An Architecture to Support Adaptive E-Learning

Vincenza Carchiolo† and Alessandro Longheu† and Michele Malgeri† and Giuseppe Mangioni†

† Dipartimento di Ingegneria Informatica e delle Telecomunicazioni**,** Viale A. Doria, 6 - I95100 Catania, ITALY

Summary
E-learning technology is currently pushed by several factors. Among these, the advantage of computer-based courses of sharing and reusing existing teacher materials, in addition to the possibility of personalizing courses according to students needs, preferences and capabilities. To provide such features, in this paper we introduce an architecture organized into four layers: a database layer to store, share and reuse courses and teaching materials, an adaption layer which allow personalized courses generation, a presentation layer that arrange personalized courses into learning paths, and an interface layer to develop several learning interfaces (e.g. for use via web or mobile devices). The architecture is presented, then we focus on its personalization capabilities by presenting details of how courses are actually tailored and attended by students.

Key words:
Adaptive Distance Learning, E-Learning, LMS, Web-Based Learning System.

1. Introduction

Several factors contributed to the development and adoption of E-Learning technologies; in particular, computer-based courses can be easily shared and reused by teachers, leading to an uniform set of topics and to a significant time saving during courses creation and management. Moreover, computer-based courses can be tailored to each single student capabilities and needs, both in terms of topics (concepts) as well as in terms of the related teaching materials. Finally, the development of web–based and mobile technologies improves the learning process providing a better integration of different media both during the creation and fruition of courses.

Based on these considerations, in this paper we present an e-learning framework aiming at:

1. Courses sharing and reusing, in order to improve their creation, management and attendance. This is performed at courses level, i.e. the set of topics is shared thus new courses can reuse existing ones whenever needed, whereas teaching materials are stored separately so they can be managed and associated independently from topics.
2. Promoting the adoption of personal students profiles, allowing courses to be personalized in terms of both topics and teaching materials, in contrast with traditional lessons, where the flow of information is mainly directed from the teacher to students. To promote such active learning, the system creates and proposes to the student all the possible paths starting from his knowledge and directed toward the desired knowledge (expressed as a list of topics of interest), filtering these paths by taking into account the student point of view, expressed in terms of personal time availability, desired learning style and so on.
3. Allowing dynamic adaption, for instance if the student does not possess initially declared knowledge, the system can detect this situation from the results of exercises and can re-evaluate and suggest a new path for the student; or the system may change the path if the student's preferences (e.g., his available time) are modified.
4. Providing advanced features, as the adaption of teaching materials to the devices features (media adaption), and also allowing the development of new learning services through a set of Application Programming Interface (API).

In particular, our work focuses on courses personalization, which is performed according to the following phases:

- first, a database contains all courses, each composed of single course units properly related through precedence-succession relationships, thus determining a graph of course units. A course unit represent a set of topics to be learned, and is associated to its corresponding teaching materials, stored in a separate database. Courses and teaching materials are inserted into databases by teachers, so several courses are available.
- the student register himself into the system, in order to provide personal information, used both for simple identification purposes (e.g., name, academic year of attendance), and to express student's preferences and characteristics, such as his knowledge (here expressed as a set of keywords belonging to a common ontology), available time, desired media and so on. All this information is stored into a personal profile, used to tailor courses to each single student.

Manuscript received January 5, 2007
Manuscript revised January 25, 2007

- when the student wants to attend a course, i.e. he wants to learn topics it includes, he provide the set of keywords representing the aimed knowledge (again, chosen from the common ontology); the course can already exist if it has been previously created by some teacher, otherwise the system builds the course on-the-fly by selecting all course units whose topics includes desired goals. In both cases, a graph of course units is actually selected; the system will discards all course units representing topics already included in that student's knowledge, and it will also retain just course unit needed to reach desired goals, thus determining a subgraph. Finally, a tree containing all possible learning paths for that student is derived from the subgraph. All such tasks are performed by a course generation module.
- subsequently, the student will follow one learning path from all those making up the tree previously generated; the choice is made according to that student personal preferences, i.e. each next course unit to be attended along his learning path is provided with a set of parameters based on personal profile information, so that the student can choose the next course unit most suitable to his needs. Moreover, since preferences can be changed even for each single course unit, the student is offered with a high degree of personalization. All these tasks are performed by the course presentation module.
- finally, the student can attend the chosen course unit, accessing to the associated teaching material. At the end of the learning phase, the student has to perform a test in order to get increased his personal knowledge with topics representing the attended course unit. If the test was performed successfully, the student can proceed along his personalized learning path, choosing the next course unit until the course is completed. If the test fails, the system (depending on test results) suggests the student to attend again the last course unit, but the system can also discover that previously acquired and/or declared knowledge is not confirmed (this also depends on the issues the test is about) thus suggesting the student to move back along his learning path, attending again previous course units. Such paths are still managed by the course presentation module.

In the following sections, we start from related work (section 2), then we introduce the architecture of e-learning framework in section 3, focusing on its support for personalizing courses in subsequent sections. Section 4 indeed describes student profiles, whose information take into account students needs and preferences used by course generation and presentation modules. Section 5 describes the course generation module, whereas course presentation is examined in section 6. We also present first evaluation results in section 7, finally presenting our conclusions and indications for future work.

2. Related work

Comparison of the work introduced in this paper with others is performed focusing on methodologies for courses adaption and personalization and related issues, being these topics the core of our work.

Considering underlying databases, the idea of separating topics and teaching materials we adopted is derived from Content Management Systems (CMS) area, whose principles are used in the E-learning context [1]. Two questions have to be addressed about teaching materials, i.e. the choice of the materials and its characterization; to facilitate such tasks several tools can be used, e.g. [2] propose a learning design framework to enhance learning resources reusability, helping teacher to find learning documents matching their needs, whereas characterization can be improved by using simple keywords, or adopting metadata standards as XML, RDF schemas and so on; more details can be found in [3], in particular in the related work; in this paper, materials are characterized through a set of properties illustrated in section 3.1.

In our learning approach, whose phases are outlined in section 1, the student is essentially autonomous during the whole learning process, i.e. we do not focus on direct interaction with the teacher nor with other students, although our proposal does not contrasts these interactions. In particular, the teacher's contribution essentially consists of providing educational guidelines to be followed by students, as indicated in [4]; in our work, this is accomplished when teacher creates courses; in other learning strategies, e.g. the cognitive apprenticeship approach [5][6], the teacher can help students also during the learning process. Interaction among students is known as "collaborative learning", a wide area where several work can be found; see for instance [7].

Adaptive systems, as defined in [9], "cater information to the user and may guide the user in the information space to present the most relevant material, taking into account a model of the users goals, interests and preferences"; these ideas can be schematized through a set of principles for adaptive learning that can be found in [10]. In an educational context, adaptivity to student needs (personalisation) is a key point for the success of an E-learning system, being one of the most significant improvement computer-based learning can provide with respect to the classical learning in a classroom, where a teacher must tailor the lesson to the students average capabilities and needs. Adaptivity is present in several other works; for instance, [11] proposes an author-defined storage for LMS (Learning Management Systems) capable

of providing adaptivity: the student is presented with an overview of a document containing material tailored to the information stored in the ”student model” (similar to the ”student profile” defined in this paper). [12] provide adaption by determining the most relevant learning path for every learner, also allowing course pages to be presented with a different look-and-feel according to user preferences (adaptive navigation). [13] also emphasizes adaption, named “appropriation”, proposing an approach to organize curriculum according to student’s personal objectives (similar to the ”desired knowledge” in our work), and knowledge (to propose different views of curriculum). Construction of personalized learning paths can be found in several other works, e.g. [14][15][16][17][18][19]. A comparison of some adaptive systems can be found in [20].

