Roma Tre University
Ph.D. in Computer Science and Engineering

Adaptive Techniques in Web-Based Education

Giulia Vaste

Adaptive Techniques in Web-Based Education

A thesis presented by
Giulia Vaste
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Engineering

Roma Tre University
Dept. of Informatics and Automation

April 2010

To Carlo, to my family and to my best friends.

Abstract

This dissertation proposes the use of artificial intelligence methodologies and techniques for providing personalization of web-based courses. Almost all web-based systems try to carry out personalization in order to be more useful, more attractive, and more efficient in fulfilling user's needs. However, personalization has a cost in terms of background operations, for instance in educational systems in terms of teacher's effort. What is a reasonable compromise between sofisticated personalization methodologies and their realization? Is it possible to provide personalization with an acceptable effort for the domain expert responsible for contents producing?

This dissertation focuses on these questions, considering in particular systems for web-based education, and proposes a methodology that aims to carry out a reasonable compromise between an effective personalization from the student's point of view and from the teacher's point of view, or, in general, from the user's point of view and from the domain expert's point of view.

From the student's point of view, personalization is provided on the basis of student's knowledge and learning styles and guiding the student during the fruition of the course, like a teacher could do: proposing a sequence of contents suitable for the student at the beginning of the course and performing recovery strategies, during the fruition of the course, if the study does not proceed as it should.

From the teacher's point of view personalized courses are generated automatically, on the basis of the student model. The teacher is required to specify few metadata, necessary for characterizing learning materials, such as prerequisite relations and suitability of contents for a given type of student. The teacher is helped by a graphical interface, allowing a global vision of the course, and he can express didactic preferencies, such as the level of the course, according to the Bloom's Taxonomy. The effort required to the teacher is as near as possible to his "way of thinking": prerequisite relations are generally defined, even if implicitly, when a course is arranged; learning materials are tagged, according to the Felder and Silverman's learning styles model. Adherence to this model is not, however, a strict constraint for the teacher: the estimate of student's learning styles is in fact updated taking into account the teacher's tagging of the learning materials. In this way, relevance is given to the matching between the teacher's tagging of the learning materials and the student's way of learning: if the student studies a given material with success, the material is considered suitable for the student and his learning styles are updated towards the weights given by the teacher to that material.

On the basis of the above-mentioned methodologies the LS-Plan system has been proposed. LS-Plan provides educational hypermedia with adaptivity; it has been integrated in the Lecomps educational hypermedia in order to carry out evaluations both from the student's and from the teacher's point of view. A *layered* and an *as a whole* evaluation, together with evaluations of teacher's functionalities, have been performed and have shown positive results.

Different approaches have been proposed in the literature for *curriculum sequencing*, that is *"help the student to find an "optimal path" through the learning material"*. According to the aim of providing support for teachers in performing personalization, a suitable system, LS-Lab, for comparing different algorithms has been proposed. LS-Lab provides a uniform environment in which several algorithms can be compared using the same input, i.e. the same set of didactic materials, the same sample student models and the same learning objective. The subjective comparison, made by teachers or domain experts, is supported by some metrics and by the visualization of the produced sequences.

According to the necessity of providing an easy-to-use personalization for background actors, the personalization methodologies proposed for the educational domain have been applied also for cultural visits personalization. Analogies and differences between course personalization and cultural visits personalization have been detected and the framework for course personalization has been adapted and enhanced taking into account visitor's interests.

Acknowledgments

First, I would like to thank my advisor Carla Limongelli, who supervised and guided my work. During these years she has been a reference point both from a professional and from a personal point of view.

I would like to thank the other coauthors of my works, Filippo Sciarrone and Marco Temperini. I learned a lot from them.

I am thankful to the members of the Dept. of Informatics and Automation at Roma Tre University, and in particular to the members of the Artificial Intelligence Group, who welcomed me. A special thanks to Andrea Orlandini, a colleague, but also a friend.

Finally, I am thankful to all those I love: Carlo, my family and all my friends.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"The challenge in an information-rich world is not only to make information available to people at any time, at any place, and in any form, but specifically to say the "right" thing at the "right" time in the "right" way." [Fis01]

This Fischer's sentence expresses the widespread need for personalization in information systems: it is necessary to "know" the user and to "adapt" the systems to the user's needs in order to say the "right" thing at the "right" time in the "right" way, in other words in order to provide "personalization". Almost all web-based systems try to carry out personalization: search engines, e-commerce applications, educational systems, virtual museums, are some examples. Information about the user is collected explicitly, by suitable questions submitted to the user, or implicitly, by observing the user's behavior. The system produces a model of the user and uses this information for providing a "personalized service", that is for sorting search results on the basis of the possible relevance for the user, for suggesting books potentially interesting for the user, for providing suitable learning materials in a suitable order for a given student, for explaining a work of art according to the visitor's interests, etc. Research in user modeling and adaptation is a wide field and several methodologies and solutions have been proposed in the literature for personalization of web-based applications. Personalization makes information systems more useful, more attractive and more efficient in fulfilling user's needs. However, it has a cost in terms of background operations, in particular in systems, such as educational ones, in which several "actors" are involved: the student, but also the teacher, the tutor, the course designer, etc. Personalization of contents can be effective for students, but how much does it cost in terms of teacher's or domain expert's effort? What is a reasonable compromise between sofisticated personalization methodologies and their realization? Is it possible to provide personalization with an acceptable effort for the domain expert responsible for contents producing?

This dissertation focuses on these questions and proposes a methodology that aims to carry out a reasonable compromise between an effective personalization for students and teachers, or, in general, for users and domain experts. The work has in fact found its origins in educational systems, but it can be generalized to other contexts: the proposed methodology can be applied in contexts different from the educational one, for instance for personalization of cultural visits, in which the involved "actors" are the visitor and the domain expert. A cultural tour, personalized on the basis of the visitor's needs, can be attractive and useful for the user, but it is necessary that this personalization is "feasible" from the expert's point of view.

Personalization in education means that students with different starting knowledge, background, interests, objectives and learning styles, can have the possibility to study in their preferred way, through personalized contents, learning paths, modalities of delivery of the learning material, teaching methodology, etc. From the student's point of view a personalized educational system can ideally be a personalized tutor that follows the student during the learning experience and provides him with prompts, help, explanations, in the "right" moment and in the "right" way. But how much does it cost from the teacher's point of view? Producing contents is an hard task that can not be performed automatically and can be only lightened through reusability and interoperability. Providing personalization means to adapt the course to different students, that is producing different materials for explaining the same concepts, or sequencing them in different ways, providing different didactic strategies, and so on. An *ideal* distance learning environment will provide: "student's satisfaction", through functionalities such as contents fruition, assessment and self-assessment, collaboration tools, etc., offered in a personalized way; "teacher's satisfaction", that is providing support for contents creation and for their sequencing, for assessment management and for expressing didactic strategies, etc., in an easy-to-use environment; reusability and interoperability of contents, tests, courses, etc. by applying suitable standards; administration functionalities, such as log-in management, logging, etc.

Currently two are the most important families of systems for Web-based education: Learning Management Systems on the one hand and Intelligent Tutoring Systems and Adaptive Educational Hypermedia on the other hand.

Learning Management Systems, such as Moodle, Docebo, Ilias, etc., usually guarantee contents riusability and interoperability, by adhering to standard "de facto". They offer numerous functionalities, such as courses registration, logging, agenda, and so on. From the student's point of view they provide an environment for course taking, for assessment and self-assessment, and collaboration tools, such as chat and forum. From the teacher's point of view they offer support for contents and test management. However Learning Management Systems allow a very limited, often null, personalization to student's needs, progress, previous knowledge, and personal characteristics.

On the contrary, Intelligent Tutoring Systems and Adaptive Educational Hypermedia have been developed with the aim of providing personalized educational systems. Intelligent Tutoring Systems have been developed by 1960s-1970s, producing tutors for specific domains, capable of supporting students mainly in problem solving, as a human tutor should do. These systems are very sofisticated in student modeling techniques: they mainly represent student's knowledge and misconceptions using different methodologies both for building such models and for updating them. Moreover they implement suitable didactic strategies, used on the basis of the current student model. In this way, these systems constantly monitor the student, detecting difficulties and progress and performing the suitable teaching action. However the effort for producing such systems is considerable, it is necessary to code the desired tutoring methodologies and the expert's knowledge. Moreover they focus in specific domain for obtaining such sofisticated personalization.

Adaptive Educational Hypermedia *"were born in a trial to combine an intelligent tutoring system and an educational hypermedia"* [Bru00]. These systems provide personalization of contents and navigation support, taking advantages by experiences done in research in intelligent tutoring systems, especially for student modeling techniques. Several systems have been developed focusing on *curriculum sequencing*, that is *"help the student to find an "optimal path" through the learning material"* [Bru00]. The "optimal path" is often obtained on the basis of student's knowledge and learning styles. However these systems are generally prototypes, they are usually not very spread, and a limited attention has been dedicated to the "teacher's satisfaction" aspect.

Summarizing, the state of the art in educational systems presents on the one hand Learning Management Systems, very spread, but with limited support for personalization, and on the other hand Intelligent Tutoring Systems and Adaptive Educational Hypermedia, often available only at a prototype level and not very spread, but providing sofisticated support for personalization. Both the typologies of systems require a significant effort by the teacher or course designer: beyond contents production, both expressing didactic strategies and providing courses with personalization are difficult tasks.

This dissertation presents methodologies combining "student's satisfaction" and "teacher's satisfaction" aspects. The aim is to carry out a personalization, that can be considered effective both from the student's point of view, i.e. personalized contents and personalized sequencing of learning materials, and from the teacher's point of view, i.e. ease of use for teachers.

In particular, from the student's point of view, personalization is based on student's knowledge and learning styles. It is carried out proposing a sequence of contents, suitable for the student, at the beginning of the course and guiding the student during the fruition of the course. In particular the idea is to implement recovery strategies that mimic a usual teacher's behavior in presence of learner's difficulties in the study of a given topic. Sequencing of contents is modeled as a planning problem and is realized by the Pdk planner [MLOP07]. Recovery activities are performed by an algorithm that takes in input self-assessment results, the time spent for studying a content, the characteristics of the content proposed to the student and the current student model, and mimics the behavior of a teacher that tries to re-explain the concept not acquired, initially with the same material, then with different materials, if available, and that verify the prerequisites if the concept is not acquired yet.

From the teacher's point of view personalized courses are built automatically, on the basis of the student model. The teacher is required to specify few metadata, necessary for characterizing learning materials, such as prerequisite relations and suitability of contents for a given type of student. The teacher is helped by a graphical interface, allowing a global vision of the course, and he can express didactic preferencies, such as the level of the course, according to the Bloom's Taxonomy [Blo64]. The effort required to the teacher is as near as possible to his "way of thinking": prerequisite relations are generally defined, even if implicitly, when a course is arranged; learning materials are tagged considering students learning preferences, according to the Felder and Silverman's learning styles model [FS88]. Adherence to this model is not, however, a strict constraint for the teacher, in fact the estimate of student's learning styles is updated on the basis of the teacher's tagging of the learning materials. If the student studies a given material with success, the material is considered suitable for the student and his learning styles are updated towards the weights given by the teacher to that material. In this way, relevance is given to the matching between the teacher's tagging of the learning materials and the student's way of learning.

On the basis of the above-mentioned methodologies the LS-Plan system has been developed. LS-Plan main components are: the Teacher Assistant, responsible for teacher's functionalities, the Pdk Planner, used for providing *curriculum sequencing*, and the Adaptation Engine, responsible for the management of the student model and for the adaptation decision making, i.e. responsible for student's functionalities. LS-Plan is not an adaptive system: it provides educational hypermedia with adaptivity. Future works aim to fully integrate LS-Plan in a widespread Learning Management System, such as Moodle, in order to exploit the above-mentioned functionalities of a Learning Management System, such as courses registration, logging, agenda, collaboration tools, etc. Reusability and interoperability of contents are guaranteed, in fact the didactic material used by LS-Plan can be described through a widespread standard for learning objects description, i.e. IEEE LOM[1].

In order to evaluate the system, it has been integrated in the Lecomps educational hypermedia [TV07] and the performed evaluations have shown positive results, both from the student's and from the teacher's point of view. In particular, according to the literature about methods for the empirical evaluation of adaptive systems [BKS04, Gen05], a *layered evaluation* has been performed for assessing the suitability both of the user modeling and of the adaptation components of the system. The adaptation decision making evaluation aimed to answer to questions like *"are the adaptation decisions valid and meaningful, for the given state of the user model?"* [BKS04]. In particular, sequences produced at the beginning of the course for two given student model, and adaptation performed during the course taking have been evaluated by a sample of teachers. The adaptation decisions have been also evaluated observing the students' behavior, i.e. measuring the agreement of students about the system suggestions. The user model evaluation aimed to answer to questions like *"are the user's characteristics being successfully detected by the system and stored in the user model?"* [BKS04]. Students were asked to express their agreement about their own model. An *as a whole* evaluation has been also performed, comparing two students groups, using and not-using personalization provided by LS-Plan. This evaluation aimed to measure the added value provided by the system to students' learning experience. Also evaluations of teacher's functionalities have been conducted in order to collect teachers' opinions about general usefulness of the system and about learning material sequences produced by the system, in terms of similarity with the teachers' ones and in terms of their own quality.

In LS-Plan, personalization is based on a suitable *curriculum sequencing*, produced through the Pdk planner at the beginning of the course and through

[1]http://ltsc.ieee.org/wg12/

the adaptation algorithm during the student's course taking. Different approaches have been proposed in the literature for *curriculum sequencing*, and it is difficult to compare them. According to the aim of providing support for teachers in performing personalization, in order to compare different algorithms in a uniform environment, a suitable system has been developed, LS-Lab. For uniform environment we intend that several algorithms can be compared using the same input, i.e. the same set of didactic materials, the same sample student models and the same learning objective. The system provides also some early metrics for an objective comparison of the sequences produced by the available algorithms. The subjective comparison, made by teachers or domain experts, is supported by these metrics and by the visualization of the produced sequences.

According to the necessity of providing an easy-to-use personalization for background actors, the above-mentioned personalization methodologies can be applied also in different domains: course personalization can in fact be compared, for instance, to cultural visits personalization. Analogies and differences between course personalization and cultural visits personalization have been detected: methodologies provided for teacher's support can also be exploited for domain expert's support, especially for building personalized virtual visits, in which logical constraints among works of art can be considered regardless of physical constraints. From the visitor's point of view personalization based on previous knowledge can be considered, providing cultural visits not including works of art already visited, and producing "didactic" visits. Visitor's interests have been also taken into account, for providing an attractive cultural visits personalization.

The dissertation is organized as follows: Chapter 2 gives a wide introduction to personalization in web-based education as managed in the literature, in particular it focuses on Learning Management Systems characteristics, and proposes a non-exhaustive review about Intelligent Tutoring Systems and Adaptive Educational Hypermedia; Chapter 3 provides a detailed description of LS-Plan, focusing on the system components and on student's and teacher's functionalities; Chapter 4 shows the evaluations conducted on the system, i.e the *layered evaluation*, the *as a whole* evaluation, and the evaluations of teacher's functionalities. In Chapter 5 analogies between course personalization and cultural visits personalization are shown, together with an application to visits to *Lucus Feroniae*, an archeological site near Rome. Chapter 6 illustrates LS-Lab, describing the system architecture, the conceptual framework provided for making uniform the input of the available algorithms, the metrics currently implemented and two case studies. Finally, the last chapter illustrates conclusions and future works.

Chapter 2

Personalization in web-based education

Two are the most important families of systems for Web-based education: Learning Management Systems (LMS) on the one hand and Intelligent Tutoring Systems (ITS) and Adaptive Educational Hypermedia (AEH) on the other hand. LMS guarantee contents riusability and interoperability, and they offer numerous functionalities; however they allow a very limited, often null, personalization, as shown in Section 2.1. On the contrary, research in information systems focuses more and more on personalization; in particular, personalization in education has been sought in ITS and in AEH.

ITS are artificial intelligence-based systems, born with the aim of providing students with personal tutors, similar, as much as possible, to human tutors. These systems provide very sofisticated student modeling and personalization techniques, as shown in Section 2.2. Experience in ITS together with a rapid increase in using the web brought to a new research branch: Adaptive Educational Hypermedia. *"By Adaptive Hypermedia Systems we mean all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user."* [Bru96] *"Adaptive hypermedia is an alternative to the traditional "one-size-fits-all" approach in the development of hypermedia systems. Adaptive hypermedia systems build a model of the goals, preferences and knowledge of each individual user, and use this model throughout the interaction with the user, in order to adapt to the needs of that user."* [Bru01]. When applied to education these systems are called AEH [Bru03]. The main techniques used in these systems are *adaptive presentation*, that is adapting the contents of the hypermedia pages, and *adaptive navigation support*, that is helping the student in the hypermedia navigation, using techniques such as direct guidance and link annotation [Bru01]. The main components of an adaptive hypermedia are the user model and the adaptation methodology [BKS04]. In educational systems one of the main adaptation techniques is *Curriculum Sequencing. Curriculum Sequencing* means to *"help the student to find an "optimal path" through the learning material"* [Bru00]. Research in this field aims to automatically produce a personalized sequence of didactic materials or activities, on the basis of each student's needs, by dynamically selecting the most appropriate didactic materials at any moment [BV03]. In Section 2.3, an overview of adaptive hypermedia techniques and most important systems, is shown, focusing on educational aspects, in particular on adaptation provided through *curriculum sequencing*, based on student's knowledge and learning styles. Section 2.4 points out some limits of current systems for web-based education.

2.1 Learning Management Systems and standards for e-learning

Learning Management Systems are widespread platforms, proprietary or open-source, used for distance learning. They provide functionalities for students, teachers, tutors, administrators. Common functionalities are: course creation and delivery, user's registration, assessment and self-assessment management, logging, tracking, statistical reports, collaboration tools, such as chat and forum, scheduling tools, such as agenda, etc. Moodle, Docebo, Ilias, Lotus Learning Space, Blackboard are some of the many available Learning Management Systems.

In order to provide reusability and interoperability several initiatives for standardization have been proposed, for instance by ADL (Advanced Distributed Learning)[1], IEEE-LTSC (Institute of Electrical and Electronics Engineers - Learning Technologies Standards Committe)[2], IMS (Instructional Management Systems)[3], ARIADNE (Alliance of Remote Instructional Authoring and Distribution Networks for Europe)[4], etc. Several standards have been proposed for learning objects description, such as the IEEE LOM (Learning

[1]http://www.adlnet.org/
[2]http://ieeeltsc.org
[3]http://www.imsglobal.org/
[4]http://www.ariadne-eu.org/

Object Metadata)[5], for test description, such as the IMS QTI (Question and Test Interoperability)[6], for student description, such as the IEEE P1484.2-PAPI Learner (Personal and Private Information) and the IMS LIP (Learner Information Package)[7].

The ADL initiative, aiming to integrate several standards, proposed the SCORM (Sharable Content Object Reference Model) specification[8]. SCORM is a “standard de facto” followed by LMS and contents developers. It is currently at the SCORM 2004 4th Edition Version 1.1, but several LMS, such as Moodle, are SCORM 1.2 compliant. SCORM 1.2 involves the *Content Aggregation Model* and the *Run Time Environment* books. SCORM 2004 has included the *Sequencing and Navigation* book. The *Content Aggregation Model* book is related to contents description and packaging, and introduces the imsmanifest.xml file, necessary for describing the content package resources and their organization. The *Run Time Environment* book is responsible for the description of communications between LMS and contents, in particular it describes the API and the data model, i.e. functions and data for communications between LMS and contents. The *Sequencing and Navigation* book describes how rules for contents sequencing can be defined in the imsmanifest file.

The *Sequencing and Navigation* book has been developed on the basis of the IMS Simple Sequencing specification[9]. A lot of studies have been performed to exploit the potentialities of this specification, and of the IMS Learning Design[10] one, to provide course personalization. The aim is looking for a point of contact between adaptive hypermedia techniques and standardization: adaptive hypermedia are, in fact, advanced from personalization point of view, but lack in reusability and interoperability, on the contrary standardization assures reusability and interoperability, but provides a limited personalization from the student’s point of view and a limited usability from the teacher’s point of view. This contrast is pointed out in [AD03], considering the IMS SS specification. The authors highlight some differences between Adaptive Hypermedia Systems and IMS SS: beyond different techniques employed, different tracking mechanisms and different conceptual structures, the most significant differences between the two approaches are in the constitutional components and in their objectives. Abdullah and Davis, in fact, point out that Adap-

[5]http://ltsc.ieee.org/wg12/
[6]http://www.imsglobal.org/question/
[7]http://www.imsglobal.org/profiles/
[8]http://www.adlnet.gov/Technologies/scorm/
[9]http://www.imsglobal.org/simplesequencing/
[10]http://www.imsglobal.org/learningdesign/

tive Hypermedia Systems are "user centred" while the IMS SS specification is "instructor centred", that is Adaptive Hypermedia Systems are focused on helping the student in his free navigation of the course materials, while the IMS SS specification guides the student through the learning materials, in the way prefixed by the teacher. Moreover Adaptive Hypermedia Systems, according to the AHAM reference model [BHW99] have a Domain Model, representing the concepts of the subject and the relationships among them, a User Model, describing the user characteristics, an Adaptation Model, i.e. a Pedagogical Model, and an Adaptation Engine, responsible for adaptation. IMS SS has only an implicit User Model, in fact it provides a tracking of the user browsing behaviour and progress during the fruition of the learning materials, but it does not take into account any other individual characteristics. Also the Domain Model is implicit in the IMS SS: resources are only linked through the Sequencing Rules, expressed by the instructor. IMS SS does not support a real individualized sequencing, providing few possibilities for personalization. Moreover, as pointed out in [KS05c], it focuses only on a "single learner model": it does not consider interactions among learners and between learner and tutor, i.e. among the actors of the instructional process. Moreover Karampiperis and Sampson point out that the learning activities considered by SCORM are only content-based, so different pedagogical approaches are not taken into account. To overcome these limits the IMS-LD specification was developed. The aim of this specification is to provide a standard language for the description of learning scenarios, describing interactions among the actors of the instructional process and considering different types of learning resources. This standard approach allows the reusability of learning designs, separating them from the learning resources [KS05c]. The IMS-LD specification is based on the Educational Modelling Language (EML) [KM04]. EML, an XML-based language, allows to model different pedagogical approaches, different types of learning resources (included tests), and, above all, introduces the possibility of a kind of user modelling and personalization. User modelling is obtained through the definition of properties, personalization is based on if-then rules, in which the if condition can be formulated in terms of user properties. The relationship between adaptation and IMS-LD is investigated in [BTK06, CB06]. IMS-LD can be used to define adaptive learning designs [KB05, TH05], but the required designer's effort is consistent: he has to foresee all the possibilities and consequently to define the correspondent user properties and adaptation rules. Some authoring tools have been developed to support instructors: ASK-LDT [KS05c] provides a graphical user interface, that helps also in the definition of personalized designs, Boticario et al. have developed Alfanet[11] [SBB04], a LMS, based on IMS-LD. Alfanet tries to combine design and runtime adaptation, providing both IMS-LD compliant authoring tool and agents performing adaptation. The authoring is facilitated by the definition of pre-fixed pedagogical scenarios, described in terms of IMS-LD. Runtime adaptation is provided by means of recommendation, given to the student on the basis of a collaborative approach, i.e. on the basis of positive results obtained with students with similar characteristics (clusters of students are derived through machine learning techniques). A synthetic review of IMS-LD authoring tools is provided in [BG05]. Research efforts are yet directed in making easier and as more automatic as possible, the IMS-LD designer’s task. Morales, Castillo, Fernandez-Olivares and Gonzalez-Ferrer [MCFOGF08] propose the use of the LPG-td planner [GS02] to build personalized IMS-LD, translated as a guideline for the student in ILIAS.

2.2 Intelligent Tutoring Systems

ITS have been developed by 1960s-1970s, with the aim of providing, in an automatic way, students with personal tutors [Bru94]. They are rule-based systems, in which three types of knowledge are represented:

- knowledge of the subject matter: expert’s knowledge about concepts and procedures related to the topics to be learnt;
- knowledge of teaching strategies: methodological didactic aspects, for instance the necessity of a step-by-step proof or exercise development, for a beginner student;
- knowledge of the student: student’s knowledge, behavior, and misconceptions.

Knowledge is used for performing tutoring actions: help during problem solving, identification of student’s bugs and related causes, choice of the teaching action to be performed. These actions are performed taking into account the Student Model, that is considering the student characteristics and knowledge, updated during the study. In particular questionnaires and problems are submitted to the student and his responses are compared to the expert’s ones: the student model is consequently updated and the teaching strategy to be applied is selected. According to the three types of knowledge of an ITS, its three main

[11]documentation available at http://dspace.ou.nl/handle/1820/2

components are: the Student Module, that manages the Student Model, the Expertise Module, that compares students' responses and actions to expert's ones, and the Tutoring Module, that is responsible for selecting the teaching strategies to be applied and for managing student's interactions. Brusilovsky in [Bru94] proposes a review about student models in ITS and distinguishes: i) what information a student model contains; ii) how a student model is initialized and updated; iii) how a student model is used.

Regarding the first item, ITS are mainly focused on modeling the student knowledge, that is modeling both the knowledge possessed with respect to the topics of the course and misconceptions. The most used model for representing the knowledge possessed by the student is the *overlay model*. It divides the course topics in small elements and assigns a boolean value or a numeric value to each element for representing that the student possesses or does not possess the element and with what degree. The student knowledge is therefore represented at each instant of time as a subset of the expert's knowledge. The numeric value can be an integer or a probability measure.

An important theory for modeling student's knowledge, the *ACT theory*, has been proposed by Anderson and applied to different tutoring systems for teaching LISP, geometry and algebra [ACKP95, ABCL90]. The ACT theory distinguishes declarative knowledge and procedural knowledge. Declarative knowledge is not linked to how it will be used, it is memorized for instance when someone reads a text. Procedural knowledge is only acquired by using declarative knowledge, it is often acquired through a trial and error mechanism, and it is represented by a set of production rules, i.e. a skill. Production rules are learnt through problem solving activities. Both declarative and procedural knowledge acquire strength with practice.

