Interactive Learning Environments

LIA: an Intelligent Advisor for e-Learning

Nicola Capuano[a*,b], Matteo Gaeta[a], Agostino Marengo[c], Sergio Miranda[a], Francesco Orciuoli[a,b] and Pierluigi Ritrovato[a]

[a]*DIIMA, University of Salerno, Italy;* [b]*CRMPA, Fisciano (SA), Italy;* [c]*DSS, University of Bari, Italy*

Intelligent e-learning systems have revolutionized online education by providing individualized and personalized instruction for each learner. Nevertheless, till now very few systems were able to leave academic labs and be integrated in real commercial products. One of these few exceptions is the Learning Intelligent Advisor (LIA) described in this paper, built on results coming from several research projects and currently integrated in a complete e-learning solution named IWT. The purpose of this paper is to describe how LIA works and cooperates with IWT in the provisioning of individualized e-learning experiences. Defined algorithms and underlying models are described as well as architectural aspects related to the integration in IWT. Results of experimentations with real users are discussed to demonstrate the benefits of LIA as an add-on in on-line learning.

Keywords: Course Sequencing, Knowledge Representation, Learner Modeling.

1. Introduction and Related Work

The Learning Intelligent Advisor (LIA) is an intelligent tutoring engine capable of integrating, in "traditional" e-learning systems, "intelligent" features like learner modelling and learning experience individualisation. LIA was born from the cooperation of an high-tech company named MoMA with the Research Centre in Pure and Applied Mathematics (CRMPA) and the Department of Information Engineering and Applied Mathematics (DIIMA) of the University of Salerno and it is currently included in a complete solution for e-learning (MoMA, 2006) named Intelligent Web Teacher (IWT).

LIA is based on a set of models able to represent the main entities involved in the process of teaching/learning and on a set of methodologies, leveraging on such models, for the generation of individualised learning experiences with respect to learning objectives, pre-existing knowledge and learning preferences. The research about LIA falls in the field of Intelligent Tutoring Systems (ITS) and deals with ***course sequencing techniques*** (Brusilovsky & Vassileva, 2003).

Several approaches to course sequencing have been defined till now and are exploited by experimental ITS. The majority of such systems, like (Eliot et al., 1997) and (Rios et al., 1999), only deals with *task sequencing* i.e. they are able to generate sequences of problems or questions in order to improve the effectiveness and the efficiency of the evaluation process. Instead, some other systems like (Brusilovsky, 1994) and (Capell & Dannenberg, 1993), only deal with *lessons* intended as big learning objects fully explaining and assessing the knowledge about a given topic. In both cases the system generates sequences composed by only one kind of learning object. Only the most advanced systems, like (Brusilovsky, 1992) and (Khuwaja et al., 1996), are

* Corresponding author. Email: ncapuano@unisa.it.

able to generate sequences composed by different kinds of learning objects like presentations, tests, examples, etc.
LIA falls in this latter category given its ability to sequence several kind of *learning activities*. As we will see in 3.3, moreover, the concept of learning activity adopted by LIA is in itself an extension of the concept of learning object adopted by traditional ITSs. Models and methodologies behind LIA integrate and extend results coming from several researches on knowledge representation and ITS made by the authors, like (Gaeta et al., 2008), (Albano et al., 2006) and (Gaeta et al. 2008) only to name a few, some of which partially founded by the European Commission.
A first prototype of an intelligent system for learning, based on conceptual graphs and software agents and able to generate individualised learning courses was proposed by authors in (Capuano et al., 2000). Defined models and methodologies were then improved introducing description logic for knowledge representation, planning techniques for learning path generation and machine learning techniques for learning preferences discovery and adaptation leading to a new prototype described in (Capuano et al., 2002).
Obtained results convinced the authors to start the development of a complete system for learning (IWT) also involving a spin-off company (MoMA). The obtained system, providing a comprehensive set of "traditional" e-learning features and including a component (LIA) offering "intelligent" features coming from research was described in (Capuano et al., 2003).
This first version of the system had several limitations like: limiting domain model offering only a small and pre-defined set of relations between concepts, impossibility to specify any teaching preference connected to domain concepts, imprecise and computationally expensive algorithms for course generation applying no optimisation techniques, impossibility to consider global optimisation parameters like the total cost or duration of a learning experience, impossibility to deal with resources different from learning objects (i.e. lack of support for learning services), impossibility to revise the evaluation of concepts once they are considered as known by the system, lack of support for pre-test.
In order to overcome these limitations, models and methodologies applied by LIA have been fully reorganised and, in some cases, completely re-thought. The purpose of this paper is to describe improved models (chapter 2) as well as related methodologies and how they are used during the whole learning life-cycle (chapter 3). Architectural aspects related to the integration in IWT (chapter 4) and results of a small-scale experimentations with real users (chapter 5) are also presented.

2. LIA Models

The first step needed to describe the whole process of teaching/learning is to formally represent main involved actors and objects by means of appropriate models. The next paragraphs describe the four modes adopted by LIA while the next chapter shows how they are used in the teaching/learning process.

2.1. The Domain Model.

The domain model describes the knowledge that is object of teaching through a set of concepts (representing the topics to be taught) and a set of relations between concepts (representing connections among topics). Such structure can be formally represented with a ***concepts graph*** G (C, R_1,… R_n) where C is the set of nodes representing domain concepts and each R_i is a set of arcs corresponding to the i-th kind of relation. Two categories of relations are supported: *hierarchical* relations are used to factorise high-level concepts in low-

level concepts while *ordering* relations are used to impose partial orderings in the concept set. Without loss of generality in this paper we consider a concept graph G (C, BT, IRB, SO) with three relations BT, IRB and SO whose meaning is explained below (where a and b are two concepts of C):

- BT (a, b) means that the concept a *belongs to* the concept b i.e. b is understood iif every a so that a belongs to b is understood (hierarchical relation);
- IRB (a, b) means that the concept a *is required by* the concept b i.e. a necessary condition to study b is to have understood a (ordering relation);
- SO (a, b) means that the *suggested order* between a and b is that a precedes b i.e. to favour learning, it is desirable to study a before b (ordering relation).