Our system exploits “knowledge” to build personalized path; actually, different types could be considered, such as declarative and procedural knowledge [21], the former representing a knowledge derived from theories (”facts” in [21]) and the latter a knowledge coming from practice (e.g. laboratory work). In other works, e.g. [22], the distinction between different types of knowledge reflects the classification of memories present in the human brain, i.e. semantic, episodic and procedural knowledge. In this paper, knowledge stored into personal profiles simply consists of a set of keywords indicating topics known by the student. Such keywords should be arranged into an ontology in order to provide a common set; we do not however address ontology issue here. The “initial” knowledge possessed by a student can be established using several approaches, as proposed in [23]: the student knows nothing or has some standard prior knowledge of the domain, or a pre-test is administered, or the system may use patterns to group similar students. [23] and [24] do not focus on knowledge only, but they address the issue of a correct and complete initialization of a student model.

Considering text and exercises, several evaluation methodologies has been proposed in literature, starting from the mid nineteen-sixties, when Project Essay Grade(PEG) [25] was developed. The PEG approach is based on the superficial surface features of an essay (number of commas, word count, etc.) as indicators of quality. This approach lacks into evaluation of text where the content is more important than the style of the essay. Another approach developed in the late 90s is Latent Semantic Analysis (LSA) [26], originally developed for indexing documents and text retrieval. Also this approach does not take into account grammar and word order and is not suitable to disciplines where text is very important. Electronic Essay Rater (e-rater) [26] is based on Microsoft Natural Language Processing Tool and is able to parse all sentences contained into the essay discovering the syntactical structure of the phrases. The role of exercises, as indicated in [27] and [28] is fundamental for feedback purposes, to allow both a student to verify acquired knowledge, and a teacher to evaluate student learning and the quality of the path followed.

Finally, our work cannot be compared with commercial web-based learning system support tools, as WebCT (www.webct.com), Blackboard (www.blackboard.com), LearningSpace (www.lotus.com), TopClass (www.wbtsystems.com) , since these tools indeed offer complete student and teacher interfaces to provide web-based course attendance, but they do not focus on course topics managements, and they do not consider how to reuse existing teaching material in similar

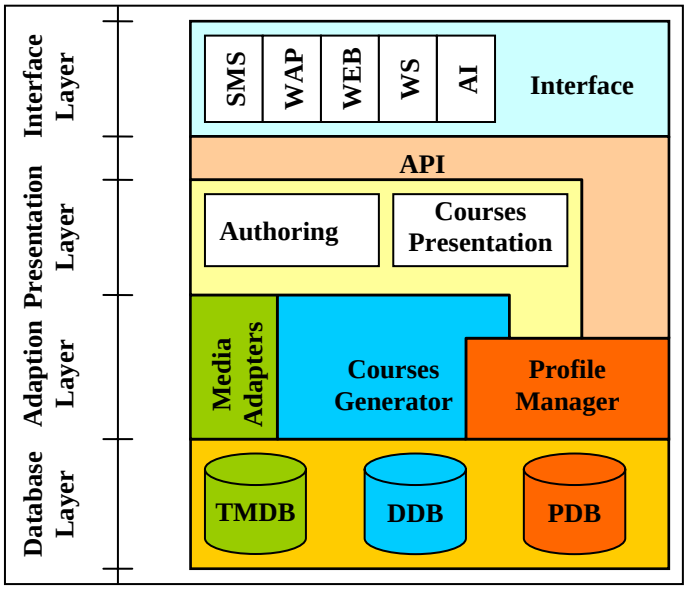

Figure 1

courses, nor how to personalize courses based on personal capabilities and needs each student provides.

3. E-Learning System Architecture

In this section we introduce the architecture of our e-learning system using a layer based model. In Figure 1 the proposed framework is shown. In particular it consists of four layers: Database, Adaption, Presentation & Authoring and Interface. In the following the four layers are described pointing out their structure and features.

3.1 Database Layer

The Database layer includes three databases.

Domain Database (DDB). The DDB contains information about the courses and units they consist of, regardless of the specific domain of interests, i.e. the DDB can store courses/units belonging to different, (possibly) unrelated domain of interest. A Course Unit (*CU*) is a logical unit representing a set of topics (concepts) to be learned. Each *CU* is described with a set of properties, which are:

- Title.
- Description.
- Requisites, that is the knowledge required to understand *CU* topics. Such knowledge is actually represented through a set of keywords extracted from a common ontology of a given domain of interest. Different ontologies may be actually available when several domain of interest are included in the DDB, though is beyond the scope of this work to address ontologies issues.
- Objectives, that is the knowledge that will be acquired by learning *CU*, again expressed through a list of keywords.
- A list of the teaching materials (*CM*) available for the *CU*.
- Creation, last access/modification dates, and *CU* creator identifier.

All the *CU*s stored in the DDB are organized in an **AND**/**OR** directed acyclic graph with the various nodes (*CU*s) connected by arcs representing whether one is preliminary to another. The orientation of the arcs between *CU*s is defined as follows: an arc oriented from *X* to *Y* means that node *X* depends on node *Y*; that is, if *X* depends on *Y*, all topics included in *Y* must be learned before topics of *X* can be understood. Two or more arcs involving the same node, for example the two arcs $Z \rightarrow X$ and $Z \rightarrow Y$ can represent alternative paths (either *X* or *Y* must be known in order to understand the topics in *Z*), or they may represent paths that are both necessary to understand the topics in the node *Z* to which they refer. The two situations are modelled as **OR** and **AND** arcs respectively. This feature will be dealt with in more detail in Section 4 to build learning paths.

The second entity stored in the DDB is the Course Node (*CN*), which represent a set of *CU*s. Similarly as for *CU*s, each *CN* comes with a set of properties, which are:

- Title and description.
- Requisites. These are defined as the union of all requisites of those *CU*s that have no predecessors in the graph of that *CN*, i.e. the union of "starting" *CU*s requisites;
- Objectives. Similarly to requisites, these are the union of all objectives of "final" *CU*s (with no successor);
- A list of all *CU*s making up the course. Such list actually determines a sub-graph extracted from the **AND**/**OR** graph of *CU*s placed at the unit level.
- Finally, creation and last access/modification dates, and *CN* creator identifier.

Teaching Material Database (TMDB). It contains all the teaching material (*CM*) used in the various stages of a course/lesson (presentation, testing, etc...), and generally comprises multimedia and/or hypertext material (HTML pages, animated graphics, etc...). A given *CU* can include different sets of *CM*s, each set is a number of *CM*s that allows to learn the objectives of the associated *CU*, starting from its requisites. Sets can differ for their internal arrangement and for the following properties:

- Timing, i.e. total estimated length of time required for learning concepts belonging to the *CU*.
- Level of abstraction of the *CU* (i.e. highly theoretical, mostly practical...).
- Level at which topics are dealt with (introductory, in-depth treatment, for specialists...).
- Level of detail with which topics are dealt with (general overview of problem, details of specific problems). It should be noted that the level of detail and the general level at which topics are dealt with may overlap, although there may be *CU*s on specialist topics with a low level of detail (e.g. a survey of a specialist problem).

For instance, a set could actually consists of a single *CM* (e.g., a set of slides), whereas another set could include more *CM*s (e.g., slides plus tutorial files and exercises); clearly, sets (consequently, all *CM*s they include) must have been designed to share the same requisites/objectives.

Profile Database (PDB). It contains all information about students (e.g. owned knowledge, available time), used to build course tailored both in terms of topics and material. Students profiles are discussed in Section 4.