Student's misconceptions are represented through a *bug model*, that allows to identify incorrect knowledge and its causes. For instance, a perturbation model represents errors as application of perturbations of an element of expert's knowledge. In [SS98] Sison and Shimura analyze possible uses of machine learning for student modeling. They present a review of systems using machine learning for student modeling and focus on the utility of these techniques for extending bug libraries.

Regarding the second item, information for constructing and updating the student model can be obtained by the system either in an *explicit* way, by a direct dialogue with the student, or in an *implicit* way, by observing student's progress, difficulties, behavior in problem solving, etc. Knowledge can also be inferred, according with relations among elements of knowledge; for instance if a prerequisite relation between elements A and B is present, the knowledge of the prerequisite A can be inferred by the knowledge of the element B. In order to initialize the student model one or several tests can be proposed to the student before the course taking. Also student's background and previous experiences can be considered in the model. The model is updated on the basis of several information:

- student responses to simple or complex questions, that is questions related to a single element of knowledge or to more than one element of knowledge;
- analysis of student's problem solving, that is observing the rules applied by the student and comparing them to the correct and incorrect rules known by the system [ACKP95, ABCL90];
- interpretation of student's actions based on *plan recognition*, that is the system knowledge, for instance in ITS for teaching programming, of the plan chosen by the student for his program. Plan recognition can be obtained either by limiting the possible student's choices and asking an explicit disambiguation to the student when necessary [ACKP95, ABCL90], or allowing uncertainty, like in the Andes system that uses Bayesian networks for finding the most likely explanation of a given student action [CGV02].

Machine learning techniques have been used in several systems for constructing and updating student models, such as in HYDRIVE [MG95] and Andes [CGV02], a review is proposed in [SS98] and in [Jam96].

Regarding the third item, the student model is mainly used for:

- detecting a lack of knowledge and consequently proposing the suitable teaching action. It can be done either performing a goal-oriented tutoring or providing help in an active or passive way, that is offering support to the student or answering to student's request of hints respectively;
- error remediation, that is detecting a misconception and providing explanations, or prompts or showing the solution to the student.

Experience in ITS together with a rapid increase in using the web brought to a new research branch: Adaptive Educational Hypermedia.

2.3 Adaptive systems for web-based education

Brusilovsky points out in [BM07] that *"User modeling and adaptation are two sides of the same coin"*. Two are in fact the main interrelated questions to be answered for producing an adaptive system:

- *"adapting to what?"*. What are the user's characteristics to be taken into account? To what user's characteristics the system has to be adapted? The main used features in adaptive hypermedia are: knowledge, goals, background, interests, personal traits, such as cognitive styles and learning styles [BM07].
- *"what can be adapted?"*. What are the features of the system changing according to the user's characteristics? It is possible to adapt contents of the hypermedia pages, i.e. *adaptive presentation*, and links helping the user in the hypermedia navigation, i.e. *adaptive navigation support.*

In [Bru01] Brusilovsky provides a taxonomy of adaptive hypermedia technologies. *Adaptive presentation* includes for instance inserting, removing, sorting of page fragments, techniques widely used in AHA! [BSS06]; *adaptive navigation support* includes direct guidance, link annotation, link hiding, link sorting. When applied to education adaptive systems are called AEH [Bru03]. In several educational hypermedia *adaptive navigation support* is based on a technique inherited by ITS, that is *Curriculum sequencing* [Bru00], as shown in the following. According to user modeling in educational domains, the most considered feature to be taken into account is the student's knowledge. Moreover several studies have been performed for considering student's learning styles.

Student Modeling techniques

The overlay approach, inherited by ITS, is the most used one for modeling user knowledge [BM07]. It represents student's knowledge as a subset of the domain model, i.e. of the expert's knowledge of the subject. The overlay model can use Boolean, qualitative or quantitative values for indicating *if and how much* a fragment of the domain is thought to be already known by the student; it can be layered for taking into account the different sources used for the estimations of the user knowledge. Moreover, the overlay approach can model conceptual or procedural knowledge and it can be expanded, similarly as in ITS, through a *bug model* for taking into consideration user's misconceptions. Bug models are especially used for procedural knowledge; their practical use is complicated and is limited to Intelligent Tutoring Systems based on simple domains and to few adaptive educational systems focused on problem solving. Different techniques have been proposed for estimating the student's knowledge, such as the ones proposed in the following systems:

- IWT [SCGM08] focuses on tests results for estimating the student's knowledge: it uses an overlay model, in which each concept of the domain is associated to two numerical value. The first value ranges between 0 and 1, is obtained on the basis of tests results, and represents the student's degree of knowledge of the given concept. The second numerical value represents the number of tests performed by the student for assessing the concept.

- AHA! does not exploit assessment for student model updating and it is based only on the user's browsing behavior. However it provides an interesting mechanism of knowledge propagation, that is, modifying the estimate of the knowledge of a given concept on the basis of the estimate of the knowledge of a related concept. AHA! allows the authors to define different relationships among concepts and the correspondent knowledge propagation mechanism [BAR02].

- Netcoach [WKW01], developed on the basis of the latest version of ELM-ART [WB01], uses a layered overlay model, composed by: pages visited by the student; tests; inferences about the knowledge of a concept on the basis of the student's success in more advanced ones; concepts marked as known by the student. Netcoach builds a fifth layer, the learned layer, on the basis of the other levels, i.e. a concept is assumed learnt if it is tested, inferred or marked and, should there be no tests, if it is visited.

- TADV [KDB04, KDB07] uses fuzzy techniques for modeling students, groups, and classes, using tracking data of Web Content Management Systems. In particular for modeling student's knowledge TADV uses browsing of learning objects, quiz results and partecipation in discussion forum.

- TANGOW [ACM$^+$06] allows to store information on the actions the student performed while interacting with the system, including exercises scores and visited pages. Moreover it provides a formalism that allows the course author to specify the necessary adaptation rules.

Uncertainty is an issue in student modeling: fuzzy approaches are presented in [KDB04, Kav04] and an interesting discussion about bayesian networks, mainly used for modeling student's knowledge, is proposed in [BM07]. These approaches are however more considered in ITS then in AEH [BM07].

Beyond student's knowledge, a personal trait more and more considered in AEH is learning style. The actual effectiveness of learning styles-based adaptation is still a matter for discussion: it is questioned and supported, as illustrated in [BM07]. An empirical evaluation in [BFB07] shows no relevant improvement in the attainment of primary schools students using an adaptive learning styles-based hypermedia with respect to their colleagues treated more traditionally (in particular, the learning styles were modeled through the sequential-global dimension of Felder and Silverman's Learning Styles Model [FS88]).

On the other hand, many studies have been conducted applying the idea that teaching strategies, based also on student learning styles, might increase the learner's motivation, comprehension, participation and learning effectiveness. In particular Felder and Silverman's Learning Styles Model has been often taken into consideration in the literature ([SCGM08, GK07, BHF03, CHL99, ACM$^+$06]). The reason for such attention appears to be manifold: 1) this model is a combination of other models, such as Kolb's and Pask's ones [Kol84, Pas76]; 2) it provides a numerical evaluation of learning styles, which is a useful factor in computer based systems, and 3) its reliability and validity has been successfully tested, such as in [LLWF05, ZW01].

Coffield [CMHE04] states that “Different theorists make different claims for the degree of stability within their model of styles”; following this problem Felder in [FS05] states that learning styles are tendencies and they may change during the educational experiences; this claim has been also empirically shown in [BFB07].

Learning styles and in particular the Felder and Silverman's model, are used in a lot of systems, such as the following:

- the add-on for the Moodle Learning Management System proposed in [GK07], where a course personalization, based on learning styles, is presented;
- the system proposed in [BHF03], in which an interesting adaptive interface is included;
- the TANGOW system [ACM$^+$06] that uses two dimensions of the Felder and Silverman's Model; it initializes the student model in an explicit way, through the Felder and Soloman's Index of Learning Styles (ILS) Ques-

tionnaire, updates such model in an implicit way, through observations of the student's browsing behavior and uses the model information also to encourage collaborative learning through group formation;

- the CS383 system [CHL99] and the IWT system [SCGM08] that propose an adaptive presentation based on learning material typologies.

The sequencing problem

Several approaches for *Curriculum sequencing* have been proposed in the literature: rule-based sequencing, as in the AHA! system [BSS06]; planning-based sequencing, as in the work of Baldoni et al. [BBB$^+$07, BBPT04]; graph-based sequencing as in the IWT system [SCGM08], in the KBS-Hyperbook system [HN01], in the LecompS system [ST03] and in the DCG system [Vas92, BV03].

It is possible to classify course sequencing techniques into two categories:

- sequencing that plans the entire learning path at the beginning, then modifies it, when the study does not succeed as it should, e.g.: Dynamic Courseware Generation, the work of Baldoni et al. and the IWT system;
- sequencing obtained in an implicit way, step-by-step, through adaptive navigation support techniques, such as adaptive link annotation and direct guidance [Bru01](like the AHA! system and the ELM-ART system [WB01]).

The first approach is used in the work of Baldoni et al. in which course sequencing is seen as a planning problem: learning resources (*learning objects* in [BBPT04] or *courses* in [BBB$^+$07]) are seen as actions, with preconditions and effects, i.e. with prerequisites and acquired competencies, specified in the "Classification" tag of the IEEE LOM standard[12]. The definition of these metadata is based on ontologies of interest, to guarantee shared meanings, interoperability and reusability, ensuring a Semantic Web perspective. However in these approaches, "tagging" is a bottleneck: teachers may find it hard to adhere to predefined ontologies, they might be confused while searching for a particular term in predefined vocabularies to express their personal meaning of a concept, they might not share the relationship among concepts defined in the ontology of interest, and they might find this kind of resource-tagging boring, for it cannot be completely automated. Moreover, the approach proposed in

[12]http://ltsc.ieee.org/wg12/

[BBPT04, BBB$^+$07] does not consider personalization at the level of learning materials used to explain a given concept, so the teacher cannot express how to choose the most appropriate learning material among those that explain the same concept.

DCG [BV03] creates a plan of the course contents, follows the student during the fruition of the course, and makes a re-planning if the student fails to demonstrate the acquisition of a concept. Sequencing in DCG is sophisticated, and considers some personal characteristics, although, to my knowledge, it does not let re-sequencing actions depend on the occurring learning styles modifications.

IWT uses an ontology to take into account pedagogical relationships among concepts. The system uses both the ontology and the student model to generate the personalized course, covering the target knowledge defined by the student. In particular the ontology provides three types of relationships: HasPart, i.e. the concept A is composed by concepts B, C, D, etc; Requires, i.e. the prerequisite relation; Suggested Order, i.e. a weak prerequisite relationship. The concepts to be studied, that the authors call Learning Path, are decided on the basis of these relationships. The ExplainedBy relationship links concepts to Learning Objects (LO). The Presentation, i.e. the actual learning materials proposed to the student, is the set of LOs selected on the basis: of the Learning Path, of the ExplainedBy relationships and of the students learning styles. The automatic course generation process takes into consideration the target concepts defined by the student. It follows the HasPart and Requires relationships, creating the set of all necessary LOs. Moreover, it selects and excludes, from the above-mentioned LOs, concepts the student already knows. This set is then ordered, using a pre-built graph that represents the ontology, expanded with all implicit relationships. This ordering process is performed starting from the node representing the target concepts and performing a depth-first visit of the edges. IWT allows the course designer to verify the presence of loops in the ontology. Moreover, learning styles are associated to learning materials types, for describing the suitability of a LO typology for a student with given learning styles. This information is used for choosing among equivalent available LOs the most suitable for the student.

The second approach is applied in several systems, for instance in AHA! and in ELM-ART. AHA! is a very flexible system, where adaptation can be performed both through navigation support and in contents, including *fragments adaptation* [Bru01]. It is based on rules, managing both user modeling and adaptation strategies. The management of such rules, and in particular their termination and confluence, might be a drawback in AHA!, in fact it guarantees termination through enforcements, while the confluence problem is left open (see [Wu02] for a complete dissertation about these problems). Another drawback is then related to producing a specification of such rules, suitable for their use in the system. So, significant efforts are presently devoted to the development of advanced authoring tools: for example MOT [CM03a] allows authoring based on LAOS [CM03b] and LAG [CC03] models, making it possible to use an adaptation language to program adaptive behaviors, which will be compiled in suitable rules. However, authors are required either to possess programming skills or to rely upon pre-defined strategies.

The first version of ELM-ART, presented in [BSW96], showed how ITS technologies can be implemented in a web-based environment, proposing an adaptive electronic textbook with an integrated problem solving environment, to support study of the LISP programming language. Based on the approach to adaptivity proposed in the first version of ELM-ART, but allowing domain independence, in [BES98] Interbook, an authoring and delivering tool for adaptive electronic textbooks, was presented. An electronic textbook, according to the author's definition, can be any hierarchically structured hypermedia material. The textbook contains chapters, sections, subsections, terminal pages, all said "units". Terminal units are linked to domain concepts, in particular to outcome concepts and to prerequisites concepts. Outcome concepts are explained in the unit, prerequisite concepts are concepts that the student has to know to understand the unit. Interbook provides adaptive navigation support through adaptive link annotation and direct guidance, based on the current student's knowledge, i.e. the content of a unit can be known, ready to be learnt or not ready to be learnt (prerequisites do not fulfilled in the student model). Student is guided step-by-step, like in AHA!: the system does not produce a complete learning path at the beginning, but recommends suitable units on the basis of the current student model. The student can choose among different recommended units or can ask help to the system that suggests, applying heuristics, the most suitable unit among the recommended ones. Moreover Interbook has a prerequisite-based help, i.e. it provides to the student a link to a recommended list of prerequisites of the current unit. If the student finds difficulties in the study of a given unit, he can use this link and he receives a recommended list of units to study, sorted on the basis of his current model. Regarding authoring, Interbook expects the textbook described in a Word file. Specific formats and characters styles within the Word file allow to define sections, subsections, concepts annotations.

In the direction of close systems to teachers an interesting approach is proposed by Karampiperis [KS05a, KS05b], that, observing expert's suitability evaluation of given Learning Objects for students with given characteristics, produces a decision model that mimics expert's decisions. Obviously this approach needs a start-up phase.

2.4 Limits of current systems

As shown in Section 2.1, LMS are widespread distance learning platforms, they provide useful functionalities, but they do not allow personalization based on student's knowledge, needs, learning styles, interests. Moreover they do not adapt the learning experience on the basis of student's progress and difficulties. However efficiency and effectiveness of web-based distance learning can be greatly enhanced when the learning activities are devised and presented in accordance with the personal differences existing among learners: such differences can be seen on several different levels, such as the knowledge and skills already possessed, the aims of learning, the learning styles, the motivation and interests. Individualized content might be better understood by the student and might be considered relevant to his apparent needs; this, in turn, may positively affect the learner's motivation and collaboration, for a more efficient learning. All the above motivations might be considered different facets of the term "learner's satisfaction". On the other hand, the aspect of "teacher's satisfaction" is quite seldom taken into account: producing learning material is a task that needs a considerable effort by the teacher; moreover, teachers are often asked to manually sequence the learning materials, complying with "difficult-to-use" de-facto standards, or to define appropriate metadata or rules in order to automatically sequence the material. Learning material production cannot be performed in an automatic way; the burden of producing good quality instructional material can only be lightened through reusability; yet, content selection and sequencing can be made easier by using suitable tools. "Learner's and teacher's satisfaction" are two interrelated features for building personalized and effective courses. One of the main problems to be addressed is to sequence the contents automatically, both following the student during his fruition of the course, and allowing the teacher to easily express his didactic strategies, i.e. as-light-as-possible approaches to course personalization, from the teacher's point of view.

Chapter 3

LS-Plan: design features

LS-Plan implements methodologies that aim to carry out a reasonable compromise between an effective personalization from the student’s point of view and from the teacher’s point of view [LSV08, LSV09a, LSTV09a]. From the student’s point of view, the system guides the student during the fruition of the course, like a teacher could do; from the teacher’s point of view an as-light-as-possible approach is sought. LS-Plan is a system capable of providing educational hypermedia with adaptation and personalization. Unlike the other adaptive educational hypermedia, LS-Plan provides a personalization engine that can be plugged in any educational system. The main components of the system are the user model and the adaptation strategy. The student model takes into account student’s knowledge and learning styles. The adaptation strategy provides personalization based on a suitable curriculum sequencing carried out both planning the entire learning path at the beginning of the course and, step-by-step, during the course taking. Course concepts, i.e. the domain, are modeled as atomic elements of knowledge, according to the Knowledge Space Theory [FKV$^+$90]. The system models the student’s knowledge by a qualitative overlay model, based on three levels of Bloom’s Taxonomy [Blo64]. The student’s knowledge is estimated on the basis of tests, and, if they are not present, it is estimated by considering the “pages-seen”. Some adaptive hypermedia use an approach for student knowledge management similar to the LS-Plan one, and in some respects more advanced, for instance NetCoach, that uses a multi-layered overlay model, and AHA!, that allows a mechanism of knowledge propagation, as seen in Section 2.3. The LS-Plan approach is currently less granular and it is heavily based on tests: browsing a material or acquiring a more advanced concept is not considered sufficient (if tests are available) for estimating a known concept. LS-Plan, differently from ELM-ART, does not provide a bug model: it is hard to model user misconceptions in a wide and non-predefined domain. However through its adaptation algorithm, LS-Plan allows to estimate the presence of a lapse of memory or of a wrong estimation of student's knowledge about a given concept, i.e. it allows a modeling of "forgetfulness". Baldoni et al., in [BBPT04] and [BBB$^+$07], propose a management of student's knowledge very similar to the LS-Plan one, at least in the phase of course construction. However the authors assume that the user's competencies can only increase during the study, without considering "forgetfulness". Learning styles are modeled according to the Felder and Silverman's Learning Styles Model [FS88]. The reason for such choice appears to be manifold: 1) this model is a combination of other models, such as Kolb's and Pask's ones [Kol84, Pas76]; 2) it provides a numerical evaluation of learning styles, and 3) its reliability and validity has been successfully tested, such as in [LLWF05, ZW01].

Two main features of the LS-Plan system concern the learning styles management and the adaptation algorithm. Adhering to the Felder and Silverman's Learning Styles Model, LS-Plan models learning styles as tendencies [FS05] and estimates how the didactic material affects the success of the learning activity. In particular, the teacher associates, with the learning nodes, some weights (associated to learning styles) that represent the suitability of that material for learning preferences. If the student studies a given material with success, it means that the presentation style is consistent with the student's way of learning, so the student's learning styles move towards the learning styles of the node; on the contrary, if the study does not succeed as it should, the student's learning styles moves in the opposite direction of the learning styles of the node. The information gathered from the student's behavior is used both for learning styles refinement procedures and for evaluating the effectiveness of the current teaching strategy, modifying it, if necessary. In particular the adaptation algorithm mimics the teacher's behavior in presence of learner's difficulties in the study of a given topic: like a teacher usually does, the system tries to explain the same learning material again, supposing that the student has not paid sufficient attention on it; in case of a new student's failure, the system, if possible, tries to explain the same concept in a different way; in presence of a new failure a prerequisites check is suggested. The proposed materials and their sequencing are based on the student's knowledge, updated during the fruition of the course, and on his learning styles. Using the Pdk planner [MLOP07] for sequencing course contents allows the teacher to express in an easy way his didactic strategies, such as the difficulty level of the course, or the presence of mandatory contents for all the learners, or a particular preferred approach to teach a topic. The Pdk planner provides, ahead of the course, the whole learning path that is modified, possibly step-by-step, while the course taking. According to teacher's functionalities LS-Plan learning styles management is finer grained than the IWT one: the system allows teachers to assign different weights to the actual learning material - and not only to its typology - according to the four Felder-Silverman's Learning Styles dimensions. In this way the system provides the teacher with the possibility to implement different didactic strategies for different learners. The effort required to the teacher is as near as possible to his "way of thinking": he is required to specify few metadata, necessary for characterizing learning materials, such as prerequisite relations, concepts acquired through the study of the given material, and suitability of contents for a given type of student; he is not forced to adhere to predefined ontologies, differently from [BBPT04, BBB+07]; he is helped by a graphical interface, allowing a global vision of the course and he can express didactic preferencies; he is not requested to possess programming skills; he is not requested to adopt "difficult-to-use" de-facto standards or to define appropriate rules in order to automatically sequence the material.

Section 3.1 describes the overall system, showing how its components, i.e. the Adaptation Engine, the Pdk Planner and the Teacher Assistant, work together for carrying out the complete functioning of the system; then section 3.2 and section 3.3 focus on the functionalities provided for students and teachers respectively. In particular section 3.2 explains in detail the student modeling and the adaptation decision making methodologies, provided by the Adaptation Engine, whereas section 3.3 focuses on the functionalities provided by the Pdk Planner and by the Teacher Assistant for supporting the teacher in course personalization.

3.1 The system

Fig. 3.1 shows the overall system. The LS-Plan system provides the educational hypermedia with adaptivity; the main components are highlighted with grey blocks and described in the following.

The *Teacher Assistant* is responsible for the teacher's functionalities. It allows the teacher to arrange a pool of learning objects, i.e., learning nodes, that is to define all the metadata necessary to tag such materials. This information is stored in a database, belonging to LS-Plan, while the actual repository of

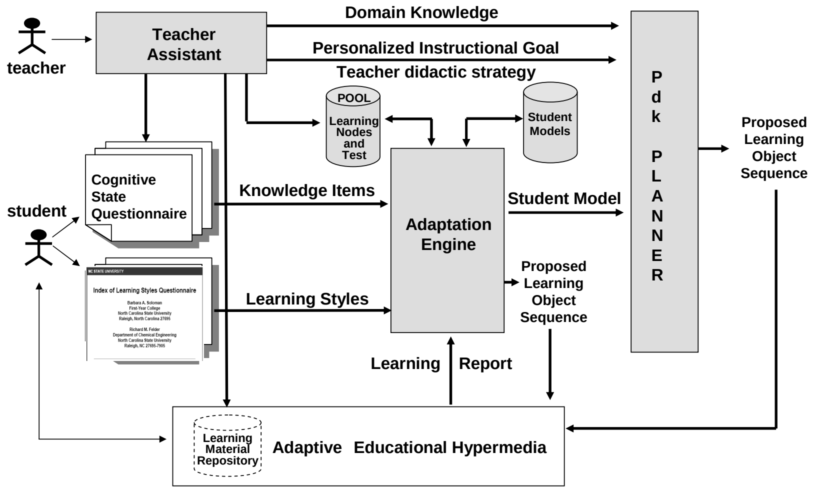

teacher
Teacher Assistant
Domain Knowledge
Personalized Instructional Goal
Teacher didactic strategy
POOL
Learning Nodes and Test
Student Models
Pdk PLANNER
Proposed Learning Object Sequence
Cognitive State Questionnaire
student
Knowledge Items
Adaptation Engine
Student Model
Index of Learning Styles Questionnaire
Learning Styles
Proposed Learning Object Sequence
Learning
Report
Learning Material Repository
Adaptive Educational Hypermedia

Figure 3.1: The Functional Schema of the Adaptive System. Grey blocks form LS-Plan.

learning material is stored in the educational hypermedia. The Teacher Assistant allows also the teacher to define tests related to learning nodes, and to create the initial *Cognitive State Questionnaire* to evaluate the student's starting knowledge, that is, the knowledge already possessed by the student with respect to the topic to be learned. The student fills in both the *Cognitive State Questionnaire* and the *Index of Learning Styles (ILS) Questionnaire*, i.e. a test, developed by Felder and Soloman[1], which extracts the student's learning preferences according to the four dimensions of the Felder and Silverman's Model: active-reflective, sensing-intuitive, visual-verbal, sequential-global. This information is managed by the Adaptation Engine, in order to initialize the student

[1]available at http://www.engr.ncsu.edu/learningstyles/ilsweb.html

model, which is then stored in the *Student Models Database*. Through the Teacher Assistant, the teacher also specifies his didactic strategies and defines the instructional goal for each student. This information, together with both the results of the two initial questionnaires and the descriptions of the learning nodes, i.e. the Domain Knowledge, is coded in PDDL (see Section 3.3) files and sent to the Pdk Planner. In particular information about the student, the didactic strategies and the instructional goal, are used for building the problem.pddl file and information about the domain constitute the domain.pddl file.

The *Pdk Planner* uses the domain.pddl file and the problem.pddl file and produces in output to the hypermedia a personalized Learning Object Sequence (*LOS*) for the given student. The student is not forced to follow the *LOS* generated by the planner.

The *Adaptation Engine* follows the student's progress during the fruition of the course, taking into account results from intermediate questionnaires and the time spent studying each learning node. This information is used both for updating the student model and for the adaptation decision-making, as is discussed in Section 3.2.

Before describing more in depth the components of the system, the algorithms used for managing the student model updating, and the adaptation decision making, some definitions are necessary.

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

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

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

Only three out of the six levels of Bloom's taxonomy in the cognitive area have been chosen, in order to test the correct behavior of the planner: it is possible to provide the KI with all the six levels, or to consider different meanings for levels.

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

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

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

The range $[-11, +11]$ has been chosen according to the Felder-Soloman's ILS scale.

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

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

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

AK *Acquired Knowledge. It is a KI_ℓ that represents the knowledge that the student acquires at a given level as specified in Definition 1, after having passed the assessment test related to the KI_ℓ of the node (i.e. the obtained score is greater than the success threshold σ_{KI_ℓ}). If such a test is not present in the node the AK is considered acquired anyway.*

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

LS *is given in Definition 2 and represents learning preferences for which the material is suitable.*

T *is a pair of reals $T = (t_{min}, t_{max})$ that represents the estimated time interval for studying the material of the node, as prefixed by the teacher. Obviously it is not possible to know if the fruition time (t_f) is actually spent on studying or if it is affected by other factors. However, the thresholds t_{min} and t_{max}, allow to eliminate at least two student behaviors: the so-called "coffee break" effect, when the fruition time t_f is greater than t_{max}, and a casual browsing of a given material, when t_f is less than t_{min}.*

Definition 4 (pool). *A pool is the particular set of LN, selected or created by the teacher in order to arrange a course about a particular topic.*

Definition 5 (Domain Knowledge). *The Domain Knowledge DK is the set of all the KI present in a pool.*

Definition 6 (Cognitive State). *The Cognitive State CS is the set of all the KI_ℓ possessed by the student with respect to the given topic: $CS \subseteq DK$.*

Definition 7 (Student Model). *The student model SM is a pair:*

$$SM = (CS, LS)$$

where, CS is given in Definition 6 and LS is given in Definition 2.