Any number of additional relations may be introduced provided that they belong to one of the two categories above. The figure 1 shows a sample domain model in the didactics of artificial intelligence exploiting the relations defined above and stating that to understand "logics" means to understand "formal systems", "propositional logic" and "first order logic" but, before approaching any of these topics it is necessary to have an "outline of set theory" first. Moreover, "formal systems" must be taught before both "propositional logics" and "first order logic" while it is desirable (but not compulsory) to teach "propositional logics" before "first order logic".

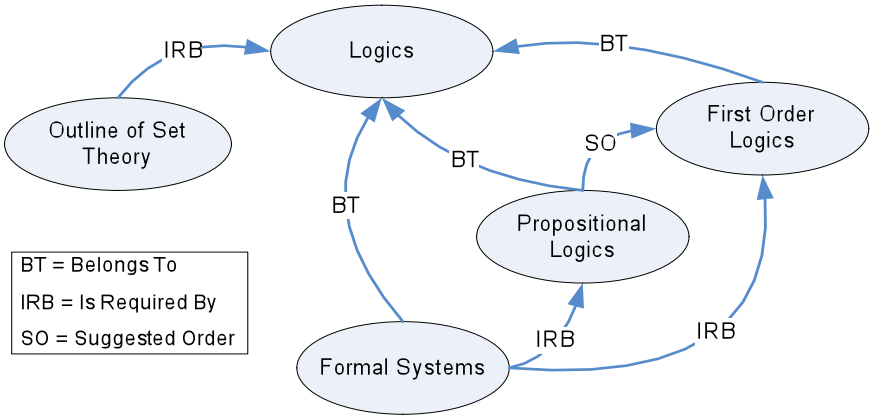

Figure 1. A sample concepts graph.

A set of ***teaching preferences*** may be added to the domain model to define feasible teaching strategies that may be applied for each available concept. Such preferences are represented as an application TP ($C \times Props \times PropVals$) $\rightarrow$ [0, 10] where $Props$ is the set of didactical properties and $PropVals$ is the set of feasible values for such properties. The table 1 provides some (non exhaustive) example of didactical property and associated feasible values. It is worth noting that TP is defined only for couples of $Props$ and $PropVals$ elements belonging to the same row in table 1.
As an example, the following definition for TP states that to teach formal systems a deductive didactic method is more suitable with respect to an inductive one while, to give an outline of set theory, both methods are equally good.

TP (“formal systems”, “didactic method”, “deductive”) = 10;
TP (“formal systems”, “didactic method”, “inductive”) = 4; (1)
TP (“outline of set theory”, “didactic method”, “deductive”) = 8;
TP (“outline of set theory”, “didactic method”, “inductive”) = 8.

Table 1. Example of didactical properties and feasible values.

Properties	**Feasible values**
didactic method	deductive, inductive, etc.
activity type	text reading, video clip, simulation, discussion with a peer, discussion with the teacher, etc.
interactivity level	high, medium, low

2.2. The Learner Model

The learner is the main actor of the whole learning process and it is represented with a cognitive state and a set of learning preferences. The ***cognitive state*** represents the knowledge reached by a learner at a given time and it is represented as an application $CS\ (C) \rightarrow [0, 10]$ where C is the set of concepts of a given domain model. Given a concept c, $CS\ (c)$ indicates the degree of knowledge (or grade) reached by a given learner for c. If such grade is greater then a given “passing” threshold θ then c is considered as known, otherwise it is considered as unknown. For example, assuming that $\theta = 6$, the following definition states that a given learner masters the outline of set theory but has a very poor knowledge of propositional logics.

CS (“outline of set theory”) = 8; (2)
CS (“formal systems”) = 4.

The ***learning preferences*** provide an evaluation of learning strategies that may be adopted for a given learner. They are represented as an application $LP\ (Props \times PropVals) \rightarrow [0, 10]$ where *Props* and *PropoVals* are the same sets defined in 3.1. Differently from teaching preferences, learning preferences are not linked to a domain concept but refer to a specific learner. As an example, the following definition states that a given learner prefers an inductive didactic method on a deductive one.

LP (“didactic method”, “deductive”) = 5; (3)
LP (“didactic method”, “inductive”) = 8.

The cognitive state of any learner is initially void (i.e. $CS\ (c) = 0$ for any c included in a given domain model) and may be initialized on a teaching domain with a pre-test. Learning preferences may be initialized by the teacher or directly by learners through a questionnaire capable of evaluating learners styles and transform them in suitable values for learning preferences. Both parts of the learner model are automatically updated during learning activities by a “learner model updating algorithm” (see 3.4).

2.3. The Learning Activity Model

A learning activity represents an activity that must be performed by a learner to acquire one or more domain concepts. According to IMS-LD specifications (IMS, 2003), activities are performed in environments that may be either learning objects (e.g. textual lessons, presentations, videoclips, podcasts, simulations, exercises, etc.) or learning services (e.g. virtual labs, wikis, folksonomies, forums, etc.). The "learning presentation generation algorithm" (see next section) uses learning activities as building blocks to generate learning experiences so the structure of a single learning activity is out of the scope of this paper. In order to be effectively used as a building block, a learning activity A is described through the following elements:

- a set of ***concepts*** C_A part of a given domain model, that are covered by in the learning activity (only leaf concepts with respect to the BT relation can be included in C_A i.e. learning activities are associated with leaf concepts);
- a set of ***didactical properties*** expressed as an application DP_A (*property*) = *value* representing learning strategies applied by the learning activity;
- a set of ***cost properties*** expressed as an application CP_A (*property*) = *value* that must be taken into account in the optimisation process connected with the "learning presentation generation algorithm".