3.2 Adaption Layer

At this level media adaption and course personalization are performed, using the following modules:

a. Media Adapters Module. It adapts teaching materials to the client device (PC, Pocket PC, SmartPhone, Mobile Phone, …).
b. Courses Generation Module. It is devoted to build courses and learning paths, based on lessons/courses contained in the DDB, as well as on information in the PDB (see Section 5).
c. Profile Manager Module. It manages students profiles, interacting directly with PDB (see Section 4).

3.3 Presentation & Authoring Layer

It consists of:
a. Authoring Module. It offers a set of tools for the managements of courses, lessons and teaching materials.
b. Courses Presentation Module. As soon as courses are built, the course presentation module retrieves the related teaching materials and arranges course units into learning paths according to profile information. The Course Presentation Module is discussed in Section 6.

3.4 Interface Layer

Using API several learning services can be implemented. The system, for examples, can provide a WWW learning service (via the WEB module), a learning interface based on Web Services (WS), or a set of interfaces for mobile devices, as GSM Short Messages Services (SMS) or WAP. The modularity of the proposed framework helps to easy extend the functionalities of the system, adding, as an example, new services or new media converters.

4. Profiles management

As regards the information to be stored in the student profile database, there are several works in the relevant literature [29][30]. Some are more pedagogically and/or psychologically oriented, as [31][32], while others focus on student collaborative learning ([33][7]). Starting from their considerations, the student profiles we propose in order to allow personalized learning paths generation is mainly used to describe the student knowledge and his preferences. Specifically, we define the student profile as a 3-uple:

$$\mathbf{StudentProfile = \{GI,\ CI,\ SI\}}$$

Given a student ST, we refer to his profile as $\mathbf{StudentProfile_{ST}}$:

$$\mathbf{StudentProfile_{ST} = \{GI_{ST},\ CI_{ST},\ SI_{ST}\}}$$

Where:

1. $\mathbf{GI_{ST}}$ contains the general information about student ST, which includes all data that are not related to courses ST is currently attending.
2. $\mathbf{CI_{ST}}$ contains the course specific information, used to manage the learning process for student ST, more specifically, information included in $\mathbf{CI_{ST}}$ is used by the course presentation module when building personalized learning paths.
3. $\mathbf{SI_{ST}}$ contains ST session information, i.e. data about the current user preferences and the currently course attended.

The general information of student ST, named $\mathbf{GI_{ST}}$, is defined as a 5-uple that includes:

1. $\mathbf{SPD_{ST}}$ representing the personal data of student ST, i.e. name, identification code, registration data, etc. These data are simply used for student identification.
2. $\mathbf{MS_{ST}}$ representing the set of all media suitable for the student ST (for instance, a deaf student cannot actually use any audio media).
3. $\mathbf{STK_{ST}}$ representing the total knowledge of student ST, described through a set of keywords chosen from the ontology of the domain of interest. This is clearly the same ontology used for CUs and CNs requisites and objectives, in order to allow keywords matching between objectives of ST and CU (essential to build personal learning paths). $\mathbf{STK_{ST}}$ is initially either manually provided by the student during the registration into the system or obtained via course admittance tests. When courses are attended, the student gains new knowledge, and $\mathbf{STK_{ST}}$ will also include the list of keywords describing the new topics known by the student. $\mathbf{STK_{ST}}$ is actually used both by course generation and presentation modules.
4. $\mathbf{H_{ST}}$ containing all useful data to retain history of student ST, i.e. access log files, tests results, etc.

$\mathbf{CI_{ST}}$ is an ordered set that contains the set of courses currently attended. The i-th course is represented by $\mathbf{CI_{ST,i}}$, and includes:

1. $\mathbf{Cident_{ST,i}}$ represents the course identifier of i-th course of the student ST.
2. $\mathbf{PathTree_{ST,i}}$ represents the tree containing all possible learning paths leading to the objectives of the i-th course. Each node of $\mathbf{PathTree_{ST,i}}$ is a CU extracted from the graph stored in the DDB. The generation of a $\mathbf{PathTree_{ST,i}}$ is described in detail in Section 4.
3. $\mathbf{LastNode_{ST,i}}$ represents the last lesson learned belonging to the i-th course.
4. $\mathbf{INIT_{ST,i}}$ represents the inital knowledge of the student ST for the i-th course.
5. $\mathbf{GOAL_{ST,i}}$ represents the knowledge student ST wont to achieve by attending the i-th course.

Finally, the third set $\mathbf{SI_{ST}}$ is about ST's session information, and includes:

1. $\mathbf{SAT_{ST}}$ which is the student ST available time for the current session.
2. $\mathbf{DLL_{ST}}$ is the desired learning style for student ST, (e.g. theoretical oriented, practical- oriented, etc).
3. $\mathbf{DLD_{ST}}$ is the level of difficulty expressed by ST, expressed as a numeric value ranging from 1 (basic) to 5 (advanced).
4. $\mathbf{DL_{ST}}$ is the desired level of detail, again provided as a numeric value ranging from 1 (low level of detail, as used when ST wants just a general overview of concepts) to 5 (when ST prefers in-depth study).
5. $\mathbf{DCC_{ST}}$ is the desired course creator (used when ST wants to select a specific teacher, e.g. for his personal teaching style).
6. $\mathbf{M_{ST}}$ is the set of media used in the current session.
7. $\mathbf{DLP_{ST}}$ is the current learning path derived from $\text{PathTree}_{\text{ST,i}}$.
8. $\mathbf{CCC_{ST}}$ represents the current course identifier.

During each session ST proceeds along his personal learning path by choosing one of (possibly) more paths departing from the last learned CU (information about branches are stored into $\mathbf{DLP_{ST}}$). The choice is based on two parameters associated as it will be explained in detail in Section 4.

Student profiles are initially created during the registration by interfacing students with the Profile manager, as indicated in Section 5. Then, $\mathbf{SI_{ST}}$ can be changed by ST to meet his personal needs for current session, in order to take advantage of adaption.

Note instead that $\mathbf{GI_{ST}}$ is provided once during the registration and is independent from courses attendance (except for $\mathbf{STK_{ST}}$ which is however modified only by the system), and finally $\mathbf{CI_{ST}}$ is managed by the system at all and cannot be modified by ST.

5. Courses Generation

As said previously, the model used to describe the DDB is a DAG (indicated below as **G**), whose nodes represent either course nodes or course units. We will refer to both course nodes and course units with the generic term node; the arcs in the graph represent the precedence-succession link between nodes: to study the topics represented by one node, those indicated by the arc must already have been studied. We can therefore say that the node the arc starts from depends on the one it leads to; for example, in Figure 2 the node *A* depends on the node *B* (indicated in the following as $B \prec A$, i.e. *B* precedes *A*) or *C*.

Below we will use lower-case italics, *x*,*y*,*z*,…, to indicate the *topics* that are contained in the DDB and capital letters *A*,*B*,*C*, …, for the graph nodes. We will also use the letter $\boldsymbol{A}$ to indicate a set of topics and the letter $\boldsymbol{N}$ for a set of nodes. Note that the DDB can be described as both the set of topics it contains, i.e. DDB = {*x*,*y*,*z*,…}, and as the set of nodes in the graph, i.e. DDB = {*A*, *B*, *C*,...}. In the former case reference is made to the contents of the DDB, whereas in the latter it is to the kind of link between the various elements.

The link between a node and the topics it deals with is described by means of the objective function $\mathbf{O}(X)$ which, given $X \in \boldsymbol{N}$, returns a set of topics belonging to $\boldsymbol{A}$. The link between a node and the topics that a student needs to have preliminary knowledge of is described by the function $\mathbf{R}(X)$ which, given $X \in \boldsymbol{N}$, returns a set of topics belonging to $\boldsymbol{A}$.