Definition 8 (Target Knowledge). *The target knowledge TK is the knowledge to be acquired by the student through the course taking.*

Definition 9 (Starting Knowledge). *The starting knowledge SK is the initial CS, that is the knowledge about the topics to be learnt, possessed by the student before the course taking.*

Definition 10 (Test). *A Test is a set of questions. S_{KI_ℓ} is the score associated with a test: it assesses the student's knowledge of the single KI_ℓ.*

Questions are currently related to the acquirement of a given knowledge item. Including questions into learning nodes treating such topics allows to "contextualize" the questions. However, the separation of the test from the learning nodes, ensuring the association of a given question with more than one knowledge item, can be feasible.

In the following sections the main components of LS-Plan are illustrated, focusing on the "actors" of the learning process taken into account. The Adaptation Engine, is, in fact, mainly responsible for student's features, providing the student model management and the course adaptation to student's progress and needs (see Section 3.2); the Teacher Assistant and the Pdk Planner allow, instead, teacher's functionalities (see Section 3.3).

3.2 Student-centered features

The Adaptation Engine

In this section the mechanisms of the student model management, i.e., the initialization and the updating processes, and the related adaptation strategies are shown. These mechanisms are carried out by the Adaptation Engine, a software module developed in Java programming language.

Student Model Initialization

At the first access to the system the student fills in the *Cognitive State Questionnaire* consisting of some questions, related to the knowledge items of the

Domain. All the acquired knowledge items constitute the Starting Knowledge, that is they initialize the cognitive state CS, which can also be an empty set, if the student does not know anything about the domain. The student also fills in the *ILS Questionnaire* whose results are used to initialize his own learning styles. On the basis of this information the system proposes a first learning path and the student is free to follow or not to follow it. In any case the student can choose a LN and, consequently, an updated SM will be generated by the system, as shown in the next section.

Student Model Updating and Adaptation Methodology

At each step of the learning process, i.e., after the student studies the contents of a Learning Node, the algorithm carries out two main actions: 1) update of the student model, and 2) computation of the Next Node to be proposed, together with the new Learning Objects Sequence. Fig. 3.2 and Fig. 3.3 present the algorithms related to these two actions, respectively. Basically the idea is to work as the teacher would do: re-explaining the failing concept (proposing the same learning material as before), then trying to propose different learning material for the same concept, and finally, on further fail, assuming that some of the prerequisites, previously taken for granted, are the source of the problem and suggesting them for re-checking.

When the student studies a LN, the function UPDATESM is activated taking in input the LN and the current SM. The function TIMESPENTONTHENODE computes and returns the time t_f, that is the time the student spent on the node. The function COMPUTESCOREPOSTTEST computes and returns the score taken by the student in the post-test related to the KI_ℓ of the node, namely, the AK related to the LN. Should the post-test not be provided, a score of 0 is assumed and the KI_ℓ related to that node is considered as acquired, though without updating the student's LS. On the other hand, if the post-test is available, the student's LS are updated according to the LS associated with the node, to the fruition time t_f, and to the score obtained with the post-test. In particular, if the KI_ℓ is acquired, that learning material can be considered adequate for the case, so the student's LS can be updated towards the LS of the node, by an extent depending directly on score and inversely on time t_f. On the contrary, if the knowledge item is not acquired the opposite behavior is applied. Such modifications are to be considered "adjustments" for the present LS estimate, so they are actually quantified as values in $[0, 1]$. The two functions η_1 and η_2 ranges in $[0, 1]$, and are defined on the basis both of the function $\alpha(S_{KI_\ell})$, that takes into account the dependency of the learning

```
UPDATESM (LearningNode $LN$ , StudentModel $SM_{old}$ )
$t_f \leftarrow$ TIMESPENTONTHENODE ( $LN$ )
$S_{KI_\ell} \leftarrow$ COMPUTESCOREPOSTTEST ( $KI_\ell$ )
/* Learning Styles Updating */
if (not post-test) then
    $LS_{new} \leftarrow LS_{old}$
else
    if ($KI$ is acquired) then
        for all $D_i$
            $D_{i_{new}} = D_{i_{old}} + \eta_1(t_f, S_{KI_\ell}) sign(\Delta D_i)$
    else
        for all $D_i$
            $D_{i_{new}} = D_{i_{old}} - \eta_2(t_f, S_{KI_\ell}) sign(\Delta D_i)$
    $LS_{new} \leftarrow \langle D_{1_{new}}, D_{2_{new}}, D_{3_{new}}, D_{4_{new}} \rangle$
/* Cognitive State Updating */
if ((not post-test) or ($KI$ acquired)) then
    $CS_{new} \leftarrow CS_{old} \cup AK$
else
    $CS_{new} \leftarrow CS_{old}$
$SM_{new} \leftarrow (CS_{new}, LS_{new})$
return $SM_{new}$
```

Figure 3.2: The function UpdateSM.

styles updating by the score obtained in the test, and of the function $\beta(t_f)$, that considers the time spent in the study. In particular $\alpha(S_{KI_\ell})$ is given by the following formula:

$$\alpha(S_{KI_\ell}) = \frac{S_{KI_\ell} - S_{min}}{S_{max} - S_{min}}$$

where $S_{min} \leq S_{KI_\ell} \leq S_{max}$ being S_{min} and S_{max} the minimum and maximum score of the test, as fixed by the teacher. $\beta(t_f)$ is given by the following formula:

$$\beta(t_f) = \frac{t_f - t_{min}}{t_{max} - t_{min}}$$

where $t_{min} \leq t_f \leq t_{max}$. Both $\alpha(S_{KI_\ell})$ and $\beta(t_f)$ ranges in $[0, 1]$.

The two functions η_1 and η_2 "quantify" the updating process if the KI_ℓ is acquired or is not acquired, respectively. In particular if the KI_ℓ is acquired ($S_{KI_\ell} \geq \sigma_{KI_\ell}$), the student LS will be updated towards the LS of the node. The more is the score, the more is the contribution given by $\alpha(S_{KI_\ell})$ to the LS updating, the less is the time, the more is the contribution of $\beta(t_f)$. The function that updates the LS is then:

$$\eta_1(t_f, S_{KI_\ell}) = \frac{\alpha(S_{KI_\ell}) + (1 - \beta(t_f))}{2}$$

If the KI_ℓ is not acquired, the student's LS will be updated towards the opposite direction of the LS of the node. The less is the score, the more is the contribution given by $\alpha(S_{KI_\ell})$ to the LS updating, the more is the time, the more is the contribution of $\beta(t_f)$. The function that updates the LS is then:

$$\eta_2(t_f, S_{KI_\ell}) = \frac{(1 - \alpha(S_{KI_\ell})) + \beta(t_f)}{2}$$

The increasing or decreasing of the LS values are decided on the sign of the difference between the student's D_i and the D_i of the node: $\Delta D_i = D_{i_{LN}} - D_{i_{old}}$. The student's D_i will change according with the following formula if the KI_ℓ is acquired:

$$D_{i_{new}} = D_{i_{old}} + \eta_1(t_f, S_{KI_\ell}) sign(\Delta D_i)$$

and according with the following formula if the KI_ℓ is not acquired:

$$D_{i_{new}} = D_{i_{old}} - \eta_2(t_f, S_{KI_\ell}) sign(\Delta D_i)$$

If the post-test is not present or KI_ℓ is acquired the list of the KI_ℓ possessed by the student is updated:

$$CS_{new} \leftarrow CS_{old} \cup AK$$

On the basis of the above updating it will be decided the next node to be proposed, as described in the function NextNode.

The function NextNode proposes the next node to be learned on the basis of the new student model as described in Fig. 3.3. If a D_i, as given in Def. 2

```
NEXTNODE (LearningNode $LN$ , StudentModel $SM_{new}$ )
if ($\exists D_i$ that changed sign) then
    REPLAN($SM$ )
    return ($LN^{first}, SM$ )
else
    if (time-out( $t_f$ ))   then
            return ($LN, SM$ )
    else
        $LN' \leftarrow$ CHECKCLOSESTNODE ( $LN$ , $SM$ )
        if ($LN' \neq NIL$ ) then
            return ($LN', SM$ )
        else
            $L \leftarrow$ ORDEREDPREDECESSORSLIST  ( $LN$ )
            if ($L \neq NIL$ ) then
                $\forall LN_i \in L$ ,  if ($AK \in CS$ )  then
                    $CS \leftarrow CS - AK$
                ADD_L_TOPLAN  ( $L$ )
                return ($LN^{first}, SM$ )
            else
                REPLAN(SM)
                return ($LN^{first}, SM$ )
```

Figure 3.3: The function NextNode.

changes sign, a significant variation in the student LS is present and makes it necessary to re-plan the LOS: the algorithm suggests the first LN of the new LOS computed by the planner. If the student does not pass the test, the time t_f is examined: the Boolean function "time-out" checks whether t_f is out of range and if it is the first time that the LN has been studied. Should that be the case, the system proposes once again the same node to the student. After the second unsuccessful trial, the system applies the function CHECKCLOSESTNODE, that looks for the closest node, not already visited by the student, computed by selecting an alternative node, LN' with the same RK and the same AK of the current LN, that is the node with the smallest distance from the student's LS, computed on the basis of the Euclidean distance metric:

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

If such a node does not exist, by means of the function OrderedPredecessorsList, the algorithm computes the list L of the LN predecessors, i.e., the nodes connected to the LN by an incoming link, in order to verify the acquisition of prerequisites, RK, related to the LN. The nodes are accommodated in the list L, according to the following precedence classes: 1) firstly, the predecessor nodes that were not yet visited by the student. In fact it is possible that the student got the AK related to that node, by giving a correct answer to the initial test, but he lacks that concept indeed; 2) then the nodes that do not provide tests, proposed according to the difficulty levels: K, A, E; 3) then the nodes that provide tests whose LS are closest to the student's LS. The AK of the prerequisite nodes, if present, are removed from the CS, because a sort of “loss” of knowledge is present. Then the algorithm puts L on the top of the LOS and suggests its first LN. If both the attempts to explain the concepts with different learning material and the prerequisite checks fail, the algorithm re-plans a new LOS and proposes its first LN.

3.3 Teacher-centered features

The Pdk Planner

Automated planners, in particular the logic based ones, can support either one of the processes of course configuration and domain validation.

A planning problem is described by the initial state, the goal, the executable actions with their preconditions and effects. The solution plan is a set of actions that, if executed in the right order, lead the agent(s) from the initial state to the goal state. In classical planning the following characteristics hold: everything is known: the environment is accessible; the environment is deterministic: effects of actions are known; the environment is static: during the plan generation it does not change. Under the above conditions, goal feasibility is guaranteed for the solution found by a planner. The solution to a planning problem, should it exist, is a complete and consistent plan.

Course personalization problems can easily be seen as planning problems, where the student is the executing agent, the initial world state is the initial

SM (including his SK, i.e. his starting CS, and LS), and actions correspond to LN belonging to the pool. When a student executes an action, he is offered the fruition of a learning material according to his LS. In order to understand such a material, the student may be already aware of other knowledge, the RK, that corresponds to the action preconditions. Once he has terminated the study of the learning material and passed the test (if present), he may be assumed to have gained some additional knowledge, the AK, that represents the action effects. The goal of the plan corresponds to the course target knowledge, TK. In this way, course generation corresponds to synthesizing a sequence of actions leading to the goal.

In Artificial Intelligence planning, the term control knowledge means the additional information that can enrich the planning domain (given as mere list of actions with their preconditions and effects) and guide the plan synthesis. Using control knowledge can be effective from the teacher's point of view for expressing didactic strategies. What it is necessary, is to allow the teacher to specify such kind of control knowledge in an easy way. The Pdk (Planning with Domain Knowledge) planner [MLOP07] conforms to the "planning as satisfiability" paradigm, and the logic used to encode planning problems is the propositional Linear Time Logic (LTL) [Wol85]. The related planning language PDDL-K guides the user into the specification of control knowledge. Pdk accepts PDDL-K as input language, translates the problem description into its LTL representation and reduces planning to model search. The following are the main functionalities provided by Pdk for the learning domain:

- **domain validation**: following the style of [BSS06] Pdk performs pool consistency checks. In particular, considering the pool represented as a graph, where the edges are the propaedeutic relations among LN, it is desirable for the graph to be acyclic: the presence of a loop means in fact that some actions, i.e. some learning materials, can never be executed because their preconditions can never be met. However it is possible that the teacher defines some relations that generate cycles. The loop check is an easy control for the planner: in this case the initial cognitive state of the student is an empty set and the course target knowledge is the set of all the KI. The control will find out that some of the actions can never be executed. In the following some examples of loops and inconsistent actions detection are shown. Let us consider a sample course like the one shown in Fig. 3.4. The teacher can test the domain with empty starting knowledge SK and all KI as goal, in order to check whether all the LN are considered. In this case the result of the planner is shown in Fig. 3.5.

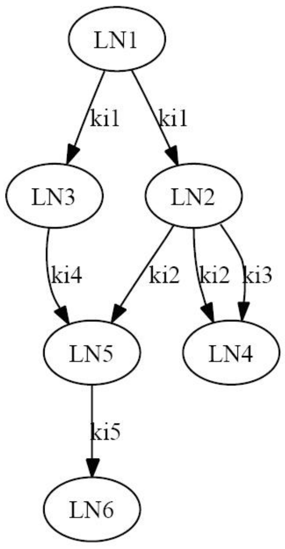
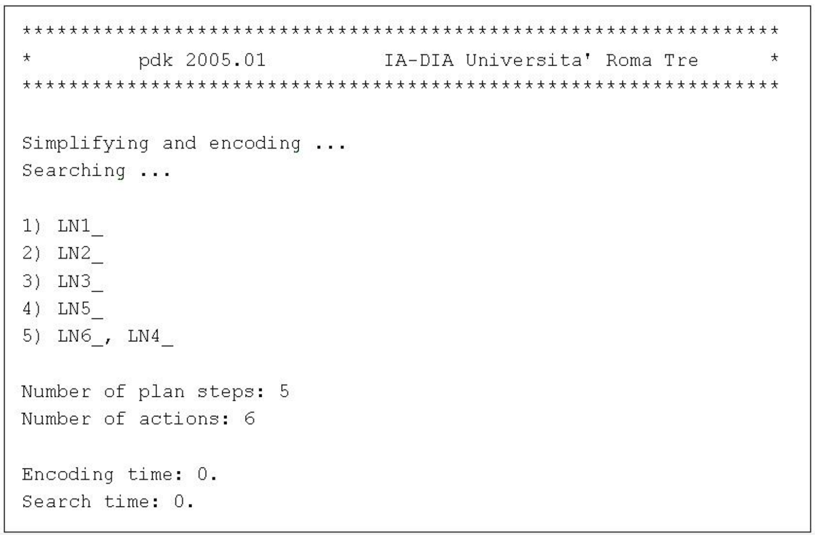


Figure 3.4: The propaedeuticy relations among LN of a sample course.

Since there can be some parallel actions, like $LN6$ and $LN4$, the teacher can decide to prefer the execution of $LN4$ before $LN6$, by adding the $:ASAP$ control schemata (as described in [MLOP07]) to the action $LN4$, as shown in the following:

```
(: action LN4
  : precondition (and (ki2) (ki3))
  : effect (ki7)
  : only − if (not (ki7))
  : asap
)
```

Let us note that the $:ONLY - IF$ control schemata is added to all the actions in order to execute a given action only once: “execute the action only if you do not have already the post-conditions (concepts) acquired”. Let us suppose now, that the teacher inserts in the node the prerequisite $ki6$, that corresponds to the following PDDL-K translation:



```
*****************************************************************
*         pdk 2005.01          IA-DIA Universita' Roma Tre      *
*****************************************************************

Simplifying and encoding ...
Searching ...

1) LN1_
2) LN2_
3) LN3_
4) LN5_
5) LN6_, LN4_

Number of plan steps: 5
Number of actions: 6

Encoding time: 0.
Search time: 0.
```

Figure 3.5: Output of the Pdk planner for the domain of Fig. 3.4 and a student with empty starting knowledge and all KI as goal.

(: *action* $LN3$
  : *precondition* (*and* ($ki1$) ($ki6$))
  : *effect* ($ki4$)
)

The propaedeuticy relations among the LN produce the graph shown in Fig. 3.6, and the checks about action executability will identify the loop as shown in the following:

$$*** \textit{ Action } LN6 \textit{ can never be executed } ***$$

$$*** \textit{ Action } LN5 \textit{ can never be executed } ***$$

$$*** \textit{ Action } LN3 \textit{ can never be executed } ***$$

Although the graph is not consistent, if a path exists, it is found: this is the case when the goal of this course is $ki7$ only. The plan is found anyway and the outcome of the planner is shown in Fig. 3.7.

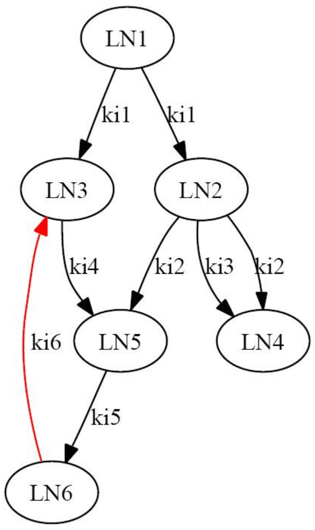
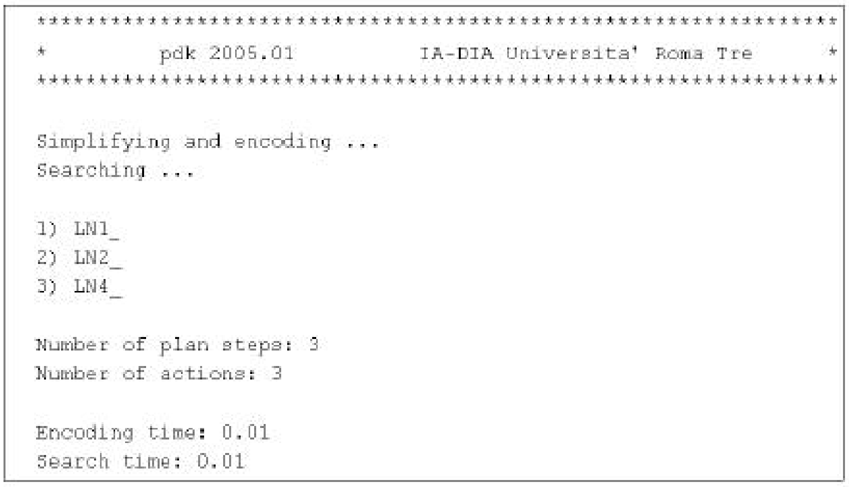


Figure 3.6: Loop example.

- **didactic strategies specification**: providing a set of control schemata Pdk allows the teacher to set some didactic strategies. For instance the teacher can express: the desired level of difficulty of the course; the particular way he prefers to explain a given concept; the constraint about the execution of some actions (LN) the teacher believes are mandatory for all the students, even if they demonstrate they know the concepts related to them; strategies related to learning styles, for example expressing that a sequential student wants to alternate explanations and examples, while a global student prefers to see first all the explanations and then examples; specification of particular constraints, such as a teacher might want to specify that a given LN must be studied before another one also if there is not a prerequisite relationship between them.

  In particular Pdk has been used for including in the system the Bloom's taxonomy cognitive levels [LSTV]. The definition of knowledge items has been extended, by indicating one out of three cognitive levels: K (*knowledge*) standing for the first level, A (*application*) standing for the third level, and E (*evaluation*) standing for the sixth level of the Bloom's taxonomy. The knowledge items can be either generic (such as *recursion*)



```
*******************************************************************
*          pdk 2005.01          IA-DIA Universita' Roma Tre       *
*******************************************************************

Simplifying and encoding ...
Searching ...

1) LN1_
2) LN2_
3) LN4_

Number of plan steps: 3
Number of actions: 3

Encoding time: 0.01
Search time: 0.01
```

Figure 3.7: Pdk finds a plan also if a loop is present.

or labeled (such as *recursion.K*). The teacher is free to define concepts in one of these styles. According to the Bloom's theory, knowledge items at level K are implicitly entailed by the same knowledge item at level A, which in turn, is entailed by the same knowledge item at level E. Two different levels of parameterization can be defined: either the knowledge items of the target knowledge can be specified to be acquired at a given level, or the level of the whole course may be specified. In the second case, each knowledge item occurring in the course and defined at different levels will be acquired at least at the desired level. The predicate *courselevel* allows to specify that the course must be taken at a given level. Fig. 3.8 shows the PDDL-K code that specifies the preconditions, depending on the knowledge level. Let us note that the syntax of the conditional instruction is: *if (predicate) (then-part) (else-part)*. The code specifies that a given action *act*, with prerequisite *c*, will have prerequisite *c.K* if the level K was requested for the course; on the other hand, if the requested level was A (or E), the actual prerequisite will be at least *c.A* (or *c.E*).

The teacher can define contents that are mandatory for all the students

```
(:action act
       :precondition
       .......................
       (if (goal (courseLevel K))
            (C_K)
       (if   (goal (courseLevel A)) (and(C_K)(C _A))
            (and (C_K)   (C_A)  (C _E)))
 )
```

Figure 3.8: Preconditions specification for action *act*, on the basis of the course level. For syntactic reasons in PDDL-K “underscores” substitute “dots”.

(even if they already know them): these contents can be simply canceled, if present, from the student CS in the problem specification, and be added to the goal.

The teacher can also handle different didactic strategies: for example, with reference to the course shown in Fig. 3.9, the teacher could decide to explain recursion either with the *induction principle* (*funzionale*) or *activation records* (*attivazione*) or with both of them. In this case, in the domain specification a predicate that represents the possible approach (*funzionale* and *attivazione* in our example) is defined, and, in the problem, which one has been decided to use is specified. In this case an action such as *Ricorsione:esercizi*, will have alternative prerequisites, as highlighted by the dashed arrows in Fig. 3.9.

- **course sequencing**: course personalization problems can easily be seen as planning problems, enriched with teacher’s didactic strategies as seen before.

Although automated planning is computationally a hard task, the practical execution time depends on many variables, such as the number of pre and post conditions of the actions and the number of goals [Byl94]. Moreover, the definition of correct control knowledge is also a difficult task and can generate inconsistent problems. However, the high-level control formulas defined in PDDL-K provide a set of predefined schemata that allow one to easy and naturally specify heuristic knowledge. An appropriate heuristic knowledge can prune the search space and can improve the performances of the planner, both in terms of execution times and plan quality. From a practical point of view, our experiment presented in [MLOP07] shows that pools with up to 100 nodes can be managed by the planner in less than 5 secs.

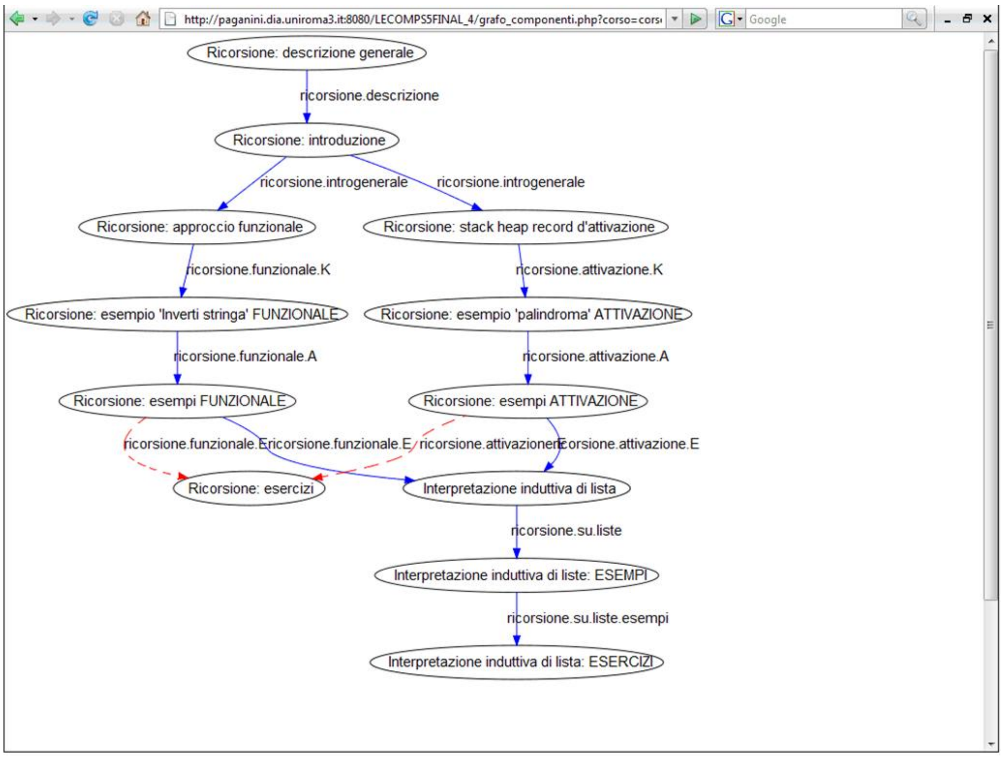

http://paganini.dia.uniroma3.it:8080/LECOMPS5FINAL_4/grafo_componenti.php?corso=cors
Google
Ricorsione: descrizione generale
ricorsione.descrizione
Ricorsione: introduzione
ricorsione.introgenerale
ricorsione.introgenerale
Ricorsione: approccio funzionale
Ricorsione: stack heap record d'attivazione
ricorsione.funzionale.K
ricorsione.attivazione.K
Ricorsione: esempio 'Inverti stringa' FUNZIONALE
Ricorsione: esempio 'palindroma' ATTIVAZIONE
ricorsione.funzionale.A
ricorsione.attivazione.A
Ricorsione: esempi FUNZIONALE
Ricorsione: esempi ATTIVAZIONE
ricorsione.funzionale.E
ricorsione.funzionale.E
ricorsione.attivazione.E
ricorsione.attivazione.E
Ricorsione: esercizi
Interpretazione induttiva di lista
ricorsione.su.liste
Interpretazione induttiva di liste: ESEMPI
ricorsione.su.liste.esempi
Interpretazione induttiva di lista: ESERCIZI

Figure 3.9: A graph of learning materials. Arrows represent prerequisite relationships; dashed arrows stand for alternative preconditions.