Didactical properties components have the same meaning with respect to teaching and learning preferences i.e. *property* and *value* may assume values from a closed vocabulary (see table 1). Differently from learning and teaching preferences, they are neither linked to a domain concept nor to a specific student but to a learning activity. As an example, the following assertions state that the activity A is a simulation that uses an inductive didactic method and have an high interactivity level.

$$\begin{aligned} &DP_A(\text{“didactic method”}) = \text{“inductive”};\\ &DP_A(\text{“activity type”}) = \text{“simulation”};\\ &DP_A(\text{“interactivity level”}) = \text{“high”}. \end{aligned} \tag{4}$$

Cost properties are couples that may be optionally associated to learning activities, whose properties may assume values from the closed vocabulary {price, duration} and whose values are positive real numbers representing, respectively the eventual price of a single learning resource and its average duration in minutes. As an example, the following assertions state that the price of an activity A is 1.50 € while its average duration is 5 minutes.

$$\begin{aligned} &CP_A(\text{“price”}) = 1.5;\\ &CP_A(\text{“duration”}) = 5. \end{aligned} \tag{5}$$

Testing activities are learning activities used to verify the knowledge acquired by the learner. As learning activities, a testing activity T is connected to a set C_T of covered concepts indicating, in this case, the list of concepts verified by the activity. Once executed by a specific learner, they return an evaluation E_T belonging to the range [0, 10] indicating the degree of fulfilment of the test by that learner. Differently from other activities here DP_T and CP_T are not significant.

2.4. The Unit of Learning

An unit of learning represents a sequence of learning activities needed to a learner in order to understand a set of target concepts in a given domain with respect to a set of defined cost constraints. It is composed by the following elements:

- a set of ***target concepts*** *TC* part of a given domain model, that have to be mastered by a given learner in order to successfully accomplish the unit of learning;
- a set of ***cost constraints*** *CC* (*property*) = *value* that must be taken into account in the optimisation process connected with the "learning presentation generation algorithm" (see next section);
- a ***learning path*** *LPath* (c_1, …, c_n) i.e. an ordered sequence of concepts that must be taught to a specific learner in order to let him master target concepts;
- a ***learning presentation*** *LPres* (a_1, …, a_m) i.e. an ordered sequence of learning activities that must be presented to a specific learner in order to let him/her master the target concepts.

While target concepts and cost constraints are defined by the course teacher (in case of supervised learning) or by the learner himself (in case of self-learning), the learning path and the learning presentation are calculated and updated after each testing activity by specific generation algorithms described in the next section basing on the domain model, on the learner model associated to the target learner and on available learning activities. Concerning cost constraints, the *property* may assume values from the closed vocabulary {price, duration}. Feasible values are positive real numbers representing, respectively the maximum total price and the maximum total duration of the unit of learning.
As an example the following assertions state that the system have to build an unit of learning explaining the concept of "logics" that have a maximum total price of 100€ and during a maximum time of 6 hours (= 360 minutes).

$$\begin{aligned} &TC = \{\text{“logics”}\}; \\ &CC\ (\text{“price”}) = 100; \\ &CC\ (\text{“duration”}) = 360. \end{aligned} \tag{6}$$

3. The Learning Life-Cycle

After having seen how the main involved actors and objects are represented by means of appropriate models, it is necessary to see how LIA uses such models to automate some of the phases of the teaching/learning process. LIA sees the learning life-cycle as composed by five phases as shown in figure 2. Each phase includes one or more activities that have to be performed by the actors involved in the process (namely the teacher and the learner) or by LIA itself.

- In the ***preparation phase*** the teacher defines or selects a feasible domain model and prepares or selects a set of learning activities that may be used in the learning experience while learners may define their learning preferences.
- In the ***starting phase*** the teacher initializes a unit of learning by setting target concepts and cost constraints and associates one or more learners to it (in self-directed learning, learners settle own

target concepts and constraints by themselves). Then LIA generates a personalised *learning path* for each learner through a "learning path generation algorithm" described in 3.1 and then introduces placeholders for testing activities (named *milestones*) through a "milestone setting algorithm" described in 3.2.

- In the ***execution phase***, LIA selects a fragment of the learning path and generates the best *learning presentation* for each enrolled learner by applying the "learning presentation generation algorithm" described in 3.3. The learner then undertakes the learning and testing activities of the learning presentation until its end.
- In the ***evaluation phase***, when the learner ends a learning presentation fragment, his/her learner model is updated on the basis of the results of testing activities included in the fragment according to the "learner model updating algorithm" described in 3.4 and a new execution phase starts by generating a new learning presentation fragment that will possibly include recovery activities for concepts that the student did not understand.
- In the ***closure phase***, once all concepts of the unit of learning are mastered by the learner, the system collects statistical information on the process and the teacher may use this information as a basis for possible improvements of the domain model and/or of learning and testing activities.

The following paragraphs describe in more details the algorithms exploited by LIA in the several phases of the learning life-cycle (un-dotted boxes in figure 2).