The aim of the courses generator is to find all paths which, given the preliminary knowledge of the STudent (ST) ($\mathbf{STK_{ST}}$), will allow him to gain the knowledge desired, respecting all the precedence-succession requirements (**goal**). We will use P_{ST} to indicate a learning path for a student ST. So the learning path providing the desired knowledge must be such that:

$$\bigcup_{X_i \in P_{ST}} O(X_i) \supseteq goal$$

To describe the fact that this property is satisfied by the union of several nodes alone it is necessary to use an and/or graph, i.e. a graph where it is possible to represent both a situation in which it is possible to reach a node via alternative paths (**or**), and the situation in which a node can only be activated from a set of antecedents (**and**). More specifically:

- The **or** connector intuitively describes the situation in which several arcs lead to a node, representing alternative paths; i.e.:

  $$\forall Y_i, X \in P_{ST} : Y_i \prec X \Rightarrow R(X) \subseteq O(Y_i)$$

  that is, the requisites of node X are the objectives of each nodes that precede it (Y_i).
- The **and** connector describes the situation in which several nodes are connected to the same node and it is necessary to study more than one to meet the requirements. Let:

  $$B_{and}^{X} = \{Y_1, Y_2, ...Y_n\}$$

  be a set of nodes connected by an **and** arc to the node it follows that:

  $$\begin{cases} R(X) \subseteq \bigcup_{Y_i \in B_{and}^{X}} O(Y_i) \\ \forall Y \in B_{and}^{X} \Rightarrow R(X) \not\subset \bigcup_{Y_i \in (B_{and}^{X} - Y)} O(Y_i) \end{cases}$$

On any one node there may be or arcs or different sets of nodes connected via different and arcs.

Figure 2 shows the graph **G** which represents a given DDB; there are arcs of both the **and** and **or** type. As can be seen, the nodes *M*,*N*,*U*,*V* do not depend on any other node. There is in fact no arc starting from it: this means,

for example, that to learn the topics **O(*V*)** it is not necessary to have any previous knowledge. The node *A*, on the other hand, depends on the node *C* or the node *B* (**or** arc); in this case, to study it is necessary for the user to possess the knowledge described by **O(*B*)** or **O(*C*)**.

The nodes *B* and *H* are both required previously to the node *C* (**and** arc); that is, the node *C* can only be studied if both nodes and have been studied, i.e. $R(C) \subseteq O(C) \cup O(H)$; as can be seen, to study *C* a possible alternative is *D*. In short, the node *C* can be studied if the student already posses the knowledge supplied by the node *D* or that supplied by both nodes *B* and *H*.

5.1 Search for a Path

Let *ST* a student, the courses generator searches for learning paths T_{ST} following three stages: the first transforms the graph **G** into the graph G_{ST} eliminating all the nodes whose objectives are known to the student; the second stage transforms G_{ST} into P_{ST}, i.e. a graph containing only the nodes needed to reach **goal** given the student's initial knowledge. The third stage transforms P_{ST} into T_{ST}, which is a tree containing all possible learning paths leading to *ST* goals.

To present learning paths, the courses generator compares them using information extracted from *ST* personal profile, which contains information about the knowledge already acquired and personal preferences.

As said previously, the first step consists of eliminating from the graph **G** all the nodes containing knowledge the student (class) already possesses, keeping the precedence-succession relations unaltered. The courses generator extracts from the personal profile the set $\boldsymbol{A} = \mathbf{STK_{ST}}$ of topics that the student has already learnt and all the nodes $X \in \mathbf{N}$ such that $\mathbf{O}(X) \subseteq \boldsymbol{A}$. All the arcs incident on or originating from *X* are also eliminated. If the graph becomes disconnected after this operation, all the sub-graphs that do not contain nodes whose objectives coincide with those of, i.e. such that $\mathbf{O}(X) \not\subset \mathbf{goal}$ are eliminated. To clarify this procedure, let us assume that a student *tom* intends to interact with the system to learn the topics x_{tom}, y_{tom}, z_{tom}. Using the notation described above, $\mathbf{goal} = \{x_{tom}, y_{tom}, z_{tom}\}$, the profile provides the information that he knows the topics $\{w_{tom}, v_{tom}, t_{tom}\}$, i.e. $\mathbf{STK_{tom}} = \{w_{tom}, v_{tom}, t_{tom}\}$. The DDB is the one represented in Figure 2, where $\mathbf{O}(A) = \{x_{tom}, y_{tom}\}$, $\mathbf{O}(I) = \{z_{tom}\}$, $\mathbf{O}(S) = \{w_{tom}, v_{tom}\}$ and $\mathbf{O}(T) = \{t_{tom}\}$. The resulting graph G_{tom} is described in Figure 3, where the sub-graph *{T, U, V}* has been eliminated, because by eliminating *T* the graph becomes disconnected and $\mathbf{O}(U) \notin \mathbf{goal}$ and $\mathbf{O}(V) \notin \mathbf{goal}$.

The search for a learning path (T_{ST}) consists of visiting the graph G_{ST} starting from the nodes that do not have requisites, i.e. those from which no arcs depart, and then proceeding as far as the nodes $X \in \mathbf{goal}$. The aim of the search is to have a single representation of all the possible paths to be proposed to the student by the presentation module on the basis of his personal preferences. The tree T_{ST} resulting from visiting the graph represented in Figure 3, is shown in Figure 4.

It is obtained by starting from the nodes which are in $\mathbf{STK_{ST}}$ and visiting the graph G_{ST} depth-first. Unlike P_{ST}, in T_{ST} several nodes can refer to the same course unit (e.g., the node *R* appears twice). In addition, nodes connected by the same **and** arc will be grouped, because in that particular path both have to be studied.

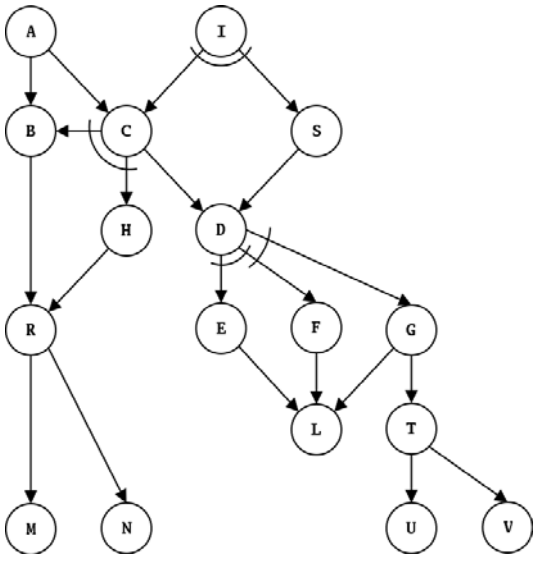

Figure 2

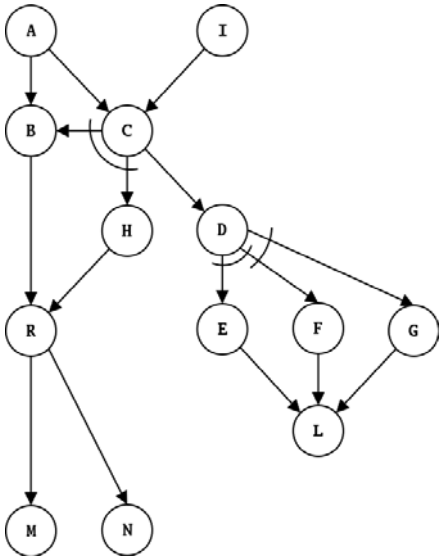

Figure 3

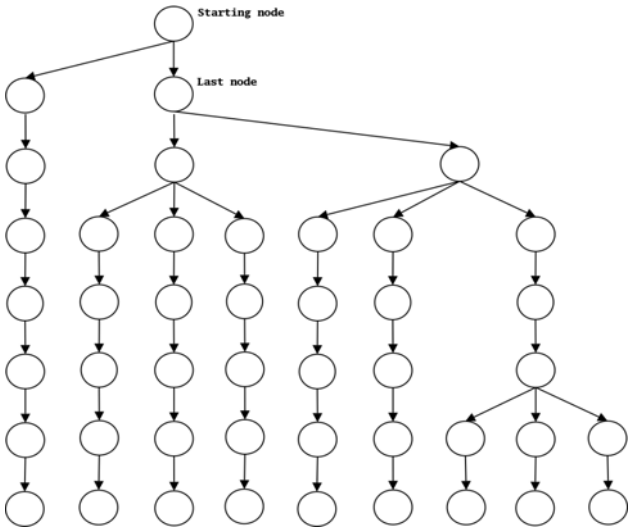

Figure 4

The aim of T_{ST} is to represent all the possible paths in a compact form. This information is then associated with the student and used by the presentation to define the final path.