The Teacher Assistant

The Teacher Assistant is the part of the system responsible for the management of the functionalities provided for the teacher. It gives support to the teacher for arranging and/or creating the pool of LN, and for validating the pool as he inserts new nodes or defines new propaedeutic relations among them. In particular the teacher decides the student's instructional goal and specifies his didactic strategies, such as the desired level of the course, or the particular way he prefers to explain a given concept. The didactic strategies and the pool validation will be managed by the Pdk planner. The teacher also prepares the Cognitive State Questionnaire and manages the students' registration to the course. Moreover, when the teacher writes the didactic contents of a LN, he indicates the required and acquired knowledge for that node, the minimum and maximum fruition time estimated, and the LS associated to the node.

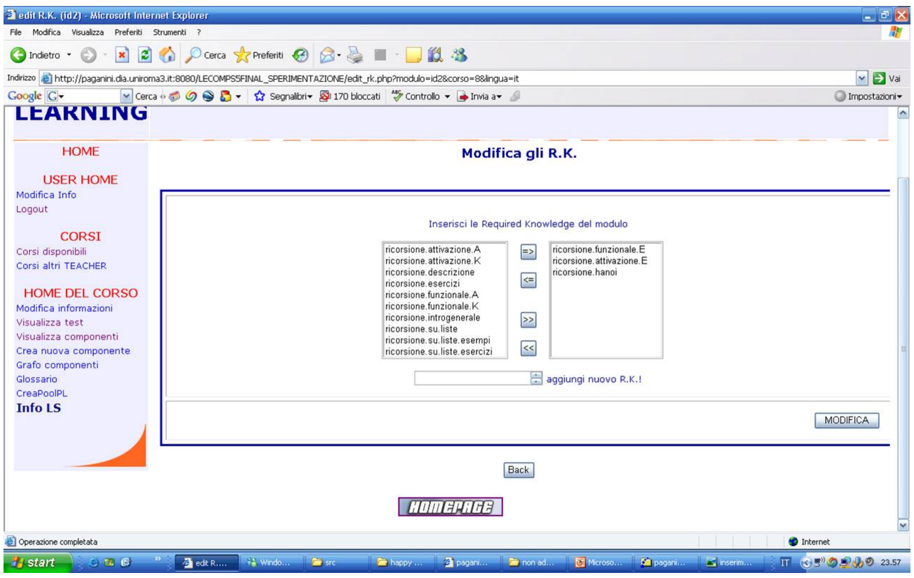
The Teacher Assistant is currently part of the LecompS system, a web application, implemented by using PHP and Java programming languages, developed in collaboration with the Dept. of Computer and Systems Science of the Sapienza University [TV07, ST03, LSTV08b]. The LecompS system is a learning environment supporting teacher's, student's and administrator's functionalities, capable to generate personalized courses on the basis of the student's starting knowledge on the domain of interest, measured by an assessment pre-test, and on learning styles. The approach to course configuration is similar to the LS-Plan one: a personalized course, related to a given topic, is characterized by the Target Knowledge (see definition 8) and by the Starting Knowledge (see definition 9). In this system, knowledge is represented by atomic elements, called Knowledge Items (in the LecompS system Knowledge Items are simpler than in the LS-Plan system: definition 1 can be considered, but levels are not taken into account). A course is composed by a set of Learning Components (LC), i.e., learning objects enriched with the Required Knowledge (see definition 3), with the Acquired Knowledge (see definition 3), with a value for the effort needed on the component by the learner and with different presentations of the learning material, according to the dimensions of Felder and Silverman's learning styles theory. AK is considered acquired, by submitting questions related to the concepts explained in that particular LC. The effort represents the supposed workload for the student to learn that material. All the LC related to a given topic are collected together into a pool. The teacher defines prerequisite relationships among LC. This task is made easy by the graphic visualization of such relationships, in a graph of LC (Fig. 3.9). LecompS configures the personalized course for a given student on the basis of his SK, measured by a pre-test, of his TK, pre-defined by the teacher, and on the basis of prerequisite relations between the RK and AK linked to each LC of the pool. LecompS selects the LC such that the AK of all such selected components, together with the SK, covers as much as possible the TK. The automated configuration of the course is achieved by the Pdk planner, that has been integrated in the LecompS5 version of the system [LSTV08a]. LC editing is performed through the FCK Web editor[2], while the graphical visualization of LC in a pool and in personalized courses is obtained by producing SVG interactive web pages through the Graphviz system [GN00]. The selected

[2]Available at http://www.fckeditor.net/ web site.

LC are divided into lessons, each of them providing an intermediate test, composed by questions belonging to the learning goal of each *LC*. Immediately the system gives feedback about the correctness of the student's answers proposing some links to the *LC* related to the wrong answers. At the end of the course a final test is submitted to the student.

The LS-Plan and the LecompS5 systems are very similar for the initial course configuration, that is carried out in both the systems on using the Pdk planner. Main differences are related to learning styles management and adaptation decision making. The LecompS5 system allows the teacher to define at most four different presentations of the learning material, one for each combination of the visual-verbal and sensing-intuitive Felder-Silverman's dimensions: the teacher can define visual-intuitive, visual-sensing, verbal-intuitive, verbal-sensing presentations of the *LC*. The sequential-global and active-reflective dimensions are used only for giving to the student some suggestions or information [LSTV08b]. The LS-Plan *LS* management is more flexible and finer grained: the teacher can associates weights related to all the Felder-Silverman's dimensions, providing a deeper vision of the materials, having the possibility to express more specific characteristics of the material: for example, the teacher can express that a given *LN* is characterized by a lot of exercises proposes, being useful for an *active* student. Moreover, LecompS5 provides a limited adaptivity, allowing the student to ask for a re-planning of the course, based on his progress.

As mentioned above, however, the LS-Plan and the LecompS5 systems have a very similar approach in modeling the course configuration problem. In particular concepts of KI, RK, and AK are basically the same in the two systems. So, the Teacher Assistant has not been implemented by scratch, but teacher's functionalities are currently provided by the LecompS5 system. From the teachers' point of view the main required effort for configuring a personalized course is based on the tagging of the learning materials, with RK and AK. This task, together with the definition of personalized target knowledge, is supported by the LecompS5 system through suitable interfaces. Fig. 3.10 shows the graphical interface for the RK management, that is very similar to the interface for the AK management; Fig. 3.9 shows the graph produced through the Graphviz system, that provides the teacher with a visualization of the defined prerequisites relations; Fig. 3.11 shows the graphical interface for the course configuration. Some functionalities allowed by the Pdk planner are currently not provided in the Teacher Assistant, such as the consistency check of the pool and the automatic production of the domain.pddl file in the case of domains with approaches and levels. The Teacher Assistant, as part of the



Modifica gli R.K.

Inserisci le Required Knowledge del modulo

ricorsione.attivazione.A
ricorsione.attivazione.K
ricorsione.descrizione
ricorsione.esercizi
ricorsione.funzionale.A
ricorsione.funzionale.K
ricorsione.introgenerale
ricorsione.su.liste
ricorsione.su.liste.esempi
ricorsione.su.liste.esercizi

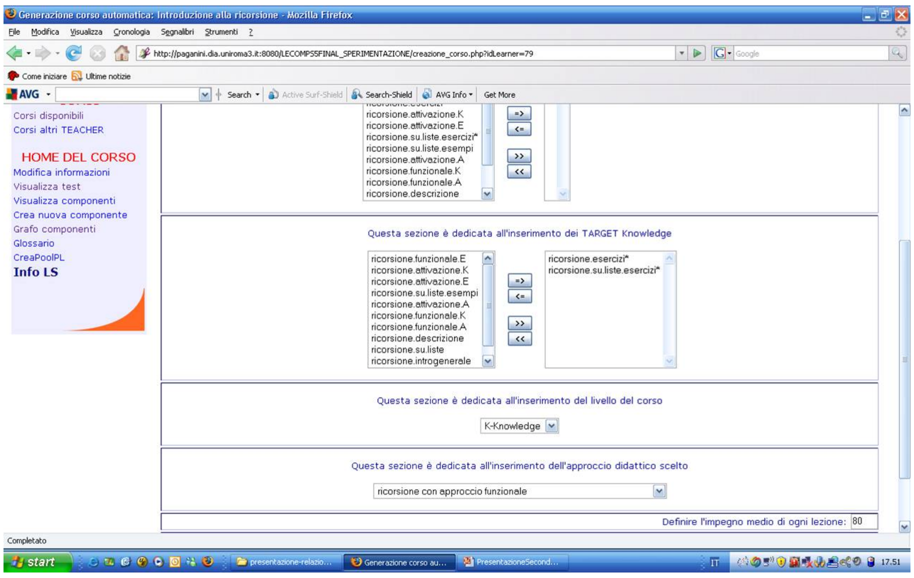
ricorsione.funzionale.E
ricorsione.attivazione.E
ricorsione.hanoi

aggiungi nuovo R.K.!

MODIFICA

Figure 3.10: Graphical interface for the *RK* management.

LECOMPS5 system has been evaluated by users [LSTV, LSTV08a], as shown in Section 4.4, showing encouraging results for the *teacher's satisfaction* aspect of the LS-PLAN system.



http://paganini.dia.uniroma3.it:8080/LECOMPS5FINAL_SPERIMENTAZIONE/creazione_corso.php?idLearner=79

ricorsione.esercizi
ricorsione.attivazione.K
ricorsione.attivazione.E
ricorsione.su.liste.esercizi*
ricorsione.su.liste.esempi
ricorsione.attivazione.A
ricorsione.funzionale.K
ricorsione.funzionale.A
ricorsione.descrizione

Questa sezione è dedicata all'inserimento dei TARGET Knowledge

ricorsione.funzionale.E
ricorsione.attivazione.K
ricorsione.attivazione.E
ricorsione.su.liste.esempi
ricorsione.attivazione.A
ricorsione.funzionale.K
ricorsione.funzionale.A
ricorsione.descrizione
ricorsione.su.liste
ricorsione.introgenerale

ricorsione.esercizi*
ricorsione.su.liste.esercizi*

Questa sezione è dedicata all'inserimento del livello del corso

K-Knowledge

Questa sezione è dedicata all'inserimento dell'approccio didattico scelto

ricorsione con approccio funzionale

Definire l'impegno medio di ogni lezione: 80

Figure 3.11: Graphical interface for the course configuration.

Chapter 4

LS-Plan: empirical evaluation

Several evaluations have been performed in order to assess the system behavior both from the teacher's and from the student's point of view. The performed experiments aimed to provide a complete evaluation of the overall system, focusing on:

- the suitability both of the learning sequences produced by the system and of the adaptation algorithm, given the user model and its evolution;
- the validity of the user model, as estimated by the users;
- the added value of adaptation for students' learning;
- students' and teachers' satisfaction in using the system.

According to the literature about methods for the empirical evaluation of adaptive systems [BKS04, Gen05, Chi01, Mas03], a *layered evaluation* has been performed. Considering the system as decomposed in two main components, i.e. the user modeling and the adaptation decision making component, the layered evaluation is composed by two phases:

- the adaptation decision making evaluation, that aims to answer to questions like *"are the adaptation decisions valid and meaningful, for the given state of the user model?"* [BKS04]
- the user model evaluation, that aims to answer to questions like *"are the user's characteristics being successfully detected by the system and stored in the user model?"* [BKS04]

These two experimental phases, performed for evaluating the LS-Plan system, are described in detail in section 4.2.

An *as a whole* evaluation, based on comparing two students' groups, one using the adaptive features of the system and one not using them, has been also performed in order to verify the added value provided by the system to students' learning experience (see section 4.3).

Teachers have been involved in the performed evaluations, both as experts for an evaluation of the system from a didactic point of view, in the layered evaluation, and as actual users of the system. In particular teachers used the system for inserting RK and AK to LC and for evaluating the sequences produced by the system for two given student models (see section 4.4).

In order to evaluate the LS-Plan system from the students' point of view, it has been integrated in an educational hypermedia, as shown in section 4.1. This integration has allowed to perform the experiments for the *layered* and for the *as a whole* evaluation.

4.1 Integrating LS-Plan in an educational hypermedia

LS-Plan has been integrated in an educational hypermedia; in particular the LecompS system has been used as the web application that enables the delivery of courses, acting as the educational hypermedia [LSTV09a]. The considered domain is related to the *Italian Neorealist Cinema* and is composed by 18 LN, each one having an associated test, and by 12 KI, selected by a domain expert. The experimental environment[1], runs on a Linux departmental server and is based on a java application, for the LS-Plan system, communicating with the php-based LecompS hypermedia. The server is a protected server and a signing procedure is needed. Each student involved in the experimentation enrolled in the educational environment and submitted the questionnaires to input his initial cognitive state in the system, as related to the subject matter, and his learning styles, measured through the Felder and Soloman's ILS test. Once the initial cognitive state and learning styles are available for a learner, it is possible to activate the process of automated configuration of the course via the LS-Plan system. In the present version of the system such process is activated by the teacher through a suitable interface, where the initial cognitive state of the learner and the aimed target knowledge are shown and can possibly be modified. The course is not generated automatically, after the student's submission of questionnaires results, because this process is

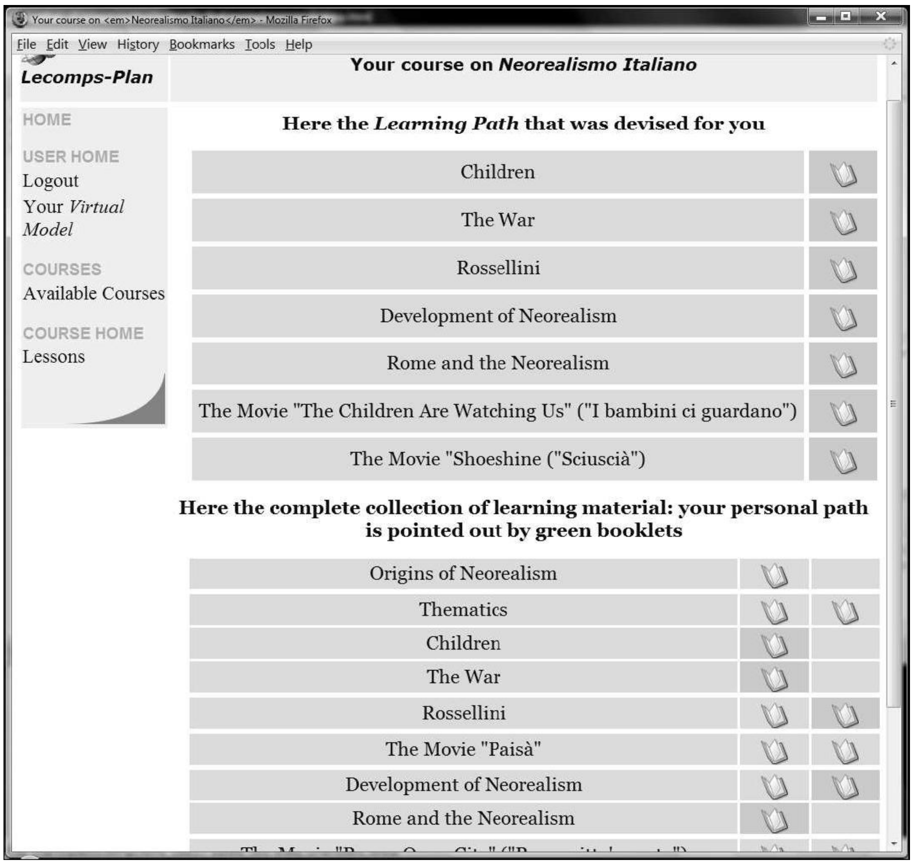
[1]available at http://paganini.dia.uniroma3.it

supervisioned by the teacher, that can personalize the target of the course or can modify the supposed student initial cognitive state, in order to obtain a more extended course for him. When the personalized course is available the learner can access the LecompS web application and take the learning material. The experimental evaluation is based on the use of two different versions of LecompS, one enabling the full application of the LS-Plan framework and another one providing a non-adaptive management of courses. Two examples of access page to a course are shown in Fig. 4.1 and Fig. 4.2. Fig. 4.1 shows the interface used by a learner to take an adaptive course. The upper part of the figure gives the sequence of the learning nodes stated for the learner according with his initial cognitive state and learning style evaluation. This is the actual personalized course, listing all the prescribed learning nodes. On the other hand, the whole set of learning nodes available in the educational environment's pool is available to the learner in the lower part of the page, enabling access to the learning material in a non-prescribed manner too.

The course is taken by selecting one learning node at time (the small books in the figure are links to learning nodes). After each learning node, the learner can take an assessment test, composed by multiple choice and true/false items; on the basis of the answers to the test the student model can be updated and the course can be possibly adapted. Feedback to such update is twofold: as a consequence of modifications in the cognitive state, the learner can see changes in the sequence of learning nodes for his course (only the learning nodes yet to be taken towards course termination are listed in the upper part of the page and an icon representing a closed book is present in the corresponding content of the lower part of the page); as for learning styles and cognitive state modifications, they can be appreciated by accessing a related page, where the learner can see a description of the present state and grade the agreement towards such an evaluation. Fig.4.2 shows the learner's interface for non-adaptive courses: this is basically the list of the learning nodes, with no further treatment by the system.

In order to integrate LS-Plan into the LecompS system a simple wrapper has been implemented for

- calling the necessary methods of LS-Plan when the teacher configures automatically the personalized course for the student and when the student submits his answer to an intermediate test;
- managing the output of LS-Plan, that is presenting the produced sequence to the student, both at the beginning of the course and during the course taking.


Your course on <em>Neorealismo Italiano</em> - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Lecomps-Plan
HOME
USER HOME
Logout
Your *Virtual Model*
COURSES
Available Courses
COURSE HOME
Lessons

Your course on *Neorealismo Italiano*

Here the *Learning Path* that was devised for you

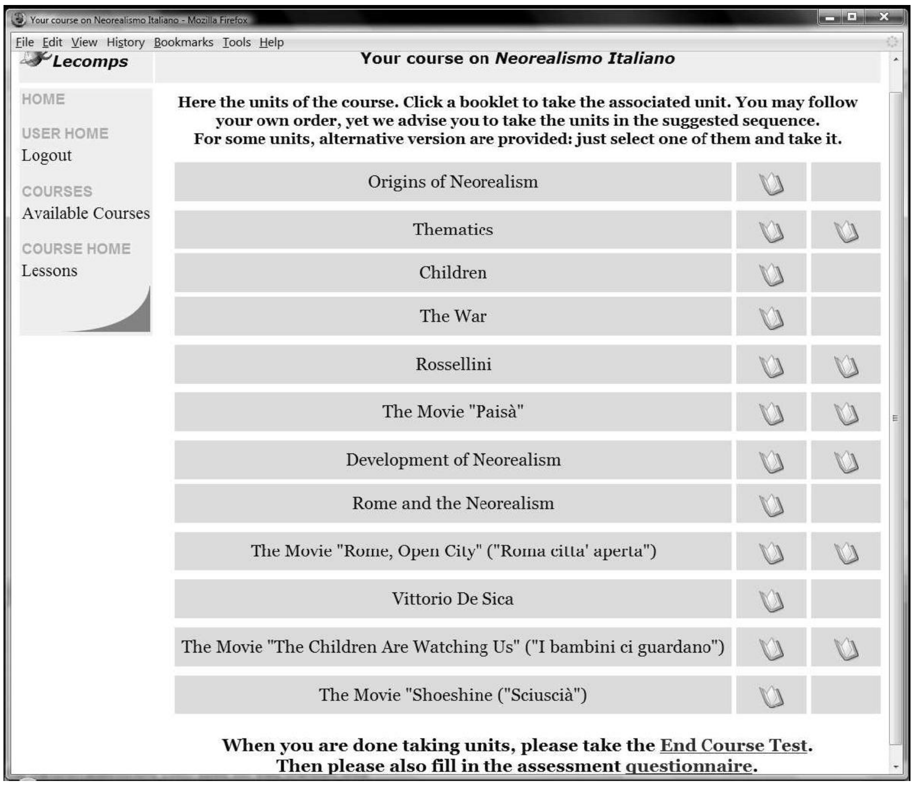
Children	📖
The War	📖
Rossellini	📖
Development of Neorealism	📖
Rome and the Neorealism	📖
The Movie "The Children Are Watching Us" ("I bambini ci guardano")	📖
The Movie "Shoeshine ("Sciuscià")	📖

Here the complete collection of learning material: your personal path is pointed out by green booklets

Origins of Neorealism	📖	
Thematics	📖	📖
Children	📖	
The War	📖	
Rossellini	📖	📖
The Movie "Paisà"	📖	📖
Development of Neorealism	📖	📖
Rome and the Neorealism	📖	

Figure 4.1: Learner’s page: adaptive management.

Integrating LS-Plan into the LecompS web application allowed the evaluation of the system from the students’ point of view. In particular it has been important for having an estimate of the added value given by the adaptation provided by the system to the students’ learning experience (see Section 4.3) and for performing the layered evaluation, i.e. for the user modeling evaluation and, partially, for the adaptation decision making evaluation (see Section 4.2).


Your course on Neorealismo Italiano - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Lecomps

Your course on *Neorealismo Italiano*

HOME

USER HOME
Logout

COURSES
Available Courses

COURSE HOME
Lessons

Here the units of the course. Click a booklet to take the associated unit. You may follow your own order, yet we advise you to take the units in the suggested sequence. For some units, alternative version are provided: just select one of them and take it.

Origins of Neorealism	📖	
Thematics	📖	📖
Children	📖	
The War	📖	
Rossellini	📖	📖
The Movie "Paisà"	📖	📖
Development of Neorealism	📖	📖
Rome and the Neorealism	📖	
The Movie "Rome, Open City" ("Roma citta' aperta")	📖	📖
Vittorio De Sica	📖	
The Movie "The Children Are Watching Us" ("I bambini ci guardano")	📖	📖
The Movie "Shoeshine ("Sciuscià")	📖	

When you are done taking units, please take the End Course Test. Then please also fill in the assessment questionnaire.

Figure 4.2: Learner’s page: non-adaptive management.

4.2 *Layered* evaluation of the system

Adaptation decision making evaluation

In this phase, the goodness of the user modeling component is not considered: the aim is the evaluation of the adaptation decision making process, considering the student model as a suitable one. The evaluation aimed to answer to the research question *RQ*: *Are the adaptation decisions valid and meaningful, for the given state of the student model?*

The early experiment aimed to check if the *LOS* computed by the planner are considered by experts didactically suitable for the SM taken into account [LSV08]. This experiment has been performed starting from a pool of 20 LN related to the *recursion* topic and arranged in such a way that the teacher could decide to explain *recursion* either with the induction principle or with activation records (run time stack management). Moreover the pool allowed that some concepts can be explained with different deepening levels, taken from the Bloom's taxonomy. The experiment considered the two following case studies, in both of them the instructional goal was learning *Recursion*.

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

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

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

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

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

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

The first planned learning path included nodes related to activation records at all the difficulty levels, i.e., K, A and E, to obtain the Evaluation level is necessary to know the previous ones: K and A. The learning path is also suitable for the student because it reflects his LS and starting knowledge: all the contents are proposed because the student has an empty starting knowledge; list recursion is explained in a *verbal* way; suggested complements are proposals for thinking about the learning material, according to the *reflective* student learning preference. Moreover the proposed LOS presents all the theoretical components before proposing exercises. The LOS includes a node that provides an overall picture of the topics of the unit. These can be considered suitable features for *global* learners. The second LOS is also suitable for the student: it does not include the *Recursive_Programs* and the *Rec_Fun_Intro* nodes because the student knows these concepts; it proposes *visual* didactic material; it suggests complements for group working, this is motivating for an *active* learner. Moreover, theoretical material is immediately followed by exercises, because the learner prefers studying in a *sequential* manner. Finally the LOS reflects the teacher didactic strategies, i.e.: *functional* nodes. These two learning sequences were assessed by a sample of 14 teachers who were required to assess the instructional validity of the two proposed didactic plans according to their related SM. To this aim, to our experimental group the following sentence has been submitted both for LOS_1 and LOS_2: *This Learning Objects Sequence is a valid Learning Objects Sequence on the basis of the starting student model SM*, with a 5-points Likert scale (strongly disagree, disagree, neither agree nor disagree, agree, strongly agree). The experimental results have shown that: 7.1% disagree, 7.1% neither agree nor disagree, 71.4% agree and 14.4% strongly agree with the LOS_1; 78.6% agree and 21.4% strongly agree with the LOS_2.

The second experiment aimed to evaluate the suitability of the adaptation algorithm, i.e. of the learning nodes proposed by the system to the student during the course taking [LSV09a]. In particular two case studies have been considered, i.e. two scenarios of student navigation that are the most significative ones in order to verify if the didactic strategies, implemented in the LS-Plan system, are considered suitable by teachers. The case studies have been evaluated by 30 domain experts. The considered learning domain is shown in Fig. 4.3; the related learning nodes, with their prerequisite relationships, are shown in Fig. 4.4, where each LN is provided with a particular KI to be acquired, that is its associated AK, written as a text label over each arrow. LN included in an oval are alternative materials for explaining the same concept.

