C. Karagiannidis, and D. Sampson. Layered evaluation of adaptive learning systems. *International Journal of Continuing Engineering Education and Lifelong Learning.*, 14(4/5):402–421, 2004.
5. P. Brusilovsky and E. Millan. User models for adaptive hypermedia and adaptive educational systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Heidelberg New York.
6. P. Brusilovsky 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.
7. N. Capuano, M. Gaeta, A. Micarelli, and E. Sangineto. Automatic student personalization in preferred learning categories. In *3rd International Conference on Universal Access in Human-Computer Interaction*, 2005.
8. C. A. Carver, R. A. Howard, and W. D. Lane. Enhancing student learning through hypermedia courseware and incorporation of student learning styles. *IEEE Transactions on Education*, 42(1):33–38, 1999.
9. G. Trentin da Silva and M. Costa Rosatelli. Adaptation in educational hypermedia based on the classification of the user profile. In Springer Verlag, editor, *8th International Conference, ITS 2006*, number 4053 in LNCS, 2006.
10. J.C. Falmagne, M. Koppen, M. Villano, J.P. Doignon, and L. Johannesen. Introduction to knowledge spaces: How to build, test, and search them. *Psychological Review*, 97(2):201–224, April 1990.
11. R. M. Felder and L. K. Silverman. Learning and teaching styles in engineering education. *Engineering Education*, 78(7):674, 1988.
12. R. M. Felder and J. Spurlin. Application, reliability and validity of the index of learning styles. *Int. Journal of Engineering Education*, 21(1):103–112, 2005.
13. C. Gena. Methods and techniques for the evaluation of user-adaptive systems. *Knowl. Eng. Rev.*, 20(1):1–37, 2005.
14. M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins. Pddl—the planning domain definition language, 1998.
15. M. Cialdea 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.
16. P. Mohan, J. Greer, and G. McCalla. Instructional planning with learning objects. In Kohlhase Melis Baumgartner, Cairns, editor, *n IJCAI-03 Workshop 'Knowledge Representation and Automated Reasoning for E-Learning Systems*, 2003.
17. J. Piaget. *Language and thought of the child*. New York: Harcourt, 1926.
18. G. Weber and P. Brusilovsky. Elm-art: An adaptive versatile system for web-based instruction. *International Journal of AI in Education*, 12(4):351–384, 2001.