6. Courses Presentation

Course Presentation module works to build adaptive interfaces to the users (students), i.e. it adds adaptivity to the system. In fact, as said in the previous section, the Course Generator module tailors a course for a given student (ST) producing a $\text{PathTree}_{ST,i,}$ taking into account *CUs* contained in the DDB and the initial knowledge of the student $\mathbf{STK_{ST}}$, but it does not actualize course to the student preferences as defined for the current session.

Course Presentation module aims at providing the student with a virtual-tutor that guides it in the choice of the best route to its goals according with its current characteristics. At each step virtual-tutor will propose to the student the lesson that best fit his requirements and flavours.

Presentation module provides two different interfaces in order to manage either ***Student course request*** or ***Student attempt to lesson***. The activation of these functions is subordinate at the student registration. During the student registration he/she must introduce general information, that, as said in the previous sections, are independent from the course or the lesson the student is currently attending. In the registration phase, the student must introduce $\mathbf{SPD_{ST}}$ (mandatory), $\mathbf{MS_{ST}}$, $\mathbf{PL_{ST}}$, $\mathbf{STK_{ST}}$(optional). If $\mathbf{MS_{ST}}$, and $\mathbf{PL_{ST}}$, are missing they are set to "any". If the student, during the registration into the system, does not introduce values for $\mathbf{STK_{ST}}$ (Student Total Knowledge) it is set to empty. Moreover, in the registration phase the session information $\mathbf{SI_{ST}}$ are initialized with default values. In particular:

1. $\mathbf{CI_{ST}}$ = empty set
2. $\mathbf{SAT_{ST}}$ = ∞
3. $\mathbf{DLL_{ST}}$ = "unexpressed"
4. $\mathbf{DLD_{ST}}$ = "unexpressed"
5. $\mathbf{DL_{ST}}$ = "unexpressed"
6. $\mathbf{DCC_{ST}}$ = "any"
7. $\mathbf{DLP_{ST}}$ = Null
8. $\mathbf{M_{ST}}$ = $\mathbf{MS_{ST}}$
9. $\mathbf{CCC_{ST}}$ = "undefined"

Student course request interface is devoted to manage the interaction between Course Generator and Student. In particular, the Student course request interface allows the student ST to select the course. This interface queries to student ST about his/her objectives, searches among the $\mathbf{C_{ST,i}}$ stored in the student profile for a good choice. If exists no suitable course, the Course Presentation asks to the Course Generator to craft a new course and stores it in the student profile as a new item of $\mathbf{C_{ST}}$.

Student attempt lesson interface works to choose the more adequate path, among that contained in the $\mathbf{PathTree_{ST,i}}$ of the selected the $\mathbf{C_{ST,i}}$. Course Presentation module takes into account several information as, for instance, the device used of the student at this time, the other courses student is attending and the specific request contained in the student profile.

In the following subsections we will discuss about these two items in order to describe how the presentation manages the other modules of the E-learning system.

6.1 Student course request

This interface manages the selection of a course by the student ST. As first step the Course Presentation offers to the student the possibility to choose among the available course stored in $\mathbf{CI_{ST.}}$ If no $\mathbf{CI_{ST,i.}}$ satisfying the student request exists, a new entry of $\mathbf{CI_{ST}}$ is crafted.

When a new course is requested, the student introduces the course objectives and the system works out the set of keywords to be introduced in the **Goal** by means of a function using the ontology related to the current domain. The system searches for the **Goal** into the $\mathbf{CI_{ST}}$ getting one of three possible results:

1. $\exists$ i | $\mathbf{Goal_{ST,i}} \equiv \mathbf{Goal}$ the system notifies the student that a course with the same goal exists in its profile; student can choose the i-th course or build a totally new one.
2. $\exists$ i | $\mathbf{Goal_{ST,i}} \supset \mathbf{Goal}$ the system notifies the student that a course providing more knowledge than requested exists; also in this case student can choose the i-th course or build a totally new one.

3. No course exists in $\mathbf{CI_{ST}}$ that satisfy the student's requests, the system goes on and requires Course Generator module to craft a new course with initial student knowledge equals to $\mathbf{STK_{ST}}$ and course objectives equal to Goal:
   - If the Course Generator return a $\mathbf{T_{ST}}$, that is Course Generator is able to generate the requested course, a new $\mathbf{CI_{ST,i}}$ is inserted in the student profile with:
     $\mathbf{PathTree_{ST,I} = T_{ST}}$
     $\mathbf{LastNode_{ST,I} = root(T_{ST})}$
     $\mathbf{Init_{ST,I} = STK_{ST}}$
     $\mathbf{Goal_{ST,I} = Goal}$
   - If the Course Generator does not return any $\mathbf{T_{ST}}$, student is invited to perform an admittance test that will allow the system to rebuild the initial value of $\mathbf{STK_{ST}}$.

6.2 Student attempt lesson

This interface manages each session *S* and takes into account the current student profile. This phase can be subdivided into two steps: ***Initialising Session*** and ***Next Lesson.*** During the Initialize session a new tree will be build and it will be stored in the section $\mathbf{SI_{ST}}$ of student profile as $\mathbf{DLP_{ST}}$. Tree $\mathbf{DLP_{ST}}$ is a subtree of $\mathbf{PathTree_{ST,i}}$ that contains only the path adequate for the current session. Each node *n* of $\mathbf{DLP_{ST}}$ contains the information of the *CU* concerning with objective and requirement and two values, $\boldsymbol{\varepsilon_n}$ and $\boldsymbol{\delta_n}$. $\boldsymbol{\varepsilon_n}$ gives information about the effectiveness of the node and $\boldsymbol{\delta_n}$ gives information about the increase of $\mathbf{STK_{ST}}$ due to attempt *CU* in the node *n*. At each subtree $\mathbf{T_n}$ of $\mathbf{DLP_{ST}}$ rooted at *n* we link two value $\mathbf{E_n}$ and $\boldsymbol{\Delta_n}$. $\mathbf{E_n}$ represents the average effectiveness of the paths belonging to $\mathbf{T_n}$ and $\boldsymbol{\Delta_n}$ represents the max number of objective, included in the $\mathbf{Goal_{ST,i}}$, that can be reached following $\mathbf{T_n}$. The end of Initialize session enables the student to attempt lessons. ***Next Lesson*** session is devoted at selecting the next lessons using values $\mathbf{E_n}$ and $\boldsymbol{\Delta_n}$ and the result of test performed during current *CU*.