The two case studies are described in Tab. 4.1 and in Tab. 4.2 respectively. In both the tables the first column contains the name of the node visited by the student, the second column contains the node suggested by the system, computed by the function NextNode, shown in Fig. 3.3. The other columns contain the LS four dimensions: LN is the LS associated to a node and Student is the LS of the student after having visited the node. The ini-

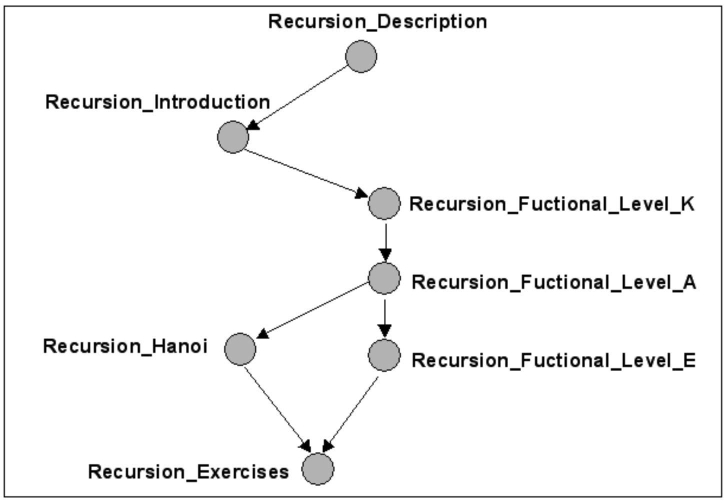

Recursion_Description
Recursion_Introduction
Recursion_Fuctional_Level_K
Recursion_Fuctional_Level_A
Recursion_Hanoi
Recursion_Fuctional_Level_E
Recursion_Exercises

Figure 4.3: The domain model of the case studies.

tial student model has in both the case studies $D_1 = -5.00$, $D_2 = -4.00$, $D_3 = 1$ and $D_4 = 8$. In the first case study the student starts with $SK = \emptyset$, i.e., an empty starting knowledge. The system proposes the following LOS: $LOS =< id3, id10, id15, id16, id13, id7, id2 >$. The student acquires the Recursion_Description KI (id3) but he does not acquire the Recursion_Introduction KI (id10) at first and second time on the same node id10. After the system proposes an alternative learning node explaining the same KI, i.e., Recursion_Introduction, with different LS (id14). The system chooses the only possible node available: if there were other alternative nodes, the system would have proposed the nearest to the student's learning style. The system, like a human teacher should do in a similar scenario, tries to explain to the student the topic in a different manner, selecting a new learning node, alternative to the previous one.

In the second case study, the student starting knowledge SK is: $SK = (Recursion\_Description, Recursion\_Introduction, Recursion\_Functional\_Level\_E)$. The system proposes the following LOS: $LOS =< id15, id16, id13, id2 >$. The student starts the course acquiring first the Recursion_Functional_Level_K KI (id15), after the Recursion_Functional_Level_A (id16) and the Recursion_Hanoi (id13) KI, but fails in acquiring the Recursion_exercises KI (id2), proposed

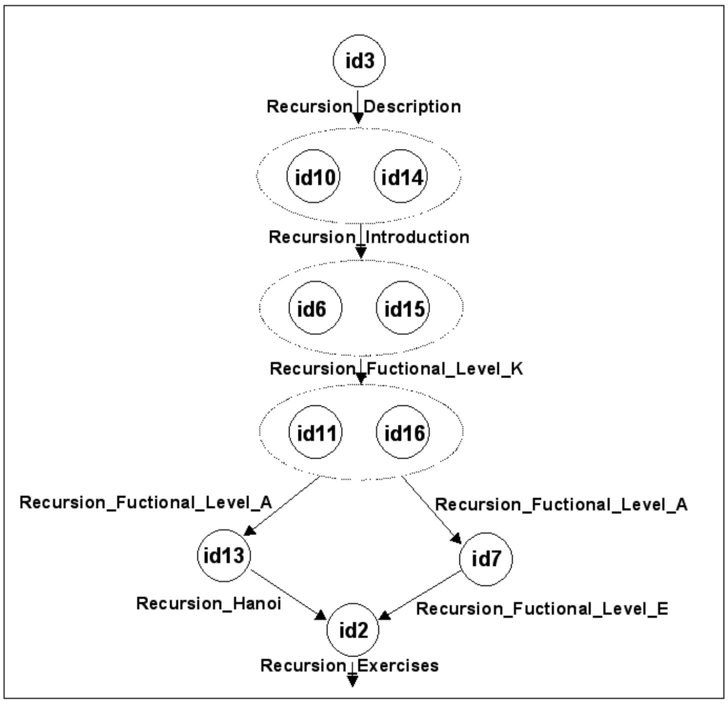


Figure 4.4: The domain learning nodes of the case studies.

as usually, twice. In this case, the Recursion_Hanoi KI (id13) and the Recursion_Functional_Level_E (id7) probably have not been acquired. The system proposes to the student to study again the learning nodes beginning from the Recursion_Functional_Level_E node not yet been visited. This case study is an extension of the previous one: the student shows difficulties in the study of the Recursion_Exercises KI, but no alternative nodes are present in the pool. So the system supposes that it is necessary to check if the prerequisites of the node are really possessed by the student. The system assumes to be in presence of a sort of "loss of knowledge" and recommends to the learner the study of the prerequisites of the node, i.e., the Recursion_Functional_Level_E and the Recursion_Hanoi KI. In particular the node that explains the Recursion_Functional_Level_E KI is proposed as first, because this node has not been visited by the student, belonging Recursion_Functional_Level_E KI to

VN	NN	D_1	D_1	D_2	D_2	D_3	D_3	D_4	D_4
		LN	Student	LN	Student	LN	Student	LN	Student
			-5,00		-4,00		1		8
id3	id10	3	-4,475	4	-3,475	1	1	8	8
id10	id10	-2	-4,85	-1	-3,85	0	1,375	5	8,375
id10	id14	-2	-5,35	-1	-4,35	0	1,875	5	8,875
id14	...								

Table 4.1: First Case Study: the system proposes an alternative LN

VN	NN	D_1	D_1	D_2	D_2	D_3	D_3	D_4	D_4
		LN	Student	LN	Student	LN	Student	LN	Student
			-5,00		-4,00		1		8
id15	id16	2	-4.45	3	-3.45	4	1.55	1	7.45
id16	id13	5	-4.025	-4	-3.875	0	1.125	9	7.875
id13	id2	1	-3.65	5	-3.5	9	1,50	0	7,5
id2	id2	-2	-4.375	4	-4.225	-1	2.225	-1	8.225
id2	loss of knowledge	-2	-5.1	4	-4.95	-1	2,95	-1	8,95

Table 4.2: Second Case Study: the system proposes prerequisites checks.

	strongly disagree	disagree	neither disagree nor agree	agree	strongly agree
N. of Experts	2	3	4	13	8
%	6.7%	10.0%	13.3%	43.3%	26.7%

Table 4.3: First Case Study Evaluation

the student starting knowledge. The new LOS is the following: $LOS =< id7, id13, id2 >$.

The question proposed to the teachers was: "Given the following behavioral student patterns, how do you assess this LOS produced in response by the system?" This assessment was performed through a 5-points Likert scale (completely disagree, disagree, not agree nor disagree, agree, strongly agree) on the two case studies. The overall statistical results are shown in Tab. 4.3 and in Tab. 4.4, where, for each point, the frequency is reported. In both cases,

	strongly disagree	disagree	neither disagree nor agree	agree	strongly agree
N. of Experts	1	4	3	13	9
%	3.0%	13.0%	10.0%	43.0%	31.0%

Table 4.4: Second Case Study Evaluation

more than or equal to 70% of experts gave a positive assessments.

The third experiment for the evaluation of the adaptation decision making aimed to evaluate how much students agreed with the proposed *LOS*. Every time a student leaves a learning node after having taken a post-*LN* questionnaire, in order to measure his knowledge about the knowledge item associated to that particular learning node, the adaptive mechanism builds a new *LOS* on the basis of the algorithm illustrated in Fig. 3.3. Students agreement with the system suggestions has been evaluated integrating the LS-Plan system in the LecompS web application, as illustrated in section 4.1. The system logged all the choices made by the 15 students that used the system in the adaptive modality in order to learn concepts about the *Italian Neorealist Cinema*. Fig. 4.5 shows all the students' choices. The most important result is that 60% of students followed 100% of the suggested *LOS*, while 6.67% followed 88.8% of the suggested *LOS*. More than 85% followed more than 60% of the suggested *LOS*.

User modeling evaluation

This evaluation aimed to answer to the following research question *RQ*: *Are the user characteristics being successfully detected by the system and stored in the user model?* In order to answer *RQ*, during navigation, the system (LS-Plan integrated in the LecompS web application) provided an assessment form, allowing the learner to express agreement or disagreement with his current *virtual model*. In this form the learner is shown the current representation of his student model and is asked to declare his agreement or disagreement through the 7-point Likert scale of Fig. 4.6. The language used to show the student model tries to be non-technical and fully comprehensible for non-insiders. The form was available at any time, while attending the course, through a suitable link included in the navigation menu. Involving students directly in the assessment

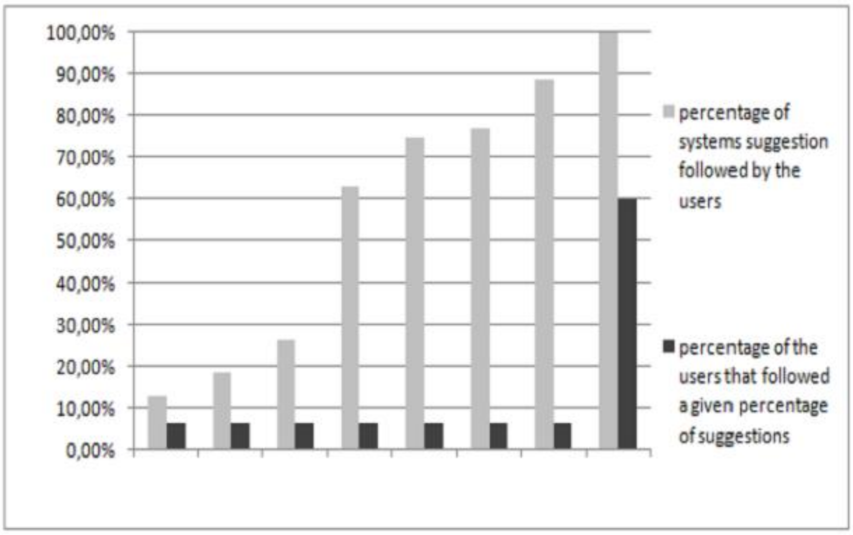


Figure 4.5: Analysis of the suggested *LOS* followed by students.

of their own model is an important issue in order to evaluate the student model reliability [BKS04, Gen05]. The system logged the frequency distribution illustrated in Fig. 4.6. The distribution of the students' rating is unbalanced towards positive values, i.e., the 93% is on the right side of the distribution. Probably most students, who navigated in the adaptive environment, deemed their model "fairly accurate" because of the short amount of time spent surfing and because of the small number of available nodes. In fact, the student model, being a dynamic representation of the student's interests, varies with time and it would have required more time to evolve itself in a more consistent and more suitable way, to converge towards the actual student model.

4.3 *As a whole* evaluation of the system

The *as a whole* evaluation was based on comparing two students' groups, one using the adaptive features of the system and one not using them, in order to verify the added value provided by the system to students' learning experience. The Research Question of this evaluation was therefore *RQ*: *Do students nav-*

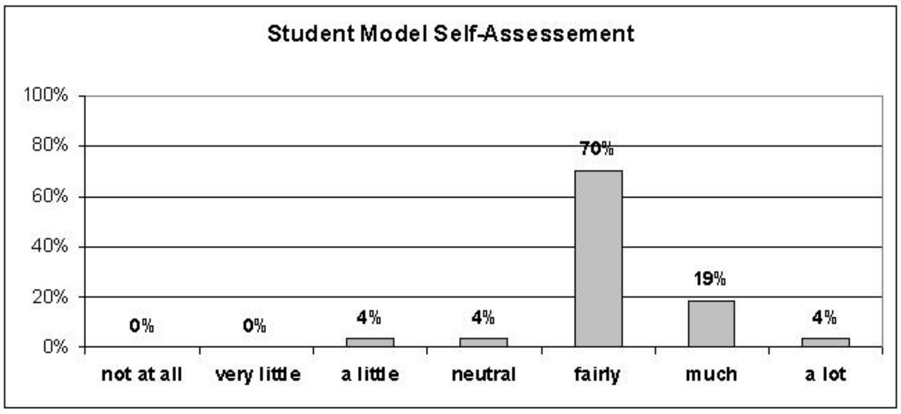


Figure 4.6: The self-assessment frequency distribution. A 7-point Likert scale was used to have a more granular judgment.

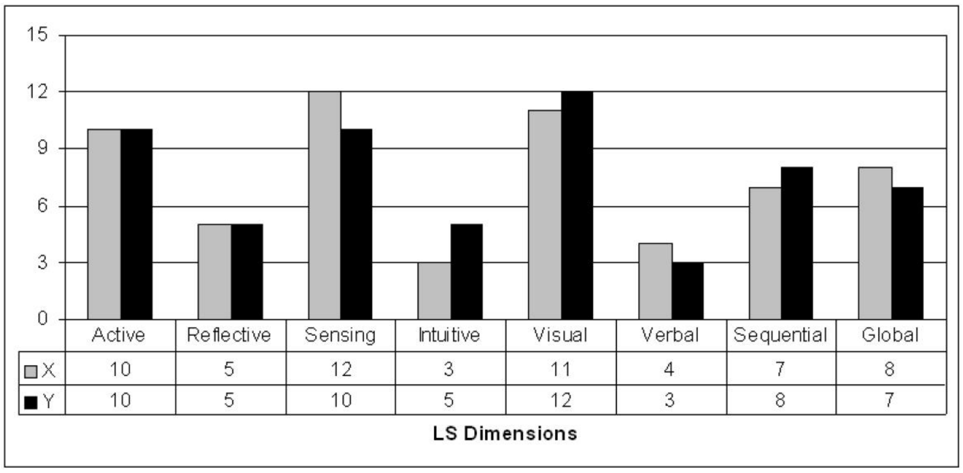
igating with the adaptive modality learn more than students navigating without the adaptive modality?. As mentioned in section 4.1 the *With* modality includes the adaptation provided by the the LS-Plan system to the LecompS web application, the *Without* modality is represented by the LecompS system in which students were free to navigate and to reach their didactic goals without any sort of guidance. Adaptive and non-adaptive GUIs are shown in Fig. 4.1 and Fig. 4.2 respectively. To evaluate the students' knowledge acquisition, a pre-test and a post-test questionnaire were proposed to the students, before and after the course taking. The pre-test aimed to evaluate the student's starting knowledge about the concepts of the domain, i.e. Italian Neorealist Cinema, before the course taking. Both questionnaires were composed by a set of multiple choice and true/false items. The sample was randomly selected among students from Universities, students from high schools, teachers and people who were interested in learning something about Italian Neorealist Cinema. The process of sample-gathering has been divided into several steps. In the first step 45 individuals were selected. In the second step, in order to have a homogeneous starting group (that is a group enjoying the same average a-priori knowledge about the learning domain) the whole group performed a questionnaire containing items about the most important issues addressed by the learning domain. In the third step a homogeneous group of 30 individuals out of the initial 45 with the lowest average starting knowledge on the domain and the lowest possible dispersion around it was formed. The resulting group of individuals had average $\overline{x} = 6.81$ and standard deviation $S_{\overline{x}} = 4.36$. These two values were considered a good compromise between a low starting knowledge and an acceptable dispersion, to minimize the well known statistical problem due to the *between subjects dispersion* [Chi01, Mas03]. In this way, the sample was composed by 30 users, equally distributed between males and females whose age fell within the range $[20, 50]$. On average, the sample started with a domain knowledge of 16.23%, i.e., every individual obtained, on average, the 16.23% of the maximum possible score. The sample has been divided into two groups, the experimental group X, formed by 15 individuals, who navigated with the adaptivity modality, and the control group Y, formed by 15 individuals, who navigated without the adaptive modality. In Fig. 4.7 the sample LS distribution is shown. In particular, the same distribution is present in the Active-Reflective dimension for both experimental group X and control group Y. In the other three dimensions the sample appears almost homogeneous. The analysis of the statistical differences between groups has been performed, by means of the non-parametric two-tails U-Test [SC88] with its associated power analysis, as suggested in [Chi01]:

- Null Hypothesis H_0: there is no difference between the experimental group X and the control group Y: the two groups began with the same starting knowledge on Italian Neorealist Cinema.
- Alternative Hypothesis H_1: the two groups X and Y are different in terms of starting knowledge on Italian Neorealist Cinema.
- Significance Level $\alpha = 0.05$

The resulting $p-value$ is 0.25 and power=0.732. H_0 can be accepted strengthening the non-difference hypothesis between X and Y while the power value is close to the value suggested in [Chi01], that is $power = 0.8$. This value has been accepted as a good compromise between the number of participants and the probability of 73% to reject H_0 when false.

In order to answer the research question, the hypothesis-testing technique has been used, with the following working assumptions:

- *Independent Variable.* The independent variable Δ_S represents the difference between the score S_{post} obtained by each student in the post-test and the score S_{pre} obtained in the pre-test: $\Delta_S = S_{post} - S_{pre}$. This independent variable allows to measure the real improvement shown by the student after his learning.



	Active	Reflective	Sensing	Intuitive	Visual	Verbal	Sequential	Global
▨ X	10	5	12	3	11	4	7	8
■ Y	10	5	10	5	12	3	8	7

Figure 4.7: The sample distribution of the *LS* dimensions.

- *Use of the Distribution-Free statistics.* The statistical distribution, which the independent variable Δ_S belongs to, is not supposed to be the normal distribution (e.g. [SC88, HWC73]). This assumption strengthens the experiment because it follows a more general statistical approach.

- *Use of the Test of Wilcoxon-Mann-Whitney for two independent samples.* This test, which proceeds from the psychological research area, is well suited for testing in experiments where humans play a crucial role [Wil47] and where one has to verify a simple shift towards higher values of the median θ of a stochastic independent variable. Besides, this test is a powerful one and corresponds to the parametric *t-test.* Δ_X and Δ_Y are the values of the variable Δ_S respectively for the group X and Y.

Students of both groups were required to navigate into the system for 45 min. at the most. All the pre-test scores S_{pre} and all the post-test scores S_{post} has been used for computing the variable Δ_S. Tab. 4.5 shows the main statistical parameters. On average, students who navigated into the adaptive environment, showed a better improvement than the students who navigated into the non-adaptive environment. Moreover, the Standard Deviation S_{Δ_S} is the same for both groups.

Modality	**St. Dev.** S_{Δ_S}	**Mean** $\overline{\Delta_S}$
Adaptive	4.06	10.30
Non-Adaptive	4.29	7.61

Table 4.5: Statistical data gathered for the independent variable Δ_S.

Applying the Wilcoxon-Mann-Whitney testing procedure, the Null Hypothesis H_0 and the Alternative Hypothesis H_1 has been defined, and the *significance level* α has been fixed:

- Null Hypothesis H_0: the two learning modalities show a non significant statistical difference between them; the variables Δ_X and Δ_Y belong to the same statistical distribution.

- Alternative Hypothesis H_1: the two learning modalities show a significant statistical difference between them; the variables Δ_X and Δ_Y belong to different statistical distributions, and $\theta_{\Delta_Y} < \theta_{\Delta_X}$, being θ_{Δ_Y} and θ_{Δ_X} the medians of the two statistical distributions respectively of Δ_Y and Δ_X. This would mean that the statistical distribution of the adaptive modality is shifted towards higher values of acquired knowledge.

- Significance level $\alpha = 0.05$.

Following the standard Wilcoxon-Mann-Whitney procedure, the resulting $p-value$ is $0.03 < \alpha$. Therefore the Null Hypothesis H_0 can be rejected and the alternative Hypothesis H_1 can be accepted.

Therefore the evaluation showed that the two independent variables Δ_X and Δ_Y belong to two different statistical populations. As a result, the student who navigated with the adaptivity modality presents, on average, an improvement in the domain knowledge of about $\Delta = 26\%$, being $\Delta = \overline{\Delta_X} - \overline{\Delta_Y}$, expressed in percentage with respect to $\overline{\Delta_Y}$. Moreover, applying the Hodges and Lehemann procedure [HL56, HL63], the estimator $\widehat{\Delta}$ of the $\overline{\Delta_S}$ variable has been computed: $\widehat{\Delta} = 2.6$. In other terms, students who used the system in the adaptive modality experienced an improvement in learning of about 27.54% as opposed to students who navigated in a *Without* modality. Finally, the U-Test with its associated power analysis strengthen the replaceability of these results.

4.4 Teachers’ evaluation of the system

In order to measure the efficiency and effectiveness of the Teacher Assistant, i.e. of the teacher’s LECOMPS5 system functionalities, from the teacher’s point of view, the experimentation presented in [LSTV, LSTV08a] has been performed. The Research Questions of this experimentation have been:

- RQ1: Does the LECOMPS5 system help the teacher to prepare his didactic plans, in terms of time spent and of generated didactic plan quality?

- RQ2: Is the LECOMPS5 system somehow able to generate reasonable didactic plans, i.e., didactic plans that are judged valid didactic plans by teachers?

- RQ3: Is the teacher satisfied with the use of the LECOMPS5 system, in terms of general usefulness, to complete his task?

For the experiments twenty teachers have been randomly selected, from graduated and undergraduate schools of Computer Science. The used domain is related to *Recursion* and is composed by a pool, consisting of twelve learning components. For example, two learning components were *Ricorsione:introduzione (Recursion:introduction)* and *Ricorsione:esercizi (Recursion:exercises).* Two different tasks were submitted to the sample. In the first task, hand-written, teachers were required to write down two distinct learning sequences, i.e., two distinct learning paths formed only by learning components belonging to the pool for two different learner profiles. To this aim, a questionnaire, suited to measure the student’s starting knowledge of the *Recursion* domain has been prepared; two different sets of answers for two different learner profiles have been simulated and submitted to the teachers in order to receive two hand-written learning paths, one for each profile. In the second task, teachers were required to actually use the LECOMPS5 system to build a learning sequence reaching the same learning goal as in the first hand-written task. Teachers were firstly invited to complete the specification of the learning components in the pool, by stating the RK and the AK for each one of them. The system was then used to automatically configure two learning sequences, one for each of the above-mentioned learner’s profiles (i.e., the representation of the two SK on the domain of interest).

For every teacher, given two different learner profiles, expressed by the two simulated questionnaires, the following data have been collected:

- two hand-written learning sequences produced by the teacher for each student's profile;
- a questionnaire with the teacher's assessment on the comparison between his hand-written learning sequences and the learning sequences produced by the system for the same learner profiles;
- one happy sheet concerning the teacher's degree of satisfaction in using the system.

The proposed learner profiles are the following:

- Student A: did not answer correctly to any question of the questionnaire. i.e., this kind of student does not know anything about *Recursion.*
- Student B: answered correctly to a subset of the questionnaire showing a partial starting knowledge of the domain.

In both cases, the TK should cover all topics of the course, with the possibility, however, to choose the *functional*, and/or the *activation record* approach. In the first task, each teacher drew a sequence of learning objects for both profiles A and B.

In the second task, every teacher was asked to insert relations among the LC of the pool, i.e. the RK and AK for all the LC, by means of the system GUI described in Fig. 3.10. In this way, every teacher generated his own set of didactic relationships among the learning components, available as a graph, as shown in Fig. 3.9. The system, starting from the prerequisite relationships among the Learning Components of the pool built by every teacher, automatically produced two sequences, on the basis of both profiles A and B. The teachers were then asked to answer to the following questions:

- $Q1$: How similar were the courses generated by the system, for both profiles A and B, to your hand-written ones?
- $Q2$: How reasonable were the sequences generated by the system with respect to the inserted prerequisite relationships?

The first question aimed to assess the closeness between the course generated by the system and the one generated by hand by the teacher. The second question aimed to assess the didactic validity of the plans built by the system, even if not exactly equal to the ones proposed by the teachers.

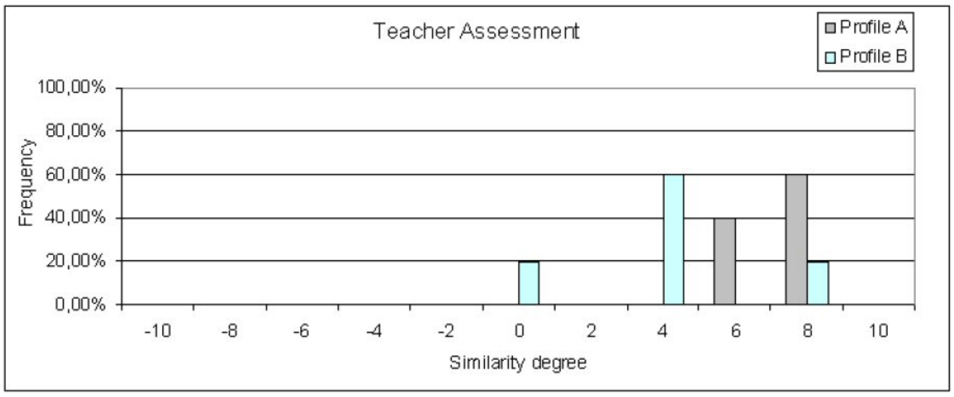


Figure 4.8: Teacher assessment of similarity degree.

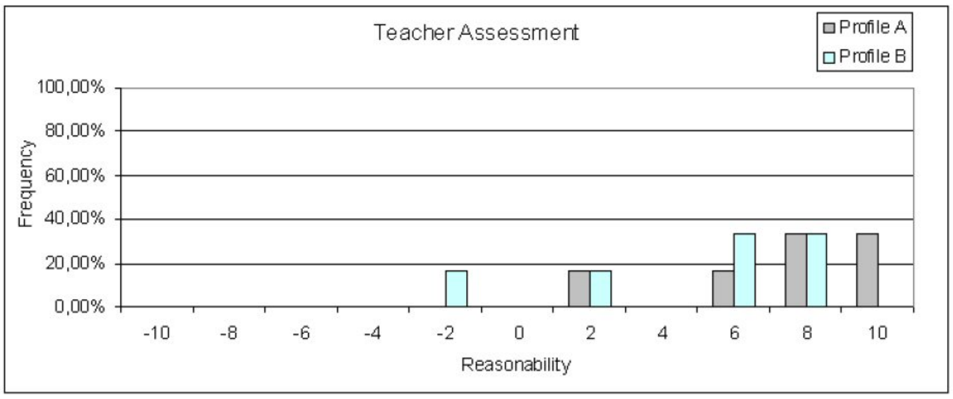


Figure 4.9: Teacher assessment of the reasonability of the course produced by the system.