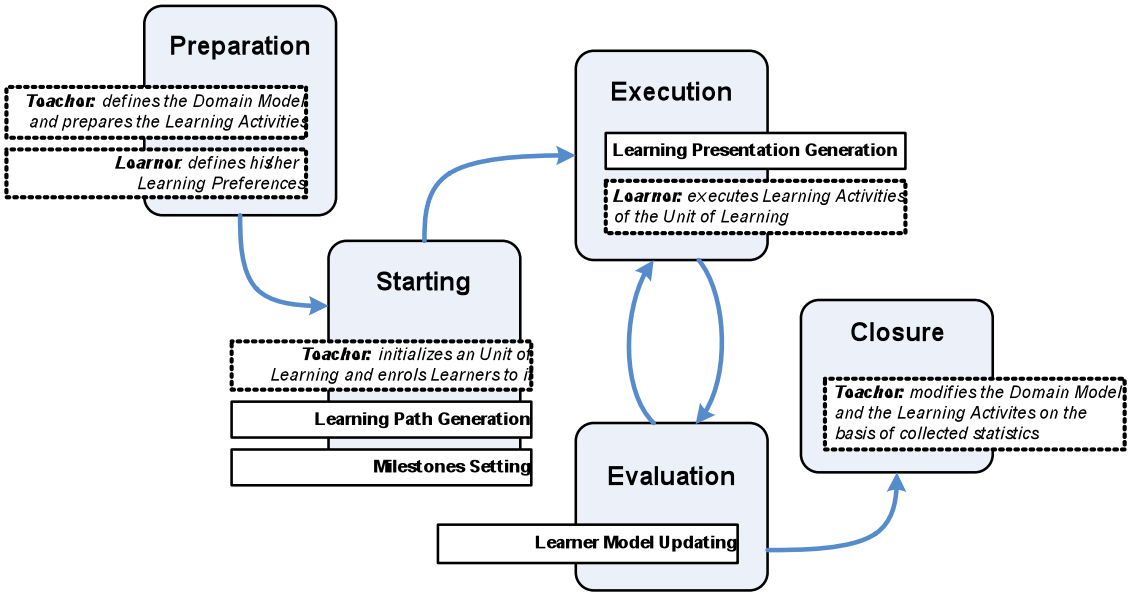

Figure 2. The Learning Life-Cycle.

3.1. Learning Path Generation

The generation of the learning path is the first step to completely generate a unit of learning. Starting from a set of ***target concepts*** *TC* and from a ***domain model***, a feasible learning path must be generated taking into account the *concepts graph* $G(C, BT, IRB, SO)$ part of the domain model (with $TC \subseteq C$). The four steps of the learning path generation algorithm are summarizes below.

- The ***first step*** is purposed to build the graph $G'(C, BT, IRB', SO')$ by propagating ordering relations downward the hierarchical relation. IRB' and SO' are initially set to IRB and SO respectively and, by applying the theory of partial ordered sets, they are modified by applying the following rule: for each arc $ab \in IRB' \cup SO'$ we have to substitute it with arcs ac for all $c \in C$ such that there exist a path from c to b on the arcs from BT.
- The ***second step*** is aimed to build the graph $G''(C', R)$ where C' is the subset of C including all concept that must be taught according to TC i.e. C' is composed by all nodes of G' from which there is a ordered path in $BT \cup IRB'$ to concepts in TC (including target concepts themselves). R is initially set to $BT \cup IRB' \cup SO'$ but all arcs referring to concepts external to C' are removed.
- The ***third step*** finds a linear ordering of nodes of G'' by using depth-first search so by visiting the graph nodes along a path as deep as possible. The obtained list L will constitute a first approximation of the learning path.
- The ***fourth step*** generates the final learning path $LPath$ by deleting from L all non-atomic concepts with respect to the graph G i.e. $LPath$ will include any concept of L apart concepts b so that $ab \in BT$ for some a. This ensures that only leaf concepts (i.e. concepts having some associated learning activity) will be part of $LPath$.

As a first example we may consider the concept graph in figure 1 as G and the set {"Logics"} as TC. The figure 3 (first case) shows the intermediate and final results of the learning path generation algorithm on these inputs carrying out to the following result stating that, to understand logics, the learner has to learn the outline of set theory, then formal systems, then propositional logics and, finally, first order logics:

$LPath$ = ("Outline of Set Theory", "Formal Systems", "Propositional Logics", "First Order Logics"). (7)

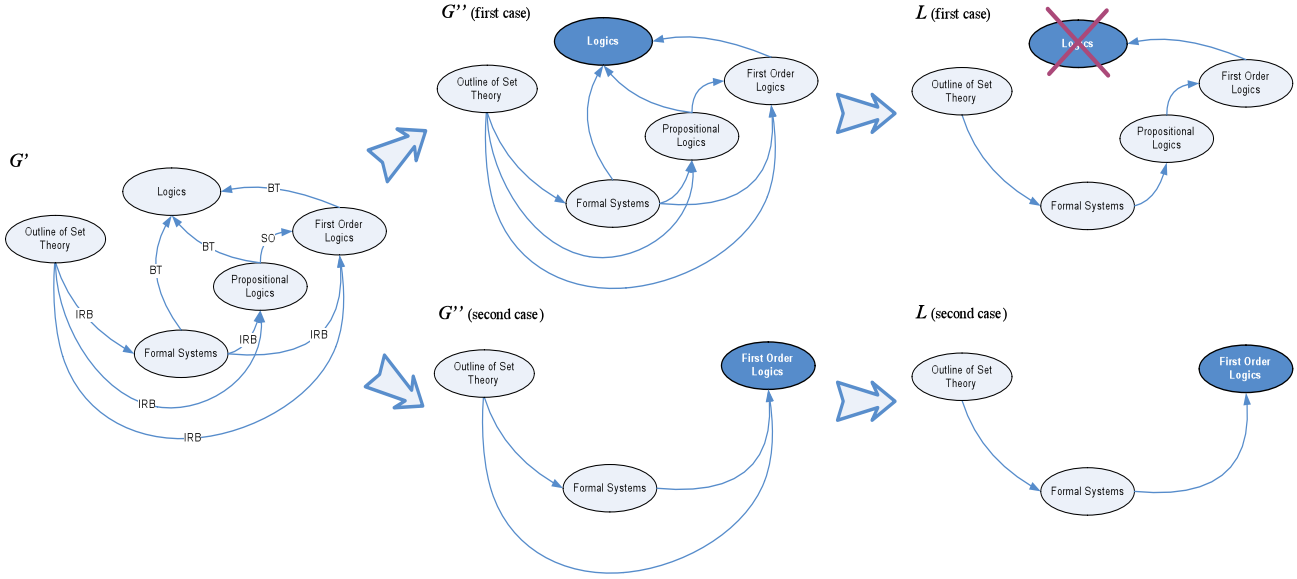

Figure 3. Examples of application of the learning path generation algorithm.

As a second example we may consider the same concept graph G while setting TC = {"First Order Logics"}. Algorithm results in this case are also shown in figure 3 (second case) and reported below:

$LPath$ = ("Outline of Set Theory", "Formal Systems", "First Order Logics"). (8)

It is important to note that the algorithm converges iif G is acyclic. If this condition cannot be guaranteed then an additional *step zero* must be added in order to remove cycles in G. Several algorithm may be applied to do that. The condition that must be respected is that, for each detected cycle, arcs are discarded starting from the less significant (arcs belonging to SO are less significant then arcs belonging to IRB that are less significant then arcs belonging to BT) until the cycle disappears.