6.2.1 Initializing session

In this phase the student can modify the session value $\mathbf{SI_{ST}}$. Then the system using $\mathbf{MS_{ST}}$ and the characteristic of the device currently in use, updates the value of $\mathbf{M_{ST}}$ (i.e. the set of media suitable for the student ST, but that are also adequate for the device selected in the current session *S*). Let $\boldsymbol{\gamma(S)}$ a function returning the set of media adequate for the device chosen during session *S*, $\mathbf{M_{ST}}$ obtained by:

$$\mathbf{M_{ST} = MS_{ST} \cap \gamma(S)}$$

Then the student can set its profile information concerning with the current session, in particular the student ST can modify its available time ($\mathbf{SAT_{ST}}$), desired style ($\mathbf{DLL_{ST}}$), level of difficulty ($\mathbf{DLT_{ST}}$), desired level of detail ($\mathbf{DL_{ST}}$) and desired course creator $\mathbf{DCC_{ST}}$. Then the student can select the i-th course among that in $\mathbf{CI_{ST}}$. The following initialisation are made:

$$\mathbf{CCC_{ST} = Cident_{ST,i}}$$

At this time the Initialising session proceeds at crafting the $\mathbf{DLP_{ST}}$ following a set of operations aiming at adapting the tree $\mathbf{PathTree_{ST,i}}$ at the current situation as introduced above. We take as $\mathbf{DLP_{ST}}$ the subtree of $\mathbf{PathTree_{ST,i}}$ having root $\mathbf{LastNode_{ST,i}}$ in figure 4 it is shown the $\mathbf{PathTree_{ST,i}}$. In order to evaluate the path contained in DLP_{ST} let us consider:

- for each node n a couple of parameter, named $\boldsymbol{\varepsilon_n}$ and $\boldsymbol{\delta_n}$ that give information about the node.
- for each subtree $\mathbf{T_n}$ of $\mathbf{DLP_{ST}}$ rooted at n a couple of parameter, named $\mathbf{E_n}$ and $\boldsymbol{\Delta_n}$, that characterizes $\mathbf{T_n}$.

To evaluate $\boldsymbol{\varepsilon_n}$ we use the function named *effectiveness(n,* $\mathbf{SI_{ST}}$*)*.

$$\boldsymbol{\varepsilon_n} = \textit{effectiveness}(n, \mathbf{SI_{ST}}) \qquad (1)$$

This function return 0 if the *CU* of node *n* does not satisfy the requirement of $\mathbf{SI_{ST}}$, that is the teaching material cannot be currently used by student ST. In other cases it returns a value ranging from 1 to *MaxValue*, where *MaxValue* represents the best student choice. If *effectiveness(n,* $\mathbf{SI_{ST}}$*)*.= 0, $\mathbf{DLP_{ST}}$ is pruned away of the subtree $\mathbf{T_n}$ because since the node *n* cannot be followed, no *CUs* in $\mathbf{T_n}$ can be reached. To evaluate $\boldsymbol{\delta_n}$, representing the number of goal added to $\mathbf{STK_{ST}}$ after having attempted the *CU* in the node *n*, we have:

$$\boldsymbol{\delta_n} = \textit{cardinality}((\mathbf{O}(n) \cap \mathbf{Goal_{St,i}}) - \mathbf{STK_{ST}}) \qquad (2)$$

So $(\mathbf{O}(n) \cap \mathbf{Goal_{St,i}})$ is the set of objectives that gives the *CU*. $\boldsymbol{\delta_n}$ could be equal zero in two cases i) if any objectives in $\mathbf{O}(n)$ not be a member of $\mathbf{Goal_{St,i}}$ or ii) if each objectives in $\mathbf{O}(n)$ are already inserted in $\mathbf{STK_{ST}}$.
Let us define for a tree $\mathbf{T_n}$, $\mathbf{E_n}$ and $\boldsymbol{\Delta_n}$ as follow:

$$\mathbf{E_n} = \textit{average}(\boldsymbol{\varepsilon_n} + \textit{max}_{k=1..f}(\mathbf{E_{n,k}})) \qquad (3)$$

where $\textit{max}_{k=1..f}(\mathbf{E_{n,k}})$ is the max value ranging over values belonging to sons of node *n*. Thus $\mathbf{E_n}$ gives feeling of the average effectiveness of the best path starting from *n*.

$$\boldsymbol{\Delta_n} = \boldsymbol{\delta_n} + \textit{max}_{k=1.f}(\boldsymbol{\Delta_{n,k}}) \qquad (4)$$

where $\textit{max}_{k=1..f}(\boldsymbol{\Delta_{n,k}})$ is the maximum among values of $\boldsymbol{\Delta_{n,k}}$ related to sons of the node *n*. Thus $\boldsymbol{\Delta_n}$ gives feeling of the max number of objective can be reached following the best path starting from *n*.

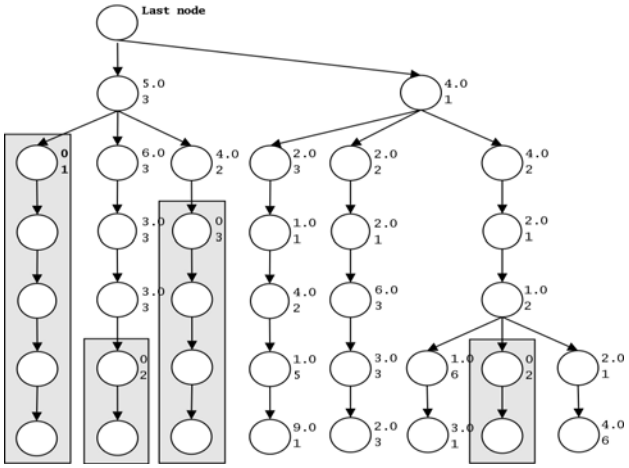

Figure 5

For example, the $\mathbf{PathTree}_{ST,i}$ shown in figure 4 produces the $\mathbf{DLP}_{ST}$ in figure 5, where are shown the values ε_n and δ_n evaluated using (1) and (2). The subtrees, marked using gray box in figure 5, must be pruned away because their root is labelled with $\varepsilon_n = 0$. In figure 6 it is shown the same $\mathbf{DLP}_{ST}$ of figure 5 after pruning the subtrees, in which each node n is labelled with values $\mathbf{E}_n$ and Δ_n, evaluated using (3) and (4).

6.2.1 Next Lesson

In this phase, the $\mathbf{DLP}_{ST}$ produced in the previous Initialising Session is used. The selection of the next lesson depends on values of Δ_n and $\mathbf{E}_n$. Since Δ_n measures the chance to get some objectives, the system proposes the node that maximize this chance. Whenever more than one node has the same value of Δ_n system select the tree having maximum value of $\mathbf{E}_n$.

Let us consider *X* the selected lesson. Before the student ST attempt this lesson, the system test if the $\mathbf{O}(X)$ are already in the $\mathbf{STK}_{ST}$, that is:

$$\mathbf{O}(X) \subseteq \mathbf{STK}_{ST} \qquad (5)$$

If condition expressed in (5) is false, the lesson *X* can be attempted by ST. At the end of the lesson *X* the function *testResult(X)* returns the set of objectives not passed. The student knowledge $\mathbf{STK}_{ST}$ is updated as follows**:**

$$\mathbf{STK}_{ST} = (\mathbf{STK}_{ST} \cup \mathbf{O}(X)) - TestResult(X) \qquad (6)$$

If condition expressed in (5) is true, all the objective of *X* are already possessed by the student and the algorithm starts to select the next lesson, because the current lesson *X* was already passed by ST.