Fig. 4.8 shows the experimental results, concerning the teacher assessments for both profiles A and B about the closeness between the course generated by the system and the one they have generated by hand. In this figure, the x-axis is an ordinal scale ranging from -10 to +10 while, the y-axis, for every value of the ordinal scale, shows the frequency of teachers that choose that value. For the profile A (i.e., a student with an empty starting knowledge about the domain) the teachers assessed a good similarity degree between their handwritten course, compared to the one produced by the system. In fact, 40% of the sample, namely, eight teachers, gave “6” as similarity degree, which means the two courses were considered similar enough. The remaining 60% of the sample, twelve teachers, assessed “8” as similarity degree, which is a very high value of similarity degree. In both cases, for this profile, the teachers judged as positive the courses produced by the system. For the profile B, i.e., a student with a poor starting knowledge, 20% of the teachers (4 teachers), evaluated the work of the system very similar to their handwritten work, rating it “8”. Four teachers, that is the 20% of the sample, rated it “0”, thus assessing a neutral position with respect to the course produced by the system. Finally, twelve teachers, 60%, rated it “4”, assessing a sufficient degree of similarity. In conclusion, both the experimental frequency distributions are entirely contained in the right part of the “0” point and consequently for both profiles the system performed well, as assessed by the sample.

Fig. 4.9 shows the distribution frequency of the answers to question $Q2$, with the same ordinal scale of the previous case, for both A and B student profiles. This question aims to measure the didactic validity of the course produced by the system, even if different from the one produced by the teacher. In fact, the course could still be a reasonable course, even if different from the one proposed by the teacher. The frequency distribution for the profile A is shifted towards the right part of the “0” point, thus indicating the reasonability of the courses produced by the system. In fact, the 33,3% of the teachers gave “8” and “10”, hence assessing a very good capability of the system in producing didactically valid courses. Values “2” and “6” were rated by the 16.7% of the sample. For the profile B, the frequency distribution was less positive, however, 84% of the values were in the right part with respect to the “0” point.

Teachers were also asked to fill in a questionnaire, a Happy Sheet, for expressing their degree of satisfaction in using the system, together with information concerning the time spent to complete the task. The aim of this questionnaire was that of gathering some indications about usability as well. The Happy Sheet analysis showed some important results, for example the question “How useful do you consider the LecompS5 system in the construction of a course?” was answered, by means of a five-points Likert scale, as shown in Fig. 4.10: more than 70% of experts answered either “useful” or “very useful”.

Fig. 4.11 shows the time spent by the teachers to insert RK/AK for every learning object of the pool. 65% of the teachers spent no more than 60 minutes to build the pool didactic relationships among learning objects. This first experimentation gave positive feedback on the usefulness of the system. Indeed,

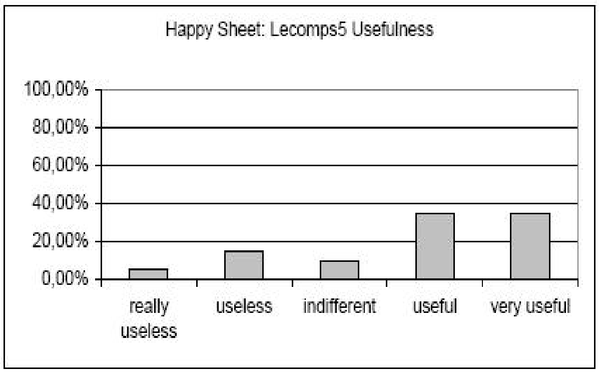


Figure 4.10: LecompS5 Usefulness.

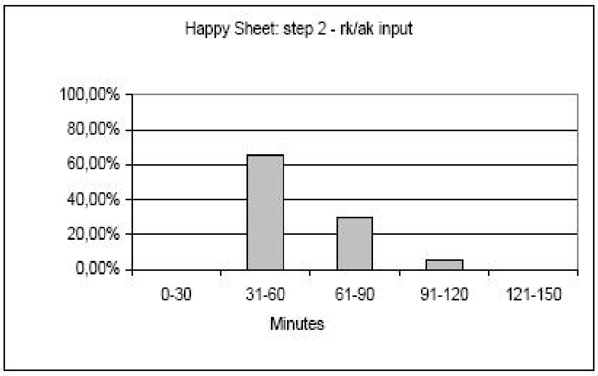


Figure 4.11: The RK/AK input process.

the most important result was the fact that the system produced courses similar to those produced by the teacher, starting from the same student profile and from the same learning components, thus showing a high degree of didactic reliability.

Chapter 5

Analogies between personalization of courses and cultural visits

Visiting a real or virtual museum or an archaeological site can be a hard task, especially in case of large sites provided with many works of art or ancient ruins. For this reason most historical sites provide guided tours, to improve visitors' satisfaction and interest. For providing attractive cultural tours, personalization is more and more considered, with the aim of guiding visitors during cultural sites tours, according to their own interests, background, and needs.

Personalized learning and personalized tours are not so far apart, many analogies can be found between these two problems: in both cases, people are requested to surf in a knowledge domain, composed either of virtual Learning Objects, as it happens for distance learning, or of virtual or actual works of art, as it happens when visiting a museum; in both cases a user model is necessary that takes into account knowledge already possessed, before the course/tour; in both cases it is desirable to build a path that guides the user in acquiring the desired knowledge, according to personal characteristics. These analogies bring to consider the possibility that personalized tours in Cultural Heritage domains can be carried out by exploiting methods and techniques developed for personalized e-learning courses. In particular, a tour in a museum or in an archaeological site, either fully virtual or blended, i.e. partially virtual and partially *on-site*, may be managed through an e-learning environment, where artworks are part of the concepts taught through Learning Objects, and a personal learning path is a sequence of Learning Objects representing a tour in a cultural site.

The Lecomps5 adaptive e-learning system, presented in section 3.3, has been used in [LSTV09b] for providing museums or other cultural sites with the capability of automatically planning personalized tours, according to visitors' needs and interests. Lecomps5 allows a domain expert, through a suitable GUI, to build a pool of learning components concerning a given site. Then the system, by means of the Pdk planner, generates a personalized tour through the works of art, on the basis of the visitor's artistic interests and needs, inferred through an initial questionnaire. At the moment adaptivity during the visit is not provided by the system.

After a brief overview of systems proposed in the literature for personalization in cultural domains, analogies between personalization of courses and tours are detailed in Section 5.2; Section 5.3 describes how the Lecomps5 system has been adapted for personalizing didactic cultural tours; finally Section 5.4 shows a first application of the Lecomps5 system for the personalization of an ancient archaeological site called *Lucus Feroniae*.

5.1 Related work

A lot of systems have been developed in recent years, focusing on user modeling and adaptive techniques for supporting visitors and for increasing their interest and motivation in cultural tours [PN05, SZB$^+$08, AGP$^+$02] and on frameworks for supporting the domain expert in contents creation and sequencing [PN05, NCMW08]. Relevance of personalization in cultural visits has been investigated in [PN05, GBGPZ06].

Petrelli and Not [PN05] conducted a survey on museum visitors to find out relevant user's characteristics and related useful adaptive features. Their study has revealed that beyond interests and knowledge, it is important to categorize users in stereotypes including first time visitors/frequent visitors, families/schools/adults, interaction preferences with the system (automatic/interactive). Moreover, in [GBGPZ06] the influence of personality factors on acceptance of adaptivity has been studied.

Research in cultural domains focuses both on virtual applications [NCMW08, WASR07] and on actual mobile applications. In [BCK05] a survey of map-based mobile guides is presented, not providing only cultural services. In [RTA05] a review of mobile guides for museums is proposed. Some other relevant works in this domain are: GUIDE [CDMF00] and INTRIGUE [AGP$^+$02], which are mobile guides systems for the cities of Lancaster and Turin respectively; HyperAudio [PN05], PEACH [SZB$^+$08], and CHIP [WASR07], that are adaptive mobile guides for museums.

GUIDE is one of the earlier context-aware system: it takes into account both *environmental* context, i.e., attractions opening times, etc., and *personal* context, i.e., actual user's position, interests, already visited places, preferences. GUIDE provides context-aware information, for instance information about the nearest monuments and attractions, and personalized tours of Lancaster city.

INTRIGUE is a similar system, developed for personalized tours in Turin, that, differently from GUIDE, uses Java-enabled mobile phones, instead of dedicated devices.

HyperAudio is an adaptive mobile guide for museum visits: it provides, on a PDA with an essential graphical interface, audio and hypertext, personalized on the basis of adaptation rules. Adaptation is based on the context and on the user model that includes interests, knowledge and stereotypes, based on first time visitors/frequent visitors, families/schools/adults, interaction preferences with the system (automatic/interactive). Adaptation is realized through an automatic selection and processing of pages fragments.

PEACH is an adaptive guide for museum visits, and, besides personalization of audio and hypertext, provides also personalization of video and of visit reports. It uses stationary devices together with PDA. User modeling is obtained through observations of the user's behavior and through explicit feedback; the user is not required to fill in questionnaires.

CHIP is a museum website able to perform an interactive adaptive dialog with the user in order to externalize visitors' preferences and interests for providing personalized access to the museum collections.

The domain expert's point of view is investigated in some systems [WASR07, PN05, NCMW08]. CHIP proposes an ontology-based approach for bridging the gap between domain experts and users. Hyperaudio provides an environment for a modular development and testing of the adaptive system; this environment has been also used for supporting the contents creation. In the Eiffel project an ontology-based approach for describing the tourism domain is proposed, and in [NCMW08] is proposed an enrichment by means of a semi-automatic mechanism for supporting domain experts in annotation tasks.

The main novelty introduced by Lecomps5 for personalization of cultural tours concerns the didactic aspect of such system. This approach considers a cultural tour as a *didactic* cultural tour, in which places to be visited and cultural contents are sequenced by following prerequisite relationships. Personalization is obtained by taking into account user's knowledge, places already visited, interests and domain expert's conceptual definition of the domain. From the domain expert's point of view Lecomps5 provides an easy tool, as shown and evaluated in [LSTV]. Using Lecomps5 for cultural visits personalization is based on the analogies between personalization of courses and personalization of visits, as shown in Section 5.2. How these analogies have been used for adapting the Lecomps5 system for cultural visits personalization is then shown in Section 5.3

5.2 Analogies between course personalization and cultural tours personalization

It is possible to detect some analogies and differences between personalization of cultural tours and personalization of courses. A cultural visit can be: virtual, *on-site*, and blended, that is partially virtual and partially *on-site*. Constraints for producing suitable tours are topological, i.e. physical constraints among places to be visited, and logical, i.e. *prerequisite* relationships that hold between places to be visited, works of art, or their explanations. As regards *on-site* visits topological constraints are very important and, probably, more important than logical constraints. On the contrary, virtual or blended visits can focus on logical constraints, that is producing a *didactic* tour, that helps the visitor in a *learning* cultural experience. The approach proposed in [LSTV09b] is currently applied to virtual or blended visits, and the topological problem is not yet addressed. For this kind of visits the main constraints for producing a suitable tour are logical, the information material to be delivered through a visit is assimilable to learning objects, and a personalized visit is a sequence of such materials that fulfills prerequisite relations among them. So, in the Lecomps5 system a visit is a sequence of *LC*, automatically selected on the basis of visitor's interests and needs. In particular analogies between personalization of courses and personalization of cultural tours are shown in Fig. 5.1, in which concepts, presented in Section 3.3, used by the Lecomps5 system for producing personalized courses, are mapped to concepts of the cultural heritage domain for producing personalized tours. As regards differences between student model and visitor model it is possible to observe that:

- learning styles are more important for the student model than for the visitor model, however thay can also be considered for the visitor model, if more than one *LC* is present for explaining a given concept;

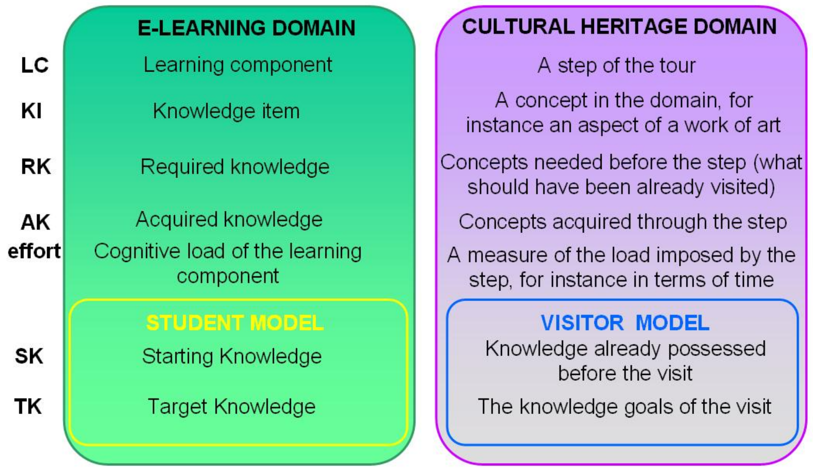

E-LEARNING DOMAIN
CULTURAL HERITAGE DOMAIN
LC
Learning component
A step of the tour
KI
Knowledge item
A concept in the domain, for instance an aspect of a work of art
RK
Required knowledge
Concepts needed before the step (what should have been already visited)
AK
Acquired knowledge
Concepts acquired through the step
effort
Cognitive load of the learning component
A measure of the load imposed by the step, for instance in terms of time
STUDENT MODEL
VISITOR MODEL
SK
Starting Knowledge
Knowledge already possessed before the visit
TK
Target Knowledge
The knowledge goals of the visit

Figure 5.1: Analogies between personalization of courses and personalization of cultural tours.

- modeling user’s interests is very important for the visitor model, in fact, especially in case of large sites provided with many works of art or ancient ruins, the visitor can be interested only in some aspects of the available steps.

Moreover visitors are not evaluated during their tours, while students are evaluated during their learning process. Given the above-mentioned similarities and differences between personalization of courses and personalization of cultural tours, the LECOMPS5 system has been adapted for producing personalized cultural tours taking into account the need of modeling visitor’s interests and the need of using an as-light-as-possible, or absent, Starting Knowledge Questionnaire, for modeling places already visited that the user does not want to visit again or knowledge already possessed. Section 5.3 illustrates how the LECOMPS5 system has been adapted according to the above-mentioned considerations.

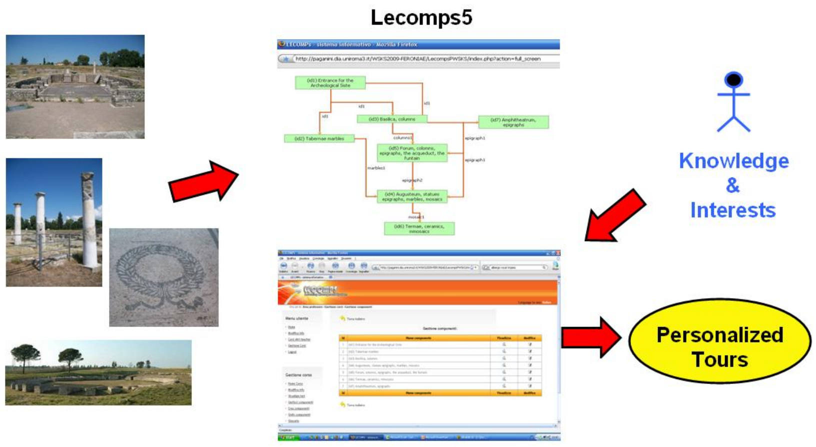


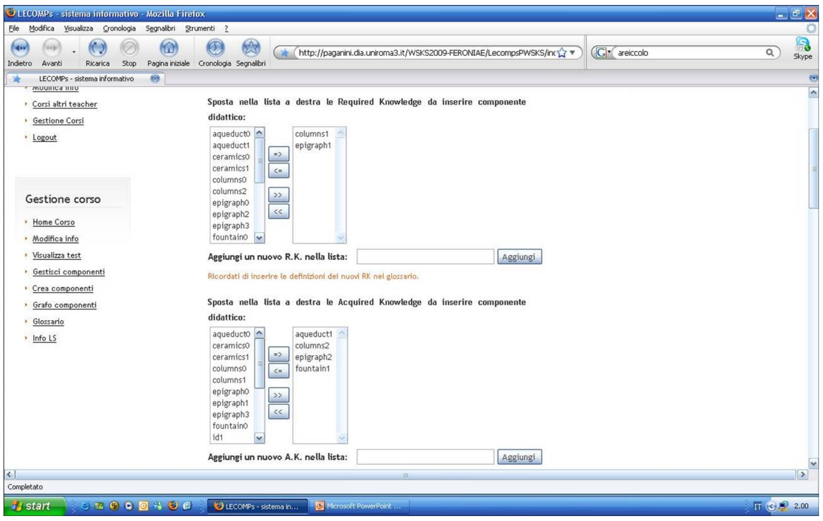
Figure 5.2: The LECOMPS5 system for personalizing cultural tours.

5.3 Adapting an e-learning system for personalizing didactic cultural tours

Personalization of cultural tours using the LECOMPS5 system consists of the following steps, shown in Fig. 5.2:

1. information about the archaeological site, or about the museum, or about monuments of the given city, in other words about the cultural domain, are mapped in a repository of learning resources, i.e. LC, of the LECOMPS5 system. The domain expert edits the contents supported by the editing service offered by LECOMPS5, giving, if he wants, different versions of the contents according to the possible combinations of sensing-intuitive and visual-verbal Felder and Silverman's dimensions, i.e. *sensing-visual*, *sensing-verbal*, *intuitive-visual*, *intuitive-verbal*. Moreover he defines prerequisite relations among LC, defining their RK and AK. The domain expert can also define interests, i.e. grouping related KI, as it will be explained in the following.

2. the visitor express his SK, that is the knowledge he already possesses or places already visited that he does not want to visit again. Moreover he express his interests.


LECOMPs - sistema informativo - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti ?
Indietro Avanti Ricarica Stop Pagina iniziale Cronologia Segnalibri
http://paganini.dia.uniroma3.it/WSKS2009-FERONIAE/LecompsPWSKS/inc
areiccolo
Skype
LECOMPs - sistema informativo
Corsi altri teacher
Gestione Corsi
Logout
Gestione corso
Home Corso
Modifica info
Visualizza test
Gestisci componenti
Crea componenti
Grafo componenti
Glossario
Info LS
=>
<=
>>
<<
Aggiungi
Completato
start
LECOMPs - sistema in...
Microsoft PowerPoint ...
IT
2.00

Sposta nella lista a destra le Required Knowledge da inserire componente didattico:

aqueduct0, aqueduct1, ceramics0, ceramics1, columns0, columns2, epigraph0, epigraph2, epigraph3, fountain0

columns1, epigraph1

Aggiungi un nuovo R.K. nella lista:

Ricordati di inserire le definizioni dei nuovi RK nel glossario.

Sposta nella lista a destra le Acquired Knowledge da inserire componente didattico:

aqueduct0, ceramics0, ceramics1, columns0, columns1, epigraph0, epigraph1, epigraph3, fountain0, id1

aqueduct1, columns2, epigraph2, fountain1

Aggiungi un nuovo A.K. nella lista:

Figure 5.3: GUI for RK and AK definition.

3. the LECOMPS5 system, given the domain model, i.e. the LC and their relationships, and the visitor model builds the personalized cultural visit.

Regarding the first step, the teacher is supported in RK and AK definition by the GUI shown in Fig. 5.3, that is a “restyling” of the GUI shown in Fig. 3.10. The resulting graph of prerequisite relations has been already shown in Fig. 5.2: the GUI shows LC, i.e. named boxes, that give access to a pop-up menu, for editing and visualizing facilities for the component. Labels of the arrows represent prerequisite relationships among LC, that is the RK of the LC at the end of the arrow. The domain expert defines also interests. For a visiting user, an *interest* is the expression of the fact that he would like to touch certain topics during the visit to be planned. In LECOMPS5, such interest is expressed as a set of KI of the domain D. An APPROACH_OF_INTEREST, A_I, is basically the association of an identifier, with a subset of D, listing all KI in the domain that are related to the interest and so should likely be met during the visit to be planned. Several approaches, i.e. several interests can be defined by the visit manager, using the GUI shown in Fig. 5.4. The menu on the left shows all the available KI; the menu on the right the KI selected for the approach. Regarding the second step, it is aimed to infer user’s characteristics, that is SK

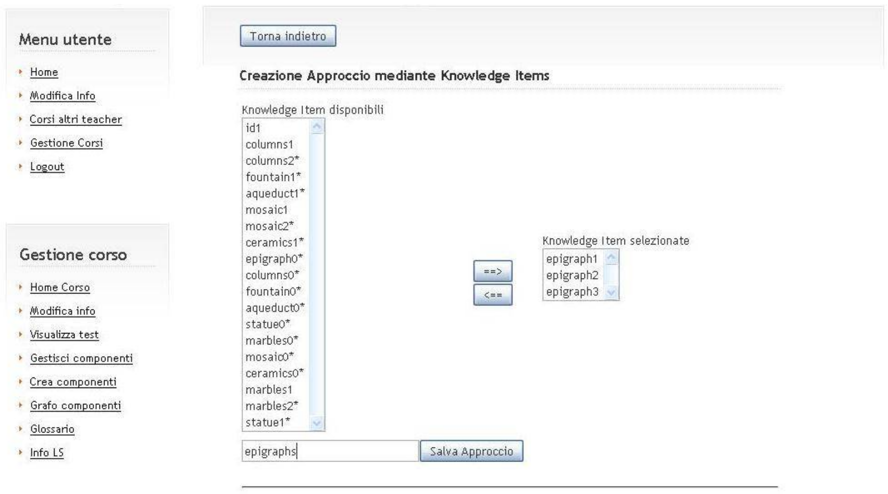


Figure 5.4: The domain expert's GUI for defining interests.

and interests. The Starting Knowledge Questionnaire is not mandatory, it can be a form to fill and should an SK evaluation totally lack, the system assume zero prior knowledge. Also interests are expressed through a questionnaire that can be a form to fill. Definition of interests allows the possibility for the visitor to express an intention to cover as much as possible about, for instance, the columns, or the epigraphs, throughout the whole site. Basically, an interest is represented by a keyword encompassing several KI, possibly appearing in several (not necessarily connected) LC.

Regarding the third step, Lecomps5 takes in input the LC of the domain of interest, characterized by their RK and AK, the Approach_of_Interest defined by the domain expert, the TK of the visit, the user SK and interests, and builds, through the Pdk planner, the personalized visit. In particular, interests are taking into account by considering *dynamic prerequisites*: given for example the graph shown in Fig. 5.5 and a user interested only in *epigraphs*, only the prerequisites related to *epigraphs* will be considered. The resulting graph is shown in Fig. 5.6. Currently, personalization is based only on user's interests and previous knowledge or places already visited that the user does not want to visit again; however, using the Pdk planner for producing the personalized tour can allow other possibilities, following the experience done in

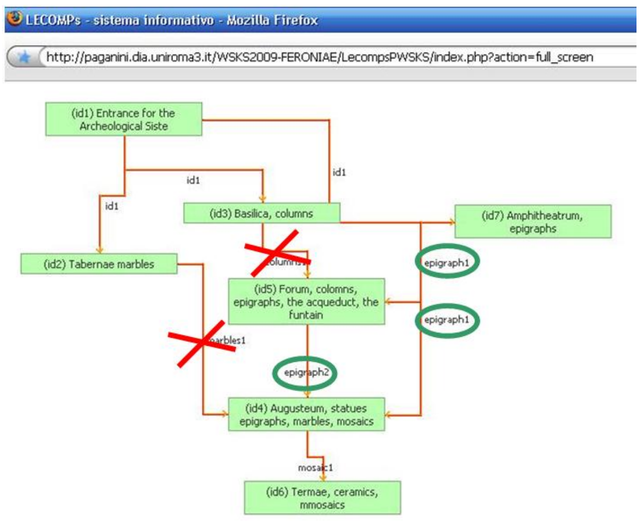

LECOMPs - sistema informativo - Mozilla Firefox
http://paganini.dia.uniroma3.it/WSKS2009-FERONIAE/LecompsPWSKS/index.php?action=full_screen
(id1) Entrance for the Archeological Siste
id1
id1
id1
(id3) Basilica, columns
(id7) Amphitheatrum, epigraphs
(id2) Tabernae marbles
columns1
epigraph1
(id5) Forum, colomns, epigraphs, the acqueduct, the funtain
epigraph1
marbles1
epigraph2
(id4) Augusteum, statues epigraphs, marbles, mosaics
mosaic1
(id6) Termae, ceramics, mmosaics

Figure 5.5: Graph of the considered cultural domain, i.e. *Lucus Feroniae*

the e-learning domain. For instance, the domain expert might want to specify that a visitor prefers to see only works of art with a given theme, or that a visitor is already expert and he wants to know only elaborations about a given work, or that a visitor does not know anything about the place he is visiting, or he is a child, and the explanation has to be as simple and direct as possible. In Section 5.4 an example of visit personalization is shown. The used domain is *Lucus Feroniae*, an archaeological site close to Rome.

5.4 Examples

Lucus Feroniae stands on a travertine platform located in *Capena*, a little old town close to Rome. It has very ancient origins, as ancient as the origins of worship of the *Feroniae Goddess*, a testimony of an italic cult like those discovered in sanctuaries of *Trebula Mutuesca*, *Terracina*, and *Amiterno*. The shrine is located at the 18*th* Mile of *Via Tiberina*, at *Scorano*, and the exact location was identified only in 1953, when Prince Victor Maximus, owner of the *Scorano Castle*, and surrounding lands, signaled to the Southern Etruria Superintendency the outcropping, during some works, of the archaeological findings.