3.2. Milestones Setting

Once the learning path is generated, it is necessary to insert in it placeholders for testing activities named ***milestones***. While concepts of the learning path will be converted in learning activities different from tests by the presentation generation algorithm, milestones will be converted in testing activities by the same algorithm. Several different possibilities exist to place milestones on the learning path:

- they can be placed directly by teachers;
- they can be placed basing on a list of percentages given by the teacher (e.g. the input list [0.2, 0.5, 0.7, 1] means that four milestones should be placed in the learning path approximately at 20%, 50%, 70% and at the end);
- they can be placed dynamically after each sub-list of concepts belonging to the same higher-level concept following the hierarchical relation (i.e. after each sub-list of concepts $c_1, \ldots, c_n$ so that $c_i a \in BT$ for some a).

Each milestone covers all preceding concepts in the learning path until the beginning of the course apart concepts already known by the learner according to his/her cognitive state (i.e. any concept a so that $CS(a) \geq$ the "passing" threshold θ as defined in 3.2). As an example, starting from the learning path $LPath$ given in (7) and the cognitive state CS defined in (2), a list of percentages [0.5, 1] carries out to the following updated learning path where the first milestone M_1 covers the concept "Formal Systems" (given that "Outline of Set Theory" is considered already known by the learner) while the second milestone M_2 covers the concepts "Formal Systems", "Propositional Logics" and "First Order Logics":

$LPath'$ = ("Outline of Set Theory", "Formal Systems", M_1, "Propositional Logics",
"First Order Logics", M_2) (9)

A milestone at 0% of the learning path is a special milestone meaning that a ***pre-test*** is necessary before entering in the unit of learning. Differently for other milestones, pre-test milestones may be of three different kinds:

- M_R indicates a ***pre-test on requirements*** testing concepts of the learning path considered as known by the learner and it is purposed to verify if such knowledge is still retained by the learner before entering in the unit of learning;
- M_C indicates a ***pre-test on content*** testing concepts of the learning path considered as unknown by the learner and is purposed to discover any possible further known concepts not stored in the cognitive state yet and to adapt the unit of learning accordingly in order to avoid to re-explain such known concepts;
- M_I indicates an ***integrated pre-test*** including both the pre-tests above so testing all concepts of the learning path.

As other milestones, pre-test milestones may be placed by teachers or by an algorithm based on a list of percentages. As other milestones they are converted in testing activities by the presentation generation algorithm.

3.3. Learning Presentation Generation

The presentation generation algorithm is purposed to build a fragment of presentation, part of an unit of learning, suitable for a specific learner basing on a ***learning path*** *LPath'* that have to be covered, on a set of teaching preferences *TP* belonging to a ***domain model***, on a cognitive state *CS* and a set of learning preferences *LP* both part of the ***learner model*** associated to the target learner, on a set of optional ***cost constraints*** *CC* and on a set of available ***learning activities*** (including tests). The three steps of the presentation generation algorithm are summarizes below.

- The ***first step*** is to select the sub-list *L* of *LPath'* that have to be converted in a presentation. *L* is the sequence of all the concepts of *LPath'* not already known by the learner (i.e. any concept *a* so that $CS(a) < \theta$) from the beginning to the first milestone preceded by at least one concept not already known by the learner. If *L* is empty then the algorithm ends because the learner already knows all concepts of the learning path.
- The ***second step*** is to define the best sequence of learning activities *P*, selected from available learning activities (not including tests), covering *L* on the basis of *TP*, *LP* and *CC*. This requires the resolution of an optimisation problem that will be discussed later.
- The ***third step*** is to add testing activities at the end of *P* so obtaining the final learning presentation *Pres*. Testing activities are selected in order to cover all concepts of *L* without taking into account didactical and cost properties eventually linked to testing activities (as said in 3.3 they are not significant for tests).

It is important to note that, in the case that *LPath* starts with a ***pre-test*** milestone (M_R, M_C or M_I), then *L* will be defined in order to include all concepts that should be tested according to the kind of pre-test milestone (as defined in 3.2), *P* is then settled to be void and only the third step is executed. Then the pre-test milestone is removed from *LPath*.

Let's give more details about the second step. Its purpose is to find the optimal set of learning activities *P* covering *L* on the basis of *TP*, *LP* and *CC*. First of all a measure of distance $d_{TP}(A, c)$ between an activity *A* and the set of preferences *TP* has to be defined with respect to a concept *c*. In a similar way a measure of

distance $d_{LP}(A)$ basing on *LP* may be defined. A further measure $d(A, c)$ shall be defined as a weighted sum of the two measures. The three formulas for the three measures are given below where each $\alpha_{property}$ is a constant in [0, 1] representing the weight that the single property (among those listed in table 1) have in the calculation of distances and β_{TP} and β_{LP} are positive real constants summing to 1 weighting teacher and learner preferences in the calculation of the overall distance.

$$d_{TP}(A,c) = \sum_{property} \alpha_{property}(10 - level) \mid DP_A(property, value) \wedge TP(c, property, value, level) \text{ if } c \in C_A, \infty \text{ otherwise;} \quad (10)$$

$$d_{LP}(A) = \sum_{property} \alpha_{property}(10 - level) \mid DP_A(property, value) \wedge LP(property, value, level); \quad (11)$$

$$d(A, c) = \beta_{TP}\, d_{TP}(A, c) + \beta_{LP}\, d_{LP}(A). \quad (12)$$

Once the measure of distance is defined the problem of selecting the best set of activities *P* covering concepts of *L* becomes a *facility location problem* (Shmoys et al., 1997) that may be outlined with the bi-partite graph in figure 4 where available activities are displayed on the left and concepts to be covered on the right. *P* must be built as the smallest set of activities covering all concepts of *L* with the minimum sum of distances between activities and covered concepts. This translates in the following linear programming problem where y_A is a binary variable assuming the value 1 if the activity *A* is used and 0 otherwise and x_{Ac} is a binary variable assuming the value 1 if the concept *c* is covered by the activity *A* and 0 otherwise:

$$\min \sum_A y_A + \sum_{A,c} d(A,c) x_{Ac} \text{ so that:} \quad (13)$$

$$\sum_A x_{Ac} = 1 \quad \forall c; \quad (14)$$

$$x_{Ac} \le y_A \; \forall A, c; \quad (15)$$

$$x_{Ac} \in \{0,1\} \; \forall A, c \text{ and } y_A \in \{0,1\} \; \forall A. \quad (16)$$