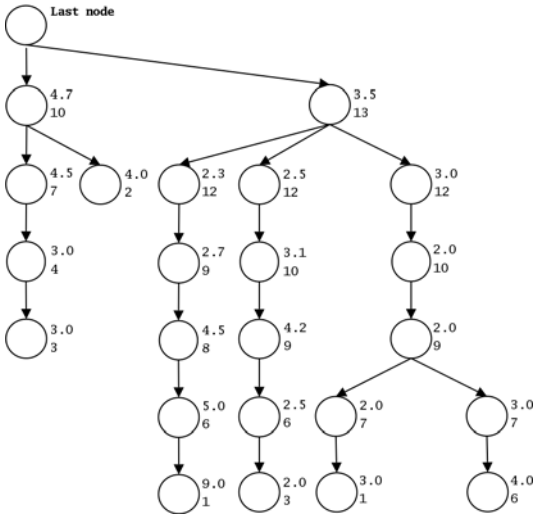

Figure 6

Before lesson selection, the algorithm verify if all the requisites for lesson *X* are already possessed by the student ST, that is:

$$R(X) \subseteq STK_{ST} \qquad (7)$$

If condition (7) is false the algorithm restart the lesson selection from the father of *X*. The Next lesson phase ends when:

1. The last selected node is a leaf. This can happen when either the $\mathbf{Goal}_{ST,i}$ is reached or the current session the goal can not be reached due the session preferences. In this last case $\mathbf{LastNode}_{ST,i} = X$.
2. The algorithm try to select as *X* the father of the root of $\mathbf{DLP}_{ST}$. In this case the Course Presentation asks to Course Generator to craft a new course.
3. The student ST logoff. In this case $\mathbf{LastNode}_{ST,i} = X$.

For example, if we consider the tree shown in figure 6 and under the hypothesis that the student pass all tests required by *CUs,* the path followed by the student is shown in figure 7.

7. Evaluation

In order to evaluate our proposal, a prototype was developed in the context of the E-learning project "P38 - CLUSTER 22 framework", sponsored by the "Ministero dell'Istruzione, dell'Universita' e della Ricerca" (MIUR). A preliminary evaluation was performed by populating the DDB with CUs in the following domains:

- computer science history
- computer architectures
- programming languages
- web-oriented programming languages
- compilers and interpreters

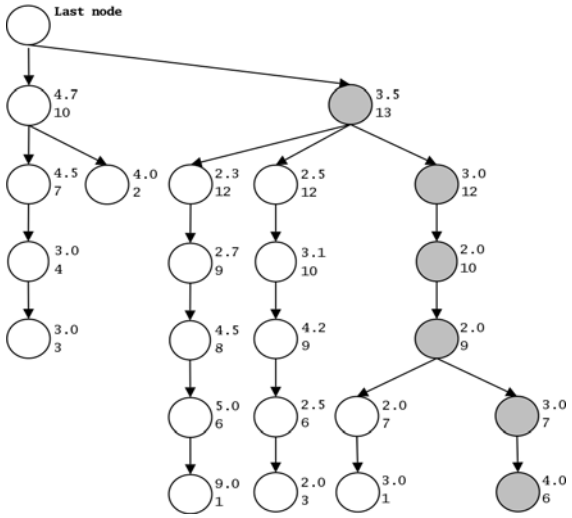

Figure 7

Several teachers made contributions to both the teaching materials and the domain databases. At the end of this startup phase the DDB contained about 60 CUs. The next phase was the generation of two Course Nodes:

- data structure. This includes topics belonging to computer science history, computer architectures, and programming languages.
- languages and compilers. This syllabus contains topics belonging to programming languages, web-oriented programming languages and compiler and interpreter domains.

The course nodes were generated reusing *CUs*. Moreover, the two courses partially overlap as they have some topics in common, mainly in the programming language domains, for example the introduction to programming paradigms (imperative, declarative, object-oriented, ...). In general, course overlap is mainly related to the generic nature of basic courses containing topics suited for students on any degree courses. This partial overlap may force students to study topics they already know, with a negative impact on the overall time needed to learn course topics.

These problems are overcome by using the proposed approach, which produces personalized learning paths avoiding overlapping, based on personal student profiles. Using personalized learning paths, the time students take to learn the topics included in "languages and compiler" courses has been decreased by approximately 15 per cent. Moreover, student satisfaction has grown (as detected from satisfaction forms compiled by students) since the system automatically tailors course topics to personal capabilities and needs.

The number of students accessing the system, initially restricted to a small set of test students, is currently growing, leading to a more accurate evaluation of our model. We plan a second evaluation phase to be performed when the number of course nodes and students involved covers a whole degree course.

8. Conclusions

In this paper we introduced an architecture for an e-learning system having two main features. The first feature is the possibility of sharing both courses topics and teaching materials, in order to provide students with an uniform set of topics to be learned and exploit, as far as possible, existing material developed by teachers/courses creators. The second feature is to promote active learning, allowing the construction of courses which are personalized in terms of both topics and teaching materials; this is based on each student's profile, thus providing students with an adaptive environment which dynamically adjusts the personalized course during the learning process. A prototype has been implemented providing both teachers and students with a simple web-based learning environment. A first evaluation give us some interesting results; we are currently performing a deep evaluation of the proposed approach over larger set of students and courses.

Some considerations define the boundaries of the model proposed in this paper, highlighting both limitations and directions for future research.

The graph model we chose to represent precedence-succession relationships between concept is simple and flexible; however, we are also investigating representations covering other kinds of links between Cus (e.g. "part of" relationships).

Besides, the database implemented in the prototype currently contains some CU and CNs belonging to the same domain of interest, i.e. computer science topics, though no conceptual limitation seems to exist either in the extension (i.e. number of CUs/CNs) or in the intension (i.e. including different domains of interest simultaneously), even if the system was designed as intended (i.e. a database with many CUs and many CNs, also to promote reuse), rather than intensive, which could lead to too many loosely coupled "knowledge islands".

It is also worth mentioning that the system becomes effective as soon as a significant number of CUs/CNs have been provided, otherwise its personalization capabilities are limited. In this sense, we have planned to develop tools (e.g. using a wizard) to allow teachers to insert their concepts and materials rapidly. However, since the aim is that of promoting reuse, no great amounts of data should be inserted into the system after the startup phase. Moreover, this should be enforced through the development of tools aiming at guiding teachers to choose from existing CUs/CNs instead of directly creating a new one if this is not strictly needed; all this aims at preventing useless repetition of similar CUs/CNs.

Concepts used to express requisites and objectives used throughout the paper should be arranged according to a proper ontology, both when characterizing both CUs/CNs and student profiles, as well as when building a new course on a given topic. A good ontology is essential, to include as many topics as possible, together with synonyms/hypernonyms, in order to allow a fine-grained description for CUs, CNs, precedence/succession relationships, student profiles, and student requests for new courses. We are currently investigating ontology issues, in order to improve system accuracy and effectiveness.

Finally, an in depth investigation must be accomplished on which API should be developed to allow learning services, and how the Interface Layer outlined in section 3.4 actually works over other layers.

References

[1] S. Bergstedt, S. Wiegreffe, J. Wittmann, D. Moller - Content management systems and elearning systems -A symbiosis? in proc. of the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT), Athens, Greece, 2003.

[2] S. Yang, A. Chang, I. Chen – Enhancing Learning Resources Reusability with a New Learning Design Framework, in proc. Of IEEE International Conference on Advanced Learning Technologies (ICALT), IEEE, 2005

[3] Mariappan, V., & Aslandogan, Y. (2004). - Vizipen: A System for Automatic Generation of Multimedia Concept Presentations. World Conference on E-Learning in Corp., Govt., Health., & Higher Ed. 2004(1), 2812-2818.

[4] M. A. Mora, R. Moriyon. - Collaborative analysis and tutoring: The FACT framework, in: Proc. of IEEE International Conference on Advanced Learning Technologies (ICALT), IEEE, Madison, Wisconsin, USA, 2001, pp. 82–85.