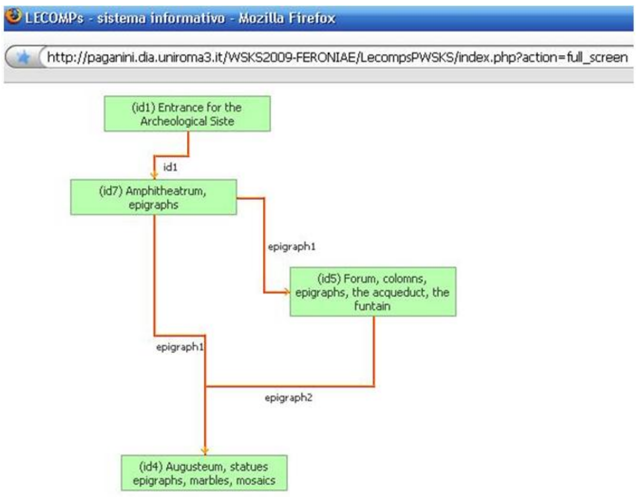

LECOMPs - sistema informativo - Mozilla Firefox
http://paganini.dia.uniroma3.it/WSKS2009-FERONIAE/LecompsPWSKS/index.php?action=full_screen
(id1) Entrance for the Archeological Siste
id1
(id7) Amphitheatrum, epigraphs
epigraph1
(id5) Forum, colomns, epigraphs, the acqueduct, the funtain
epigraph1
epigraph2
(id4) Augusteum, statues epigraphs, marbles, mosaics

Figure 5.6: Graph of the considered cultural domain, i.e. *Lucus Feroniae*, for a visitor interested only in *epigraphs*.

Fig.5.7 shows snapshots of the ancient *Via Tiberina*, where the site is located, and of the *Amphitheater*. When accessing the archaeological site, a crossroad

Figure 5.7: A Section of the ancient Via Tiberina and the Amphitheatre.

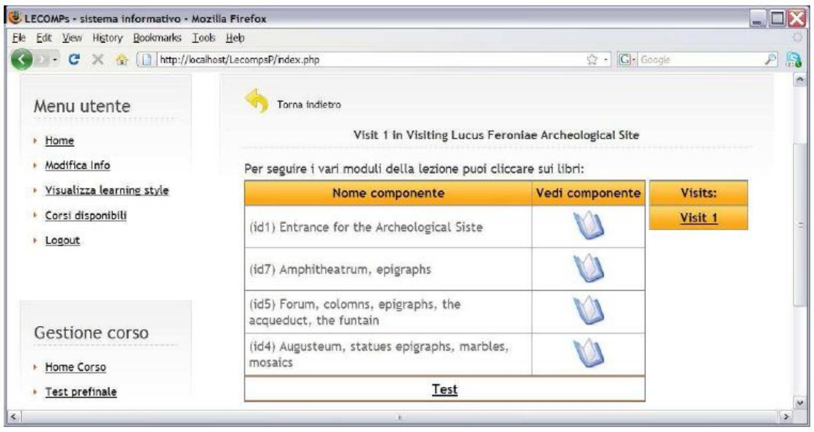
between the old *Via Tiberina* and the road to join the sanctuary to *Capena* is immediately met: the *Capenate Road* where the remains of an ancient gate can be seen. This crossroad was a very important road junction and in this place were found the *Cippi Miliari*, dated to the third century B.C., which is the dating of the most recent restoration of these roads. Continuing along the *Via Tiberina*, some not very large environments are present on the right, these environments have been identified as meeting and refreshment points, perhaps *Tabernae*. After a rectangular square with an East-West orientation where there is still a part of the ancient pavement made with rectangular slabs of limestone is present. Another interesting characteristic is the *Amphitheater* of which the load-bearing structures remained. It has a very unique form: it is almost circular, but, although very small, it presents all the characteristic aspects of a true *Amphitheater* with its doors still very well preserved, with the *Vomitoria*, that is, exits for the public, and service environments below the stairs. Finally, the south side is less preserved and recently, precisely in this area, came to light some structures certainly of the Republican Roman era. On the North there is the purely religious area, the focal point of the ancient political life and administration of worship in the city.

In [LSTV09b] two examples of personalized tours generation, based on user's interests, have been shown. These examples have considered that a visitor does not want to see all the works of art at the site, but he is interested in a special *theme track*, e.g.: he wants to see all the epigraphies, or all the tombs or all the statues and so on. Two visitors have been considered, with different preferences. The first one, with specific historical interests, wants to examine epigraphies, while the second one wants to see more in general fountains, statues, marbles and columns. The personalized tours, as generated by the system, are shown in Tab. 5.1, while Fig.5.8 and Fig.5.9 illustrate directly the tours proposed by the LECOMPS5 system.

Visitor A **epigraphies**	**Visitor B** **Statues, fountains** **marbles and columns**
Entrance to the Archeological Site	Entrance to the Archeological Site
Amphitheatrum	Tabernae
Forum	Basilica
Augusteum	Forum
	Augusteum

Table 5.1: The two different generated tours.

These generated tours are compatible with the preferences of the two different visitors. In fact, in the domain description, the *LC* present the following


LECOMPs - sistema informativo - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/LecompsP/index.php
Menu utente
Home
Modifica Info
Visualizza learning style
Corsi disponibili
Logout
Gestione corso
Home Corso
Test prefinale
Torna indietro

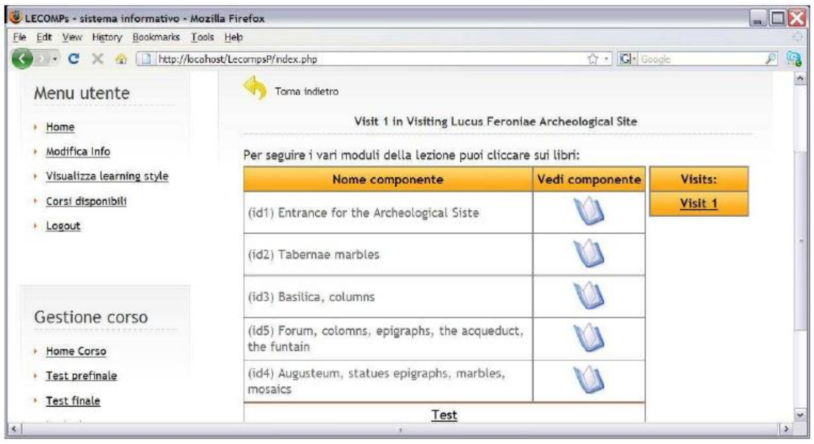
Visit 1 in Visiting Lucus Feroniae Archeological Site

Per seguire i vari moduli della lezione puoi cliccare sui libri:

Nome componente	Vedi componente
(id1) Entrance for the Archeological Siste	📖
(id7) Amphitheatrum, epigraphs	📖
(id5) Forum, colomns, epigraphs, the acqueduct, the funtain	📖
(id4) Augusteum, statues epigraphs, marbles, mosaics	📖
Test	

Visits:
Visit 1

Figure 5.8: The LECOMPS5 window with the proposed tour for the first visitor.


LECOMPs - sistema informativo - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/LecompsP/index.php
Menu utente
Home
Modifica Info
Visualizza learning style
Corsi disponibili
Logout
Gestione corso
Home Corso
Test prefinale
Test finale
Torna indietro

Visit 1 in Visiting Lucus Feroniae Archeological Site

Per seguire i vari moduli della lezione puoi cliccare sui libri:

Nome componente	Vedi componente
(id1) Entrance for the Archeological Siste	📖
(id2) Tabernae marbles	📖
(id3) Basilica, columns	📖
(id5) Forum, colomns, epigraphs, the acqueduct, the funtain	📖
(id4) Augusteum, statues epigraphs, marbles, mosaics	📖
Test	

Visits:
Visit 1

Figure 5.9: The LECOMPS5 window with the proposed tour for the second visitor.

characteristics:

- Amphitheatrum: epigraphies;
- Tabernae: marbles;
- Termae: ceramics and mosaics;
- Forum: columns, epigraphies, fountains and aqueduct;
- Augusteum: statues, epigraphies, marbles and mosaics;
- Basilica: columns.

The considered graph is shown in Fig. 5.5. The proposed sequences follow the prerequisite relations defined by the expert and the interests expressed by the users. For example, none of the two visitors have to visit the *Terme*, since they are not interested neither in ceramics nor in mosaics. The first visitor is suggested to visit only *Amphitheatrum*, *Forum* and *Augusteum* and in particular *Augusteum* is the last one to be visited, according to prerequisite relations.

Chapter 6

LS-Lab

As shown in Section 2.3, *Curriculum Sequencing* is one of the most interesting challenges in educational research area: research in this field aims to automatically produce a personalized sequence of didactic materials or activities, on the basis of each student's needs, by dynamically selecting the most appropriate didactic materials at any moment [BV03]. Several approaches addressing this issue have been proposed in the literature. In general each approach to personalization is comprised in a framework in which learning objects and student models are managed, and a suitable algorithm is employed to produce and maintain the Learning Object Sequence that represents the course. However each solution has its strengths and weaknesses, and it is conceivable that different teachers might prefer different personalization approaches. So, while from the one hand the number of proposals in the field of Curriculum Sequencing is increasing, on the other hand, to my knowledge, there is a lack of an environment where one can actually test, compare and evaluate different Curriculum Sequencing algorithms.

The necessity of a framework for comparing different Adaptive Educational Hypermedia is proved by a number of proposals for evaluating and testing different aspects of AEH, such as the framework presented in [ZYB08], for evaluating the performance of user modeling servers. Also [WL01] presents a framework to evaluate different adaptive features of systems based on the Case-Based Reasoning approach. In [BBHP02] a framework for comparing curriculum sequencing in AEH is presented. It proposes a distinction between *knowledge entities*, i.e. concepts, and *information entities*, i.e., learning materials, learning objects (or hypermedia systems nodes). This distinction is carried out for studying the sequencing problem and compare different sequencing algorithms. This comparison is based on four features: i) the relationship between knowledge entities and information entities, i.e. whether information entities correspond one-to-one to knowledge entities or there are different multiplicity between them; ii) the management of prerequisites and outcome of information entities; iii) input for sequencing algorithm; iv) sequencing algorithm. This last feature takes explicitly into account the curriculum sequencing algorithm comparison, but it is approached from a qualitative point of view.

In [LSV09b] the LS-Lab framework is proposed. LS-Lab is a self-contained integrated environment, with the aim to give researchers and teachers an instrument for quickly comparing, testing and evaluating different Curriculum Sequencing algorithms. It is not oriented to evaluate AEH as a whole, rather it wants to analyze the didactic strategies that come with different learning paths. In the LS-Lab system, different sequencing algorithms belonging to different adaptive educational environments are involved. These algorithms, through suitable software interfaces, e.g. parsers, run in the same environment with the same input. In such a way the comparison is made as fair as possible, also if some principles behind the user model and the way the instructional material is modeled are necessarily different. LS-Lab is a system in which the execution of different algorithms over sample student models is possible, and the different generated courses are presented to the teacher or to the researcher, annotated by a set of measures that can suggest and support evaluations. The system provides the means to:

- provide or use learning domains, IEEE LOM compliant, on which applying different algorithms and comparing their produced sequences;
- define sample student models;
- state a common goal for the student's courses to reach;
- allow the selection of the algorithms to be compared and the metrics to be used (among those included in the system);
- show the courses produced by the selected algorithms, each one annotated with the measures computed by the system through the selected metrics;
- store the results of each experiment, in order to allow the teacher to sum up his experiments and express motivated evaluations (this functionality is currently not provided).

In the following Sections the LS-Lab conceptual framework (see Section 6.1), architecture (see Section 6.2), algorithms and metrics (see Section 6.3) are described. Finally Section 6.4 shows two examples of comparison performed through the LS-Lab system.

6.1 The conceptual framework

To allow the comparison and measure of the behavior of several sequencing algorithms, various requirements have to be accommodated in the common framework. Firstly each algorithm has to be applicable to the respective appropriate data structures, i.e. domain description and student model. For giving a common definition of learning materials, namely Learning Objects (LO), a basic definition called Learning Node has been introduced. In this context a Learning Node has an analogous meaning as in definition 3, but it has been simplified. In particular the basic Learning Node is a 4-tuple: $LN = \langle LM, AK, RK, E \rangle$, where LM, AK, RK have the same meaning as in definition 3 (as regards KI, levels are not taken into account). E is a measure of the *effort* needed to study the LM, supposing that the requirements in RK are met. Learning Node can be enriched according to the specification of the different considered algorithms. For instance, learning styles related to the LM and LM typologies have been used in experiments involving the LS-Plan and the IWT [SCGM08] sequencing algorithms.

A common goal driven attitude is assumed to be applied by all the algorithms: each experiment applies an algorithm to produce a course with a determined Target Knowledge, as given in definition 8, expressed through KI. The common student model is an incremental one, in which new data for different algorithms can be added to the old ones. Given the possibility to run different algorithms it is also necessary to provide an initial set of metrics to be proposed for supporting the evaluating activity.

In this context a formal definition of Learning Objects Sequence can be useful:

Definition 11 Learning Objects Sequence *(LOS). It is a* $\{LN_1, \cdots, LN_n\}$ *sequence of* LN*, created by a sequencing algorithm.*

So a course (the actual student's learning activity) is a LOS defined by selecting LN from the pool, i.e. from the set of LN available for a given course (see definition 4), through a sequencing algorithm, taking care of its goal (TK) and of student's personal traits (such as SK, as given in definition 9).

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

- to be *goal driven*, i.e., to allow the learner for acquiring a TK,
- to be able to manage KI,
- to be able to manage, for each LN, its RK and its AK,
- to be able to manage IEEE-LOM [1] compliant LN.

The last one item is considered necessary in LS-Lab for allowing a common description of LO. In [LSV09b] how a LN can be mapped into some suitable IEEE-LOM metadata is shown. In particular, the contents of the fields <RK> and <AK> are represented by means of the tag <purpose> of the last IEEE-LOM category, <classification>: the tag <value> will contain *prerequisite* and *educational objective* respectively and each element of the <taxonPath>, <taxon>, will be composed by an identification tag <id> and by a string <string>, representing the name of a prerequisite, or of an AK. The *effort* can be mapped in the tag <difficulty> of the tag <educational>. This tag uses qualitative values that can be converted in numerical ones, if necessary.

Each sequencing algorithm has its own SM representation. In such a general system, it would be a very hard problem to build in advance a general SM, i.e., a SM that contains all possible SM representations. Hence, in LS-Lab in [LSV09b] the *Super Student Model* (SSM) has been introduced: it is a general SM container gradually enlarged with a new algorithm introduction into the system. The increasing law is the following:

$$SSM_i = SSM_{i-1} \cup SM_i \tag{6.1}$$

being SM_i the SM related to the new sequencing algorithm A_i to be inserted into the system, while SSM_{i-1} and SSM_i are the SSM before and after the new A_i insertion. With this type of representation, the SSM starts as an empty set: $SSM_0 = \emptyset$.

6.2 The system architecture

The LS-Lab functional components are shown in Fig. 6.1 where dashed arrows represent the input given by the teacher or by the researcher. The system is composed of the following functional modules:

[1] http://ltsc.ieee.org/wg12/

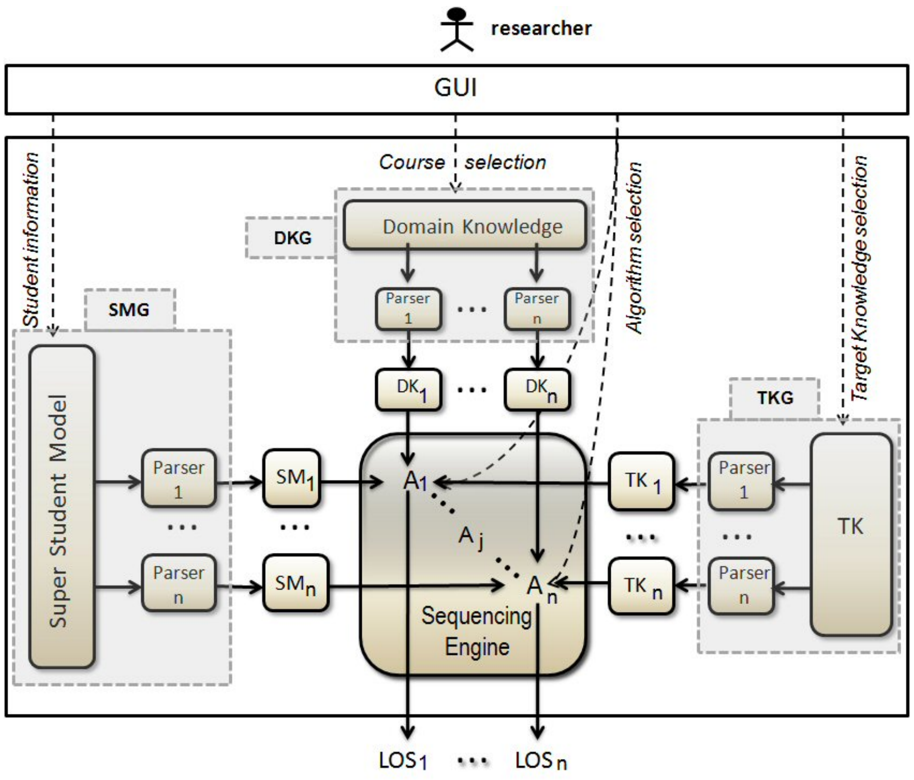
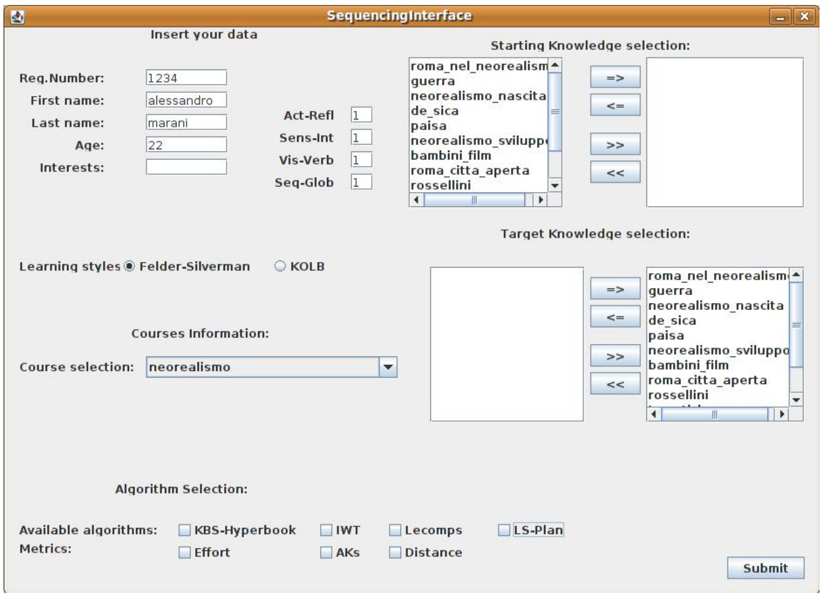


Figure 6.1: The Functional Schema of the LS-Lab system.

- *GUI*. It is the graphical environment, shown in Fig. 6.2, composed of six components. Through the *Insert your data* component the teacher or the researcher can input the personal data of the sample student. Through the *Courses Information* component one can select the course of interest, while through the *Starting Knowledge Selection* and the *Target Knowledge Selection* components the Student Starting Knowledge and the Course Target Knowledge can be input. Through the *Algorithm Selection* component the algorithms currently present in the environment can be selected in order to be used in the experiment. Finally, in the



Insert your data

Req.Number:	1234
First name:	alessandro
Last name:	marani
Age:	22
Interests:	

Act-Refl	1
Sens-Int	1
Vis-Verb	1
Seq-Glob	1

Starting Knowledge selection:

roma_nel_neorealism
guerra
neorealismo_nascita
de_sica
paisa
neorealismo_svilupp
bambini_film
roma_citta_aperta
rossellini

Target Knowledge selection:

roma_nel_neorealism
guerra
neorealismo_nascita
de_sica
paisa
neorealismo_sviluppo
bambini_film
roma_citta_aperta
rossellini

Learning styles [✓ Felder-Silverman] KOLB

Courses Information:

Course selection: neorealismo

Algorithm Selection:

Available algorithms: ☐ KBS-Hyperbook ☐ IWT ☐ Lecomps ☐ LS-Plan

Metrics: ☐ Effort ☐ AKs ☐ Distance

Figure 6.2: LS-Lab GUI.

metrics section the measures, currently provided by the system, can be used for a first objective evaluation of the produced sequences. The GUI actually allows to perform *experiments*, that is selecting: i) an algorithm (or more, if available), ii) a pool, iii) a TK, iv) a student model, v) metrics (optionally) and then activating the selected algorithms, so to produce, accordingly, comparable learning sequences for the student (model).

- *Student Model Generator* (SMG). It filters from the SSM the SM input associated with the algorithm, i.e. SM_i. In particular it takes the data inserted in the GUI, i.e. the student model information and the selected algorithms A_i, and uses the parsers related to A_i to translate the information of the GUI into the data structures needed as input by the algorithms, i.e. SM_i.

- *Domain Knowledge Generator* (DKG). This module takes as input the $Course_i$ selected by the researcher giving as output the DK_i related to that course, by launching the right parser in order to obtain the right DK (see definition 5) representation adapted to be managed by the algorithm A_i.

- *Target Knowledge Generator* (TKG). This module takes as input the TK selected by the researcher, and consequently, through a suitable parser, produces as output the right TK_i representation ready to be managed by the A_i algorithm.

- *The Sequencing Engine* (SE) contains the available sequencing algorithms $A_1 \dots A_i$. Each algorithm A_i takes as input the information processed by the three previous modules: SMG, that produces SM_i, DKG that produces DK_i, and TKG, that produces TK_i. The execution of A_i on input $\langle TK_i, DK_i, SM_i \rangle$ produces the learning objects sequence LOS_i.

LS-Lab can be used in two different scenarios. In the first one a teacher or a researcher runs two or more algorithms currently present in the system, while in the second one he wants to insert a new algorithm in the system comparing it with other available algorithms. The insertion of a new algorithm needs the effort to make uniform its associated SM, DK, and TK representations to the LS-Lab *standard*. In particular information about the algorithm input and output data and the algorithm executable file are necessary. Moreover parsers and adapters for SM, DK, and TK have to be developed. In this way in order to insert a new algorithm in LS-Lab it is necessary to provide:

- the algorithm executable file;

- the description of the algorithm input data;

- an *XML* file, describing the SM: it will be used by LS-Lab developers to extend the SSM and to provide a suitable adapter between the SSM and the actual SM, that can be given as input to the algorithm;

- a domain description compliant to IEEE-LOM standard and, if required, the semantics of the used tags: it will be used by LS-Lab developers to provide a suitable parser, that will translate the IEEE-LOM metadata of the learning materials of a given domain into the correct input for that particular sequencing algorithm;

- a TK description to allow LS-Lab developers for providing a suitable parser to give the correct input to the algorithm.

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

6.3 Algorithms, parsers and metrics in LS-Lab

In this section the state of the art of the LS-Lab system is illustrated: the algorithms currently available in the system; the parsers that have been developed in order to allow the use of the algorithms; the proposed metrics, provided by the system for supporting the comparison of the LOS produced by the algorithms. These metrics are only early instruments for the comparison and probably new metrics will be made available in future works.

Algorithms

Currently LS-Lab contains three algorithms: the algorithm used in the LS-Plan system, based on using the Pdk Planner and learning styles according to Felder and Silverman's model, the algorithm used in the IWT system [SCGM08] and the algorithm used in the KBS-Hyperbook system [HN01], both based on a classical topological sort algorithm (TSA).

Sequencing performed in the LS-Plan system has been widely illustrated in the related chapter 3: it is based on modeling the problem of course configuration as a planning problem, in which the student is the executing agent, LN are actions, RK are action preconditions, AK are action effects, the initial SM and TK constitute the initial world state and the goal respectively. The plan is the LOS proposed to the student, taking into account student LS if alternative LN are available.

The algorithm used in the IWT system has been described in Section 2.3, but it is useful to provide a brief description in the following. Ontology relations in IWT are essentially three: *HasPart*, *RequiredBy*, and *SuggestedOrder*. The *HasPart* nodes do not actually coincide with some learning material, they are used to express a higher level of concepts in the ontology. These relations produce a graph that is first augmented with the explicit relations *RequiredBy*. Then a topological sort algorithm is applied. It takes into account the *SuggestedOrder* relations when it has to choose the order of the next node among all the brothers. The *ExplainedBy* relation links learning objects to concepts of the ontology. The algorithm has been implemented by following the description given in [SCGM08], since IWT is a closed-source system. It also uses LS following Felder and Silverman model, as well as LS-Plan, but the way of deciding the suitable LN to choose is different from the LS-Plan strategy: LS-Plan associates to each LN some weights representing the suitability of the specific LN for a student with the defined LS, IWT, instead, associates weights to LN typologies.

The KBS-Hyperbook system generates a sequence of "information units" (a *trail*) to fill the learner with lacking knowledge and drives him towards the aimed topic. Such a trail comes from depth-first-traversal monitoring the possible acquisition of all "those KI that are prerequisites for the actual goal". After all the necessary units have been marked, they are sequenced by a simple topological sort algorithm. Learning styles are not considered in this process.

Parsers

Inserting a new algorithm in LS-Lab basically means to make uniform its associated domain, student model, and target knowledge representations to the LS-Lab *standard*. These phases consist in:

- according to the domain description, providing for necessary additional features of the LN, when the algorithm *intends* to manage them, and for a parser able to interpret the LN IEEE-LOM definitions to make them usable by the algorithm;
- according to the student model, updating, if necessary, the SSM with new information, when used by the algorithm, and developing a parser that filters the SSM producing the input necessary for the algorithm;
- according to the target knowledge description, providing a parser that translates the goal, expressed as a set of KI through the GUI, in the suitable data structures for the algorithm.