The constraint (14) imposes that each concept must be covered by just one learning activity while the constraint (15) imposes that a concept may be explained only by a learning activity among those available. Finally the constraint (16) states that *x* and *y* are binary variables. In order to take into account also cost constraints of *CC*, for each *CC* (*property*) = *value*, a new constraints like the following one must be added:

$$\sum_A value_A y_A \le value \quad \mid CP_A(property, value_A) \quad (17)$$

meaning that the sum of all the costs associated to selected learning activities for each constraining property must be taken below then the maximum cost stated by *CC*. To solve this problem it is possible to use a greedy algorithm to obtain a first feasible solution and, then, a local search algorithm to try to improve the initial solution. Given that these algorithm are well know in the literature (Shmoys et al., 1997), their description of such algorithms is out of the scope of this paper.

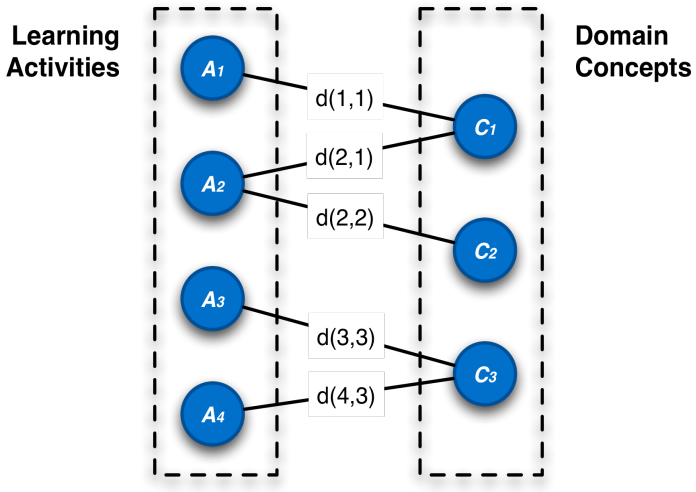

Figure 4. Facility Location Problem connected with the Learning Presentation Generation.

3.4. Learner Model Updating

For each testing activity T executed by the learner in the last learning presentation fragment, the test returns (as explained in 3.3) an evaluation E_T between 1 and 10 representing the degree of fulfilment of the test by the involved learner. For each concepts c belonging to C_T, the cognitive state of the learner is modified in this way: if $CS(c)$ is not defined then $CS(c) = E_T$, otherwise $CS(c) = (CS(c) + E_T) / 2$. This is repeated for any executed testing activity T in *LPres*.

An optional procedure consists in propagating the evaluation of each concept over required concepts in the concepts graph following backward the ordering relation *IRB*. Some times in-fact iterated failures to understand a set of concepts indicates a possible misconception in a common requirement. Such procedure helps to find these implicit misconceptions and forces the learning presentation generation algorithm to introduce activities covering such misconceptions in subsequent fragments by decreasing their grade in the learner cognitive state.

To apply this optional procedure, the grade of each concept c' so that there is an ordered path in *IRB'* (see 3.1) from c' to c should be modified in the learner cognitive state as follows: $CS(c') = (\alpha \cdot E_T + (n - \alpha) \cdot CS(c')) / n$ where α is a constant in (0,½] representing the intensity of the propagation while n represents the number of the arcs of the path between c' and c.

After a testing phase, learning preferences may be also modified in the following way: if the same learning activity A has been proposed n times to the learner (for a given n) without success (so bringing to a negative score in testing activities on related concepts), then the system decreases, for each didactical property DP_A (*property*) = *value* associated with A, learner preferences LP (*property*, *value*) of a given constant δ. In this way the learning activity A as well as learning activities with similar didactical properties will be less often selected in future learning presentations. Conversely learning activities that demonstrate to be successful (so bringing to positive scores in testing activities) will increase learner preferences related to activities' didactical properties. In this way, similar activities (i.e. activities with similar didactical properties) will be more often selected in future learning presentations.

4. Integration in IWT

As anticipated, LIA models and methodologies are integrated in a complete e-learning solution named Intelligent Web Teacher (IWT) whose aim is to fill the lack of support for flexibility and extensibility in existing e-learning systems. IWT arises from the consideration that every learning/training context requires its own specific solution. It is not realistic to use the same application for teaching, for instance, foreign languages at primary schools, mathematical analysis at university and marketing management to enterprise employees.

It should be not only the content to vary but also the didactic model, the typology of the training modules to be used, the application layout and the supporting tools. In practice, the need to introduce the e-learning in a new learning/training context brings to hard work for analysts, engineers and programmers. IWT solves this problem with a modular and extensible solution able to become the foundation for building up a virtually infinite set of applications for e-learning. IWT logical architecture is divided in three main layers as shown in figure 5.

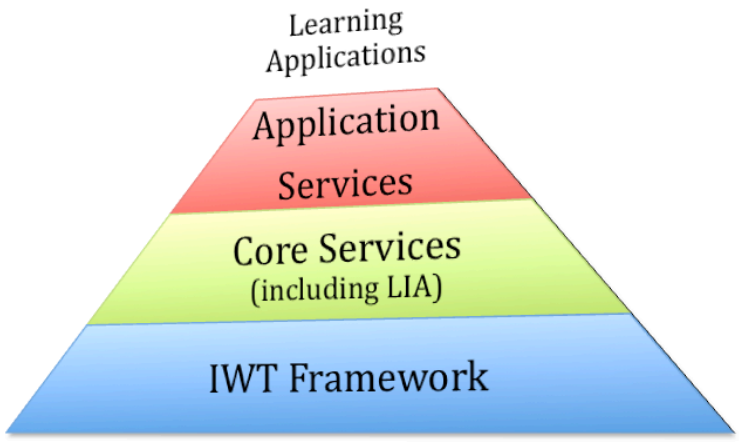

Figure 5. The three layers composing the IWT Logical Architecture.