[5] A. Tretiakov, Kinshuk, T. Tretiakov - Designing Multimedia Support for Situated Learning in proc. of the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT), Athens, Greece, 2003.

[6] A. Collins, J.S. Brown, S.E. Newman - Cognitive Apprenticeship: Teaching the crafts of reading, writing, and mathematics. Knowing, learning, and instruction: Essays in honor of Robert Glaser, Ed. L. B. Resnick, Hillsdale, New Jersey, USA, 1990.

[7] K. Wijekumar - Implementing collaborative learning research in web-based course design and management systems, in: Proc. of IEEE International Conference on Advanced Learning Technologies (ICALT), IEEE, Madison, Wisconsin, USA, 2001.

[8] Akiko Inaba, Riichiro Mizoguchi - Learners' Roles and Predictable Educational Benefits in Collaborative Learning An Ontological Approach to Support Design and Analysis of CSCL, Lecture Notes in Computer Science, Volume 3220, Jan 2004, Pages 285 – 294

[9] P. Brusilovsky, A. Kobsa, J. Vassileva - Adaptive Hipertext and Hypermedia, Dordrecht, Kluwer.

[10] A. Bork, G. Ju – Elearning versus Alearning

[11] O. Sessink, R. Beeftink, J. Tramper, R. Hartog - Author-Defined Storage in the Next Generation Learning Management Systems in proc. of the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT), Athens, Greece, 2003.

[12] G. Weber, P. Brusilovsky - ELM-ART: An Adaptive Versatile System for Web-Based Instruction, in Journal of Artificial Intelligence in Education, 12, pp351-384, 2001.

[13] D. Rasseneur, P. Jacoboni, P. Tchounikine - An Approach to Distance Learning Curriculum Appropriation in proc. of the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT), Athens, Greece, 2003.

[14] HSia, Y.. 2003. Curricular Automata in their applications. In Proc. of IEEE International Conference on Advanced Learning Technologies (ICALT), Athene, Greece, pp. 264–265.

[15] J. Vassileva, R. Deters – Dynamic courseware generation on the WWW, British Journal of Education Technologies, 1998, 29 (1), 5-14

[16] P. Brusilovsky, G. Weber – ELM-ART: and Adaptive Versatile System for Web-based Instruction, Intl. Journal of Artificial Intelligence in Education, 2001, 12, 351-384

[17] P. Brusilovsky, J. Vassileva – Course Sequencing Tecniques for large scale web-based education, Int. Journal of Cont. Engineering Education and Lifelong Learning, Inderscience, 2003, (13), 75-94

[18] M. Baldoni, C. Baroglio, V. Patti – Web-based Adaptive Tutoring: An Approach Based on Logic Agents and Reasoning about Actions, Artificial Intelligence Review, Kluwer, 2004, (22), 3-39

[19] P. Dolog, N. Henze, W. Nejdl, M. Sintek – Personalization in Distributed e-learning Environments, in proc. of ACM WWW 2004, New York, USA, 2004

[20] T. Zarraonandia, C. Fernandez, P. Diaz, J. Torres – On the way of an ideal learning system adaptive to the learner and her context, Proc. of the 5th IEEE International Conference on Advanced Learning Technologies (ICALT), 2005

[21] S. Murray, J. Ryan, C. Pahl - A Tool-Mediated Cognitive Apprenticeship Approach for a Computer Engineering Course, in proc. of the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT), Athens, Greece, 2003.

[22] M. Najjar, A. Mayers - A Computational Cognitive-Based Approach to Represent Knowledge within Intelligent Tutoring Systems, in proc. of the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT), Athens, Greece, 2003.

[23] V. Tsiriga, M. Virvou - Initializing Student Models in Web-Based ITSs: a Generic Approach, in proc. of the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT), Athens, Greece, 2003.

[24] Aimeur, E., et alii, 2002. CLARISSE: A machine learning tool to initialize student models. In Proc. of 6-th International Conference on Intelligent Tutoring Systems. LNCS, Vol .2363, pp. 718-728.

[25] E.B. Page - New Computer Grading of Student Prose: Using Modern Concepts and Software, Journal of Experimental Education, vol 62, no. 2, pp. 127-142, 1994

[26] T.K. Landauer, P.W. Folts, D. Laham - Introduction to Latent Semantic Analysis Discourse Processes, vol.25, pp. 259-284, 1998
[27] D. Callear, J. Jerrams-Smith, V. Soh. - Bridging gaps in computerised assessment of texts, in: Proc. of IEEE International Conference on Advanced Learning Technologies (ICALT), IEEE, Madison, Wisconsin, USA, 2001, pp. 139–140.
[28] M. Alvarez. - Exploring minds: Revealing ideas electronically, in: Proc. of World Multiconference On Systemics, Cybernetics And Informatics (WMSCI), Orlando, Florida (USA), 2001.
[29] Abbas, J., Norris, C., and Soloway, E. - Analyzing middle school students' use of the ARTEMIS digital library. In Proc. of IEEE International Conference on Advanced Learning Technologies (ICALT), 2001, Madison, Wisconsin, USA, pp. 107–109.
[30] Fung, I. P.-W. 2000 - A hybrid approach to represent and deliver curriculum contents. Proceedings of International Workshop on Advanced Learning Technologies (IWALT2000) (Palmerston North, New Zealand). Kinshuk, Chris Jesshope, Toshio Okamoto (Eds), pp. 38–42.
[31] Kort, B., Reilly, B., and Picard W., R. - An alective model of interplay between emotions and learning: Reengineering educational pedagogy - building a learning companion. In Proc. Of IEEE International Conference on Advanced Learning Technologies (ICALT), 2001, IEEE, Madison, Wisconsin, USA, 43–48.
[32] Giraffa M., L. and da Costa Mora, M. - Towards student models (really) based on mental states. In Proc. of World Multiconference On Systemics, Cybernetics and Informatics (WMSCI). 2001, Orlando, Florida (USA), 43–48.
[33] Ravenscroft, A. and P. Matheson P., M. 2001. - Carpe diem: Models and methodologies for designing engaging and interactive e-learning discourse. In Proc. of IEEE International Conference on Advanced Learning Technologies (ICALT). 2001, IEEE, Madison, Wisconsin, USA, 74–77.

Vincenza Carchiolo is currently full professor of Computer Science in Department of Informatics and Telecommunications at University of Catania. Her research interests include information retrievial, query languages, distributed systems, e-learning systems, and formal language.
She received a degree with Honors in Electrical Engineering from University of Catania, Italy in 1983.
She is member of ACM**.**

Alessandro Longheu received the M.S. degree in Computer Engineering in 1997 from University of Catania, and then the Ph.D. degree in 2001 from University of Palermo. He started to work on information integration in database area, then also considering such questions in WWW context. Currently he is a contract professor in Programming Languages at the Faculty of Engineering of Catania. His research interests includes e-learning, workflows, information retrieval, and information integration and exchange in a web-based environment.

Michele Malgeri is associate professor in Department of Informatics and Telecommunications at University of Catania. His research interests include distributed systems, e-learning systems information retrieval, query languages and formal language. He received a degree with Honors in Electrical Engineering from University of Catania, Italy in 1983.

Giuseppe Mangioni received the degree in Information Engineering (1995) and the Ph.D degree (2000) at the University of Catania (Faculty of Engineering), where he became a professional engineer in 1995. In 1996 he joined the Dept. of Information and Telecommunications Engineering as a contract researcher. Currently he is a contract professor in Computer Networks at the Faculty of Engineering of Catania. Presently, his main research interests concern e-learning, P2P overlay networks
and complex networks.