The first layer at the bottom of the stack is the ***IWT Framework*** used by developers to design and implement core services, application services and learning applications. The second layer is composed by ***Core Services*** providing basic IWT features like resources and workflows management, information extraction, ontology storing, user authentication, content storing, metadata, role and membership management, learning customisation (i.e. LIA), logging, profiling and data mining. Core services are used by application services and learning applications. ***Application Services*** are used as building blocks to compose e-learning applications for specific domains. They include document management, conferencing, authoring, learning management, learning content management, ontology management, communication and collaboration, business intelligence, process management and information search services.

On the top of such architecture, ***Learning Applications*** covering specific learning scenarios obtained as integration of application services are built. Such architecture is modular enough to allow the deployment of solutions capable to cover application scenarios of different complexity and for different domains by composing service building blocks. Moreover, by relying on LIA, IWT is able to apply the whole learning

life-cycle described in chapter 3 including the generation of individualised learning experiences based on learner needs and preferences as well as the evaluation of learners with respect to acquired knowledge and shown learning preferences.
To improve interoperability with external systems, IWT adopts several standards and specifications in the e-learning field (i.e. IEEE Learning Object Metadata, ADL SCORM, IMS Question and Test Interoperability, Learner Information Packaging and Learning Design) and in the knowledge representation field (OWL is used to represent LIA models).

5. Exerimental Results

IWT is currently used by about 30 Italian organisations (more then 40.000 users) including big companies like Atos Origin Italy, Italdata, and Metoda as well as departments and faculties from the Universities of Roma "La Sapienza", Milano "Bicocca", Salerno, Bari, Calabria and Napoli "Parthenope". IWT was also selected by the Italian Ministry of Education as the base technology for the project "DigiScuola" purposed to experimentally introduce e-learning in 550 Italian schools involving 3000 teachers and 33000 learners. In such contexts a small-scale experimentation was performed to demonstrate the benefits of LIA as an add-on in on-line learning.
Such experimentation involved a group of 28 voluntary learners belonging to 7 Small and Medium Enterprises dealing with vocational training on enterprise management. The group of learners was split in two separate sub-groups: a first sub-group composed of 20 learners was enabled to use all IWT facilities except LIA while a second sub-group composed of 8 learners was enabled to access the whole IWT (including LIA).
All the voluntary learners were tested before and after a training phase on the same topics. In all the tests the learners' skills in the chosen domain were quantified using three ability ranges: low-level (0-3 scores), medium-level (4-7 scores) and high-level (8-10 scores). The figure 7 shows the performances of the two sub-groups. As it can be seen, the progress made by the second group of students is much sharper with respect to the first group.

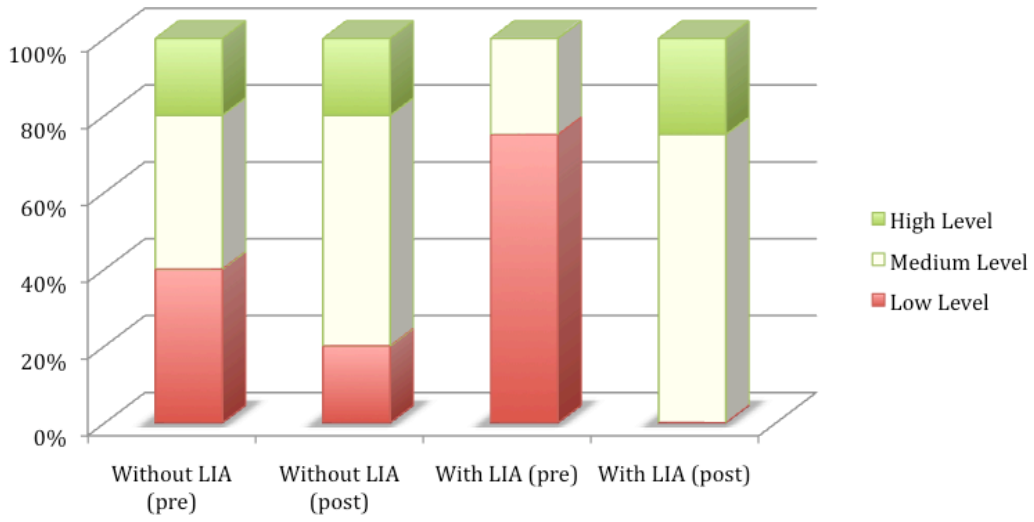

Figure 7. Experimentation results on a group of 28 learners with and without LIA.

Besides these tests, interviews to different platform's users were also performed in order to evaluate the overall system's friendliness and the user satisfaction level. Beyond the 28 learners already mentioned, 7 worker-supervisors (one work-supervisor for each of the seven involved companies), the teacher (only one teacher was involved for all learners) and a training expert (responsible of the domain model creation) were also interviewed. Table 2 summarizes some of the questions included in the used questionnaires with the corresponding results. From Table 2 it arises that most of the interviewed people (65.39 %) felt satisfied with respect to the tool's navigation and interaction facilities.

Table 2. Results of interviews.

	Yes, very much	**Yes, sufficiently**	**Not enough**	**Not at all**	**No answer**
Is the tool versatile and suitable for your needs?	15.38	42.31	30.77	11.54	0
Was the audiovisual environment clear?	42.31	30.77	3.85	11.54	11.54
Did you have any problem in the installation phase?	7.69	11.54	7.69	73.08	0
Did you feel satisfied with the tool's navigation and interaction facilities?	42.31	23.08	23.08	11.54	0
Have you had any technical problem which made the learning difficult?	15.38	7.69	15.38	61.54	0

6. Conclusions and Future Work

In this paper we have described an intelligent tutoring engine named LIA and how it cooperates with a complete system for e-learning named IWT in the provisioning of individualized and personalized e-learning experiences. Defined algorithms and underlying models have been described as well as architectural aspects related to the integration in IWT. Results of a small-scale experimentation with real users have been also presented and demonstrate the benefits of LIA as an add-on in on-line learning. A large-scale experimentation is still in course and results will be published in a future paper.

Thanks to the innovative features provided by LIA, IWT is currently adopted by about 30 Italian organisations (with more then 40.000 users) and in November 2007 it won the prize "Best Practices for Innovation" from the General Confederation for Italian Industry. Despite that, the research on LIA is still active and further improvements are under study.

Among the other things, memetic algorithms for learning presentation generation are under study (Acanfora et al., 2008) as well as algorithms and tools for the semi-automatic extraction of domain models starting from pre-existing learning material. Further research work purposed to improve the support for didactic methods in the domain model (Albano et al., 2007) as well as the support for learning styles in the learner model (Sangineto et al., 2008) is in-progress. A visionary hypotheses of extension of LIA to fully support enterprise learning has been also recently proposed (Capuano et al., 2008).

References

Acanfora, G., Gaeta, M., Loia, V., Ritrovato, P. & Salerno, S. (2008). Optimizing Learning Path Selection through Memetic Algorithms. *Proceedings of IEEE World Congress on Computational Intelligence, June 1-6 2008, Hong Kong*.

Albano, G., Gaeta, M. & Ritrovato, P. (2007). IWT: an innovative solution for AGS e-Learning model. *International Journal of Knowledge and Learning, vol. 3, nos. 2/3*, pp. 209-224.

Albano, G., Gaeta, M. & Salerno, S. (2006). E-learning: a model and process proposal. *International Journal of Knowledge and Learning, Inderscience Publisher, vol. 2, no. 1/2*, pp. 73-88.

Brusilovsky P. (1994). ILEARN: an intelligent system for teaching and learning about UNIX. *Proceedings of SUUG International Open Systems Conference, Moscow, Russia, ICSTI*, pp. 35-41.

Brusilovsky, P. (1992). A framework for intelligent knowledge sequencing and task sequencing. *Intelligent Tutoring Systems, Springer-Verlag, Berlin*, pp. 499-506.

Brusilovsky, P. & Vassileva, J. (2003). Course sequencing techniques for large-scale web-based education. *International Journal of Continuing Engineering Education and Lifelong Learning, vol. 13, nos. 1/2*, pp. 75-94.

Capell, P. & Dannenberg, R. (1993). Instructional design and intelligent tutoring: theory and the precision of design. *Journal of Artificial Intelligence in Education, vol. 4, no. 1*, pp. 95-121.

Capuano, N., Gaeta, M. & Micarelli, A. (2003). IWT: Una Piattaforma Innovativa per la Didattica Intelligente su Web. *AI*IA Notizie, year XVI, no. 1,* pp. 57-61.

Capuano, N., Gaeta, M., Micarelli, A. & Sangineto, E. (2002). An Integrated Architecture for Automatic Course Generation. *Proceedings of the IEEE International Conference on Advanced Learning Technologies, September 9-12, 2002, Kazan, Russia. V. Petrushin, P. Kommers, Kinshuk, I. Galeev eds. pp.* 322-326.

Capuano, N., Gaeta, M., Ritrovato, P. & Salerno, S. (2008). How to Integrate Technology Enhanced Learning with Business Process Management. *To appear on Journal of Knowledge Management. Emerald Group Publisher.*

Capuano, N., Marsella, M. & Salerno, S. (2000). ABITS: An Agent Based Intelligent Tutoring System for Distance Learning. *Proceedings of the International Workshop on Adaptive and Intelligent Web-Based Education Systems, ITS 2000, June 19-23, 2000, Montreal, Canada*, pp. 17-28.

Eliot, C., Neiman, D. & Lamar, M. (1997). Medtec: a web-based intelligent tutor for basic anatomy. *Proceedings of WebNet'97, World Conference of the WWW, Internet and Intranet, Toronto, Canada, AACE,* pp. 161-165.

Gaeta, A., Gaeta, M., Orciuoli, F. & Ritrovato, P. (2008). Defining a Service Oriented Architecture for e-Learning: The European Learning Grid Infrastructure Project Experience. *To appear on the International Journal of Technology Enhanced Learning. Inderscience Publisher.*

Gaeta, M., Miranda, S. & Ritrovato, P. (2008). ELeGI as enabling architecture for e-learning 2.0. *To Appear on the International Journal of Knowledge and Learning. Inderscience Publisher.*

IMS (2003). Learning Design Information Model. Form http://www.imsglobal.org/learningdesign.

Khuwaja, R., Desmarais, M. & Cheng, R. (1996). Intelligent guide: combining user knowledge assessment with pedagogical guidance. *Intelligent Tutoring Systems, Lecture Notes in Computer Science, Springer Verlag, Berlin, vol. 1086*, pp. 225-233.

MoMA (2006). *IWT White Paper*. From http://www.didatticaadistanza.com/brochure_iwt_eng.pdf.

Rios, A., Millán, E., Trella, M., Pérez, J. & Conejo, R. (1999). Internet based evaluation system. *Artificial Intelligence in Education: Open Learning Environments, IOS Press, Amsterdam*, pp. 387-394.

Sangineto, E., Capuano, N., Gaeta, M. & Micarelli, A. (2008). Adaptive Course Generation through Learning Styles Representation. *Universal Access in the Information Society International Journal, vol. 7, no. 1/2*, pp. 1-23.

Shmoys, D., Tardos, E. & Aardal, K. (1997). Approximation Algorithms for Facility Location Problems. *Proceedings of the 29th Annual ACM Symposium on Theory of Computing, El Paso, Texas, United States*, pp. 265-274.