According to the domain description, section 6.1 has shown how appropriate metadata of the IEEE-LOM standard have been used for describing RK

and AK of a LN. This is enough information to let the KBS-Hyperbook algorithm work: the parser will create a graph of LN, and the algorithm will select the appropriate LN and will sequence them according with students' knowledge. Instead, LS-Plan deals with learning styles, so the LN have to be equipped (by the teacher) with LS weights. The binding of LS specification to IEEE-LOM metadata can be obtained by exploiting the tag <purpose> that belongs to the $9-th$ category of the IEEE-LOM <classification>. In the <taxonPath> field, the tag <taxon> contains an <id> and a <string> tag where the last one describes the LS associated to the LN as shown in Fig. 6.3. The LS-Plan domain parser acquires LS weights together with RK and AK

```
<imsmd:classification>
  <imsmd:purpose>
     <imsmd:source>LOMv1.0</imsmd:source>
     <imsmd:value>accessibility restrictions</imsmd:value>
  </imsmd:purpose>
  <imsmd:taxonPath>
     <imsmd:source>
       <imsmd:string>neorealismo</imsmd:string>
     </imsmd:source>
     <imsmd:taxon>
       <imsmd:id>Learning Style</imsmd:id>
       <imsmd:entry>
           <imsmd:string>act_rfl=4|sen_int=4|vis_vrb=5|seq_glb=3</imsmd:string>
       </imsmd:entry>
     </imsmd:taxon>
 </imsmd:taxonPath>
</imsmd:classification>
```

Figure 6.3: IEEE-LOM specifications for LS.

and translates them in the *domain* description (in the PDDL-K language required for Pdk), where each LN is expressed as an *action* with preconditions from RK and post-conditions from AK. For the IWT system in order to use a domain description uniform to the domain descriptions of LS-Plan and KBS-Hyperbook algorithms at the moment only *RequiredBy* relations are managed, in particular they are referred directly to the learning objects that explain a given concept. In other words currently the domain is not described through an ontology and the related learning objects, but the system considers only learning objects, described according to IEEE-LOM, and refers to them the *RequiredBy* relations of the related concepts. For managing learning styles in IWT it is necessary that the tag <learningResourceType> is set: IWT, in fact, associates to each possible value of this tag some values for learning styles, according to the Felder and Silverman's model. A parser for the management of this information has been developed.

According to the student model, as stated in section 6.1, the SSM starts from scratch, i.e., $SSM_0 \equiv \emptyset$, increasing as new SM are loaded into the system together with their related sequencing algorithms. KBS-Hyperbook requires KI that represent the student's SK, related to the topics he has to work with. In this case the SSM becomes:

$$SSM_1 = SSM_0 \cup SK$$

LS-PLAN requires, in addition, also the LS that describes the learning preferences of the student as defined in definition 2. So, the SSM at this stage will be as follows:

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

The LS-LAB GUI allows to insert, for the simulated student, learning styles, according to definition 2. IWT also uses Felder and Silverman's learning styles, but ranging in the range $[0, 1]$, therefore LS-LAB compresses the values inserted through the GUI in this range. In general, with the uploading of a new algorithm A_i into the system, the SSM_i can stay unchanged, if the information present in the SM_i is already contained in the SSM_i, or can be increased with new information that are not yet provided by the SSM_{i-1}. Also the data for SM_i will be filtered and parsed, to be taken as input by the algorithm A_i.

According to the target knowledge description, TK is modeled in LS-LAB as a set of KI, and it can be directly used by the KBS-Hyperbook and the IWT algorithms. LS-PLAN will translate TK into the *problem* specification, written in PDDL, in fact an execution of the algorithm needs specifications of both the domain and the problem, to produce the appropriate sequence.

Metrics

In order to tackle the *evaluation* aspects, two basic attitudes could be considered for the assessment of a LOS: in a *subjective comparison* attitude, the teacher judges the appropriateness, completeness and suitability of the sequence; a more *objective comparison* attitude uses some ways to measure the above-mentioned qualities of the course. In the following some metrics and heuristics to measure certain characteristics and qualities of the LOS are proposed and the result of such a process is offered to support teacher's LOS evaluation. Such set of metrics is by no means exhaustive.

Overall_Effort metrics

One possible way to measure a LOS is by computing the cognitive effort implied by the learning objects in the sequence. The *effort* is a value associated to a learning node, that has not a univocal meaning: it can represent the time expected to go through the learning content of the node, or the complexity of such contents. Moreover, it is not considered that the effort related to a LN might influence the effort related to another LN in the sequence, i.e. presently the effort of a LN is the same if taken alone or in a sequence.

So, the metrics M_E allows to compare LOS basing on the overall effort required by their respective set of LN, and it is defined by

$$M_E(LOS) = \sum_{i=1}^{n} LN_i.E$$

where $LN_i.E$ is the effort associated to LN_i. The less the effort, the simpler (or may be *shorter*) is the course.

Overall_Acquired_Knowledge metrics

This metrics allows to compare LOS by measuring how redundantly a LOS do actually cover the gap between SK and TK.

$$M_{AK}(LOS) = \cup_{i=1}^{n} LN_i.AK$$

where $LN_i.AK$ is the acquired knowledge associated to LN_i.

The smaller $M_{AK}(LOS)$ is, the more directly the course goes to the point (i.e.: the TK). Of course a “more direct course” is not necessarily “simpler” (according to $M_E(LOS)$) and in some cases a “less direct course” can be more suitable from a didactic point of view.

Step_Distance_Ratio metrics

The third measure is meant at supporting an assessment of similarity among different LOS. In particular the LOS is constructed by navigating the direct acyclic graph of the LN, and there can be different ways to do such a navigation. For instance a *depth first* rather *breadth first* attitude reflects into different didactic characteristics/strategies for the resulting courses; even if the set of LN may be the same, it is the flow of their presentation during the course that makes the difference.

Given that each node in the LN graph is labeled by its depth (0 for the source nodes, 1 for their direct successors, and so forth), basically relative distances between couples of nodes are computed: over a learning sequence $LOS = \{LN_1, LN_2, \ldots, LN_n\}$ the metrics represents the square of the Euclidean norm among adjacent LN in LOS, as it follows

$$M_{graph}(LOS) = (LN_1 - LN_2)^2 + (LN_2 - LN_3)^2 + \ldots + (LN_{n-1} - LN_n)^2$$

The higher $M_{graph}(LOS)$ is, the more "depth-first oriented" is the sequence, meaning that the didactic strategy of the course aims at deepening into details and derivations of each topic that is met, before going for further side topics.

Conversely, the smaller the measure, the more "breadth first" is the search attitude of the algorithm and the didactic strategy of the course. This time a layered approach is implemented by the course: a topic is met and further deepening is delayed until parallel topics have been met too.

Fig.6.4 shows the simple graph of learning nodes defined in a repository about *Italian Neorelist Cinema*. Each node has assigned a numerical value, indicating the depth of the node basing on the shortest path to reach it from a root node; those connections between a node and its successor, that do not affect the determination of the successor's depth are rendered by dashed arrows.

6.4 Examples

In the following two examples of comparison performed through the LS-Lab system are shown. Two different domains have been used: the first one is quite "humanistic", while the second one is more technical.

Italian Neorealist Cinema

The first example takes into consideration a student with $SK = \emptyset$ in the learning domain of *Italian Neorealist Cinema* (Fig.6.4). The TK, common to all the course examples, is formed by all the available KI.

Using the GUI in Fig.6.2, student learning styles are specified, in terms of the Felder-Silverman model: for each one of the four dimensions of the model, an integer between -11 and $+11$ is given (for instance, in the dimension of *perception* - with orientation given as *sensing* or *intuitive* - a value of -11 means fully sensing and a value of $+11$ means fully intuitive). Moreover, the Starting Knowledge and the Target Knowledge are given by either inheriting previous settings or directly selecting the related knowledge items from a menu. In the

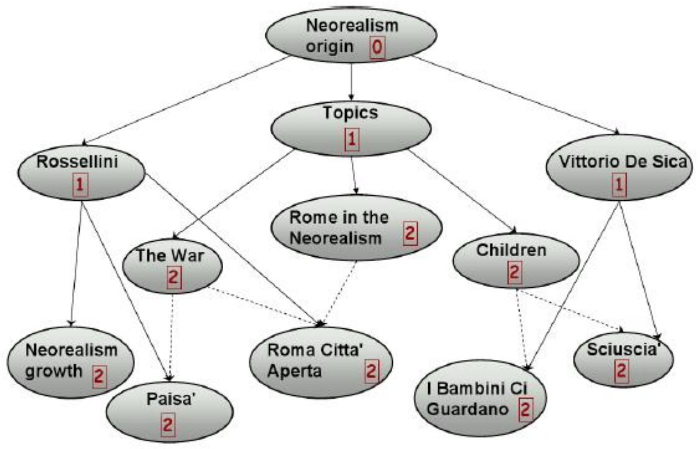


Figure 6.4: The graph of learning nodes in the *Italian Neorealist Cinema.* There might be different implementations for the same *LN*, according to the different learning styles to be covered. Here only a representative for such multiple defined *LN* is shown, with no effect other than making the representation less cumbersome.

lower part of the interface (just close to the *submit* button) the selection of the algorithms to apply and the metrics to use for the comparison is performed: in particular, all the known metrics and the KBS-Hyperbook, the IWT and the LS-Plan algorithms are selected. So, the system will produce three *LOS* to compare by three measures.

The result of the process is shown in Fig.6.5. The *LOS* are given as the list of their *LN* and are reported in the following. The *LN* can have different implementations for different learning styles; in particular, in the following, *LN*_1 and *LN*_2 stand for learning-style-aware versions of the same *LN* (*visual* based for _1 and *verbal* for _2); if no subscript is given, the *LN* had no learning-style-aware version in the repository. Names of some *LN* are translated for the reader's convenience.

LOS_{KBS}

{`Neorealism Origin_1, Topics_1, Children, The War, Rome in the Neorealism, Rossellini_1, Roma Città Aperta_1, Neorealism Growth_1, Paisà_2, Vittorio De Sica, Sciuscià, I bambini ci guardano_1` }
(learnig styles are not treated; the learning-style-aware nodes appearing in the sequence were selected by default, irrespectively to the learning style)

$LOS_{LS-Plan}$

{`Neorealism Origin_1, Rossellini_1, Neorealism Growth_1, Vittorio De Sica, Topics_1, Rome in the Neorealism, The War, Roma Città Aperta_1, Paisà_2, Children, Sciuscià, I bambini ci guardano_2` }

LOS_{IWT}

{`Neorealism Origin_1, Topics_1, Rome in the Neorealism, The War, Vittorio De Sica, Rossellini_1, Paisà_1, Neorealism Growth_1, Children, I bambini ci guardano_2, Roma Città Aperta_1, Sciuscià` }

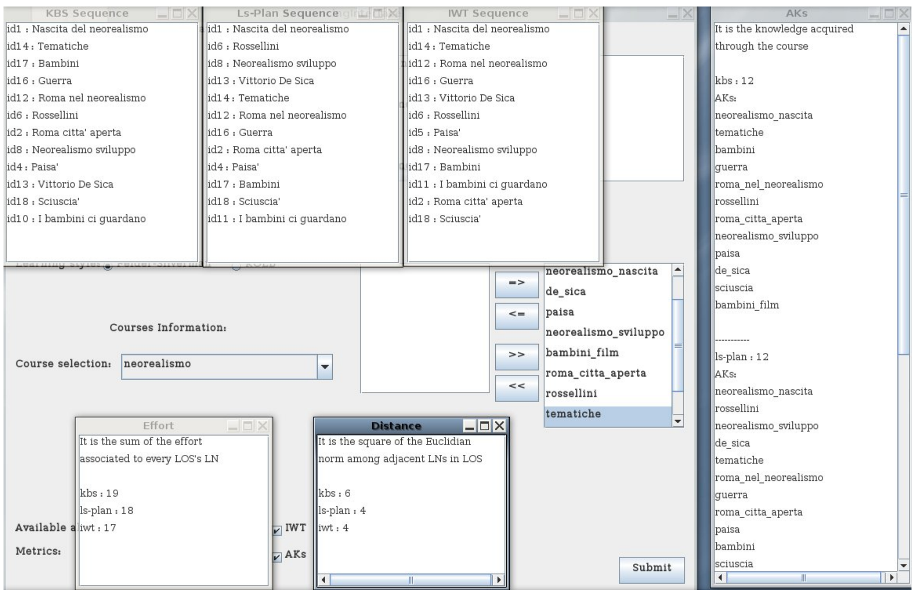
A relevant difference is in the positioning of the LN along the sequences. Moreover LS-Plan and IWT manage learning styles in a different way, consequently they select different learning-style-aware versions of the same LN (Paisà).

Since the pool of LN at hand is quite small, big differences in the measures for the first two metrics are not present; namely the three sequences contain all the same LN, although in different ordering and with different LS.

Regarding the Overall_Effort metrics, IWT proposes the less demanding sequence:

1. $M_E(LOS_{IWT}) = 17$
2. $M_E(LOS_{LS-Plan}) = 18$
3. $M_E(LOS_{KBS}) = 19$

Since the learning nodes are the same in all three sequences, this is actually due to the selection of different learning material with respect to the student's learning styles. As a matter of facts, from `LN_1` and `LN_2` (i.e. different versions of the same node, based on different learning styles) different efforts can be present.



KBS Sequence:
id1 : Nascita del neorealismo
id14 : Tematiche
id17 : Bambini
id16 : Guerra
id12 : Roma nel neorealismo
id6 : Rossellini
id2 : Roma citta' aperta
id8 : Neorealismo sviluppo
id4 : Paisa'
id13 : Vittorio De Sica
id18 : Sciuscia'
id10 : I bambini ci guardano

Ls-Plan Sequence:
id1 : Nascita del neorealismo
id6 : Rossellini
id8 : Neorealismo sviluppo
id13 : Vittorio De Sica
id14 : Tematiche
id12 : Roma nel neorealismo
id16 : Guerra
id2 : Roma citta' aperta
id4 : Paisa'
id17 : Bambini
id18 : Sciuscia'
id11 : I bambini ci guardano

IWT Sequence:
id1 : Nascita del neorealismo
id14 : Tematiche
id12 : Roma nel neorealismo
id16 : Guerra
id13 : Vittorio De Sica
id6 : Rossellini
id5 : Paisa'
id8 : Neorealismo sviluppo
id17 : Bambini
id11 : I bambini ci guardano
id2 : Roma citta' aperta
id18 : Sciuscia'

Course selection: neorealismo

neorealismo_nascita
de_sica
paisa
neorealismo_sviluppo
bambini_film
roma_citta_aperta
rossellini
tematiche

Effort:
It is the sum of the effort
associated to every LOS's LN

kbs : 19
ls-plan : 18
iwt : 17

Distance:
It is the square of the Euclidian
norm among adjacent LNs in LOS

kbs : 6
ls-plan : 4
iwt : 4

AKs:
It is the knowledge acquired
through the course

kbs : 12
AKs:
neorealismo_nascita
tematiche
bambini
guerra
roma_nel_neorealismo
rossellini
roma_citta_aperta
neorealismo_sviluppo
paisa
de_sica
sciuscia
bambini_film

----
ls-plan : 12
AKs:
neorealismo_nascita
rossellini
neorealismo_sviluppo
de_sica
tematiche
roma_nel_neorealismo
guerra
roma_citta_aperta
paisa
bambini
sciuscia

Figure 6.5: Experimental use of LS-Lab: learning sequences comparisons phase.

As of the Overall_Acquired_Knowledge metrics, since the three sequences share the same LN, they bring to the acquisition of the same KI:

$$M_{AK}(LOS_{KBS}) = M_{AK}(LOS_{LS-Plan}) = M_{AK}(LOS_{IWT}) = 12$$

This "comparison" is not very much conclusive indeed; this is basically due to the limited extent of the repository.

Finally, the Step_Distance_Ratio metrics says of a common attitude for LS-Plan and IWT:

1. $M_{graph}(LOS_{LS-Plan}) = M_{graph}(LOS_{IWT}) = 4$
2. $M_{graph}(LOS_{KBS}) = 6$

The proposed interpretation of this last measure is that it is related to the didactic strategy adopted by the algorithm. If a measure is small it means that topics have been introduced in a layered fashion, by first introducing the concept throughout the first layer and then deepening each one of them a step at a time, without deepening much before that a layer of the LN tree has been visited. This is a *breadth first* behavior: after a node has been visited, it favors the visit of nodes at the same level. Conversely, if the measure is high it means that the general attitude is to keep in touch with an aspect of the course and immediately deepen along the branch of following nodes; then, coming back to start a new branch increases the measure. This is a *depth first* behavior: once a node has been visited, it favors the visit of the derived nodes.

Recursive Programs

The second example considers a more articulate domain, shown in Fig. 6.6. Two distinct student models have been considered, with common SK, common TK, but different learning styles; the two models are as it follows (the learning styles values are specified in the same order than in Fig.6.2):

Student1

SK : rec_runtimestack_k, rec_runtimestack_a

TK : rec_exercises

LS : $-10; 8; 8; 10$ (Active-Intuitive-Verbal-Global, cf. explanations given about Fig.6.2 - first item)

Student2

SK : rec_runtimestack_k, rec_runtimestack_a

TK : rec_exercises

LS : $9; -9; -9; -9$ (Reflecting-Sensing-Visual-Sequential)

Table 6.1 shows the learning path produced over the first student model (call it *student1* the related learner in the following).
From the learning nodes sequences shown in Table 6.1 it is possible to observe that:

- LS-Plan has one additional LN (id 14) and, consequently a bigger effort;
- all the three sequences present the same learning nodes, proposed in different order;

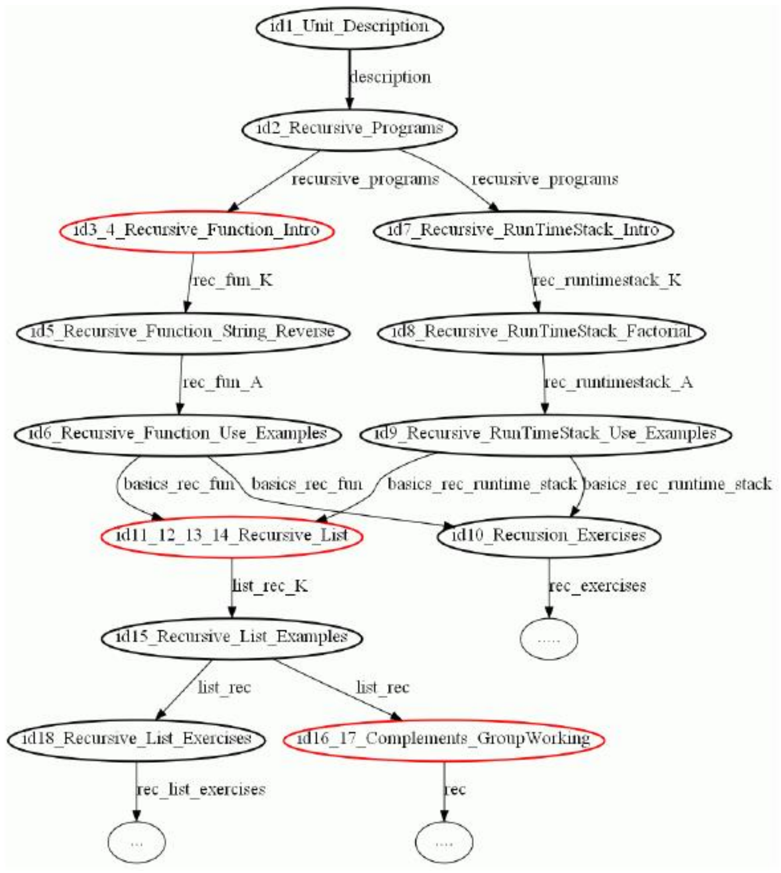

id1_Unit_Description
description
id2_Recursive_Programs
recursive_programs
recursive_programs
id3_4_Recursive_Function_Intro
id7_Recursive_RunTimeStack_Intro
rec_fun_K
rec_runtimestack_K
id5_Recursive_Function_String_Reverse
id8_Recursive_RunTimeStack_Factorial
rec_fun_A
rec_runtimestack_A
id6_Recursive_Function_Use_Examples
id9_Recursive_RunTimeStack_Use_Examples
basics_rec_fun
basics_rec_fun
basics_rec_runtime_stack
basics_rec_runtime_stack
id11_12_13_14_Recursive_List
id10_Recursion_Exercises
list_rec_K
rec_exercises
id15_Recursive_List_Examples
list_rec
list_rec
id18_Recursive_List_Exercises
id16_17_Complements_GroupWorking
rec_list_exercises
rec

Figure 6.6: The graph of *LN* in the Recursion Domain. Multiple indexes point out nodes given in different learning-styles-aware versions: `id3_4, id11_12_13_14, id16_17` (indexes correspond to actual learning material).

- the node “Recursive Function Intro” is proposed in LS-Plan with a different learning style, w.r.t. the other two algorithms. The chose of *LS* for LS-Plan is motivated from that fact that the *LS* associated into the pool to *LN* id3 is [-11; -5; -9; -2], for *LN* id4 is [8; -1; 11; 1]. Since LS-Plan computes the euclidean distance between the student *LS* and the

KBS	**LS-PLAN**	**IWT**
	Learning Paths	
id1 : Unit description	id1 : Unit description	id1 : Unit description
id2 : Recursive programs	id2 : Recursive programs	id2 : Recursive programs
id3 : Recursive function intro	*id4 : Recursive function intro*	id9 : Recursive runtime stack use examples
id5 : Recursive function string reverse	id5 : Recursive function string reverse	*id3 : Recursive function intro*
id6 : Recursive function use examples	id6 : Recursive function use examples	id5 : Recursive function string reverse
id9 : Recursive runtime stack use examples	id9 : Recursive runtime stack use examples	id6 : Recursive function use examples
id10 : Recursion exercises	id14 : Recursive list	id10 : Recursion exercises
	id10 : Recursion exercises	
	Effort	
13	16	13
	Distance	
5	5	17

Table 6.1: Results for Student model 1 with SK=rec_runtimestack_k, rec_runtimestack_a TK= rec_exercises and LS= $-10; 8; 8; 10$

LN (as shown in [LSV09a]) the node id4 is the closest one to the student1 LS, being Euclidean norm of (student1,id3) = 24.55, with respect to the Euclidean norm of (student1,id4) that is = 22.24. In the case of IWT the way of computing the closest node is different: IWT associates LS to the node, on the base of the type of the resources, that can be of fourteen different types: (Exercise, Simulation, Questionnaire, Diagram, Figure, Graph, Index, Slide, Table, Narrative Text, Exam, Experiment, Problem statement, SelfAssessment).

- the step-distance measure is the same for KBS and LS-Plan, and low indeed (= 5), while it is very high (= 17) for IWT. This means that KBS and LS-Plan follow a presentation flow that is "breadth-first" oriented, while IWT is more "depth-first" oriented (cf. Fig. 6.6): since the learner is already acquainted with the concept of runtime stack environment, IWT starts with proposing the completion for this concept and afterwards it comes back to explain the theoretical principles of recursive functions. On the other way around, KBS and LS-Plan starts from the-

KBS	**LS-PLAN**	**IWT**
	Learning Paths	
id1 : Unit description	id1 : Unit description	id1 : Unit description
id2 : Recursive programs	id2 : Recursive programs	id2 : Recursive programs
id3 : Recursive function intro	*id3 : Recursive function intro*	id9 : Recursive runtime stack use examples
id5 : Recursive function string reverse	id5 : Recursive function string reverse	*id4 : Recursive function intro*
id6 : Recursive function use examples	id6 : Recursive function use examples	id5 : Recursive function string reverse
id9 : Recursive runtime stack use examples	id9 : Recursive runtime stack use examples	id6 : Recursive function use examples
id10 : Recursion exercises	id13 : Recursive list	id10 : Recursion exercises
	id10 : Recursion exercises	
	Effort	
13	14	14
	Distance	
5	5	17

Table 6.2: Results for Student model 2 with $SK=$ rec_runtimestack_k, rec_runtimestack_a $TK=$ rec_exercises and $LS= 9; -9; -9; -9$

oretical principles of recursive functions and then, before presenting the learning node id10, propose the example (id9: Recursive runtime stack use examples).

On these bases the teacher has some elements for judging and comparing the behavior of the algorithms.

Then, Table 6.2 shows the *LOS* produced by the system over the second student model, and the same considerations made above can be applied and compared with the second student model (call it *student2*). That model shows same *SK* and *TK* as student1, but rather opposite learning styles. In this case the behavior of the algorithms, and the qualities of their *LOS* are quite similar to the previous case, but for one difference in that KBS proposes node id3, as the default node, irrespective of learning styles that KBS does not manage, while LS-Plan and IWT propose id3 and id4 respectively, basing on the respective adaptation algorithms.

Conclusion

This dissertation focused on the problem of providing a reasonable compromise between personalization and its costs, in terms of domain expert's effort necessary for producing personalization. In the educational domain some methodologies for providing an effective personalization from the student's point of view together with an "efficient" personalization from the teacher's point of view have been proposed. Personalization is provided on the basis of student's knowledge and learning styles and the student is followed and guided during his study like a teacher could do: proposing a sequence of contents suitable for the student at the beginning of the course and performing recovery strategies, during the fruition of the course, if the study does not proceed as it should. The teacher is required to specify few metadata, necessary for characterizing learning materials, such as prerequisite relations and suitability of contents for a given type of student; he is helped by a graphical interface, allowing a global vision of the course; personalized courses are generated automatically on the basis of the given student model and of teacher's didactic preferencies. These methodologies have been implemented in the LS-Plan system, that has been developed as a plug-in for providing educational hypermedia with adaptivity. It has been integrated in the Lecomps educational hypermedia in order to carry out evaluations both from the student's and from the teacher's point of view, that have shown positive results. Future works aim to fully integrate LS-Plan in a widespread Learning Management System, such as Moodle. The aim is in fact to enhance a widespread e-learning platform with personalization. In this way advantages provided by adaptive educational systems, i.e. educational experiences tailored to individual students, can be added to well-known systems for distance learning education. Technological solutions for carrying out such integration are currently in studying: a web services framework or a plug-in in Moodle can be possible solutions. Moreover this integration opens other possibilities for personalization, that is for instance exploiting Moodle logging and collaboration tools for extending the LS-Plan student modeling, and, consequently, the provided adaptation. The use of different planners for *curriculum sequencing* will be also studied, for instance planners with resources will be considered for managing other kind of constraints, such as temporal ones. The use of statistical analysis will also be considered for supporting the teacher in assigning weights to a learning material, for expressing its suitability for a given type of student: the teacher could set weigths to zero and a statistical analysis about the outcome of the students during the course fruition can suggest suitable ones.

Personalization methodologies proposed for the educational domain, have been also applied to cultural visits personalization, detecting analogies between the problem of course personalization and the problem of cultural visits personalization and performing the necessary integrations, i.e. taking into account user's interests. The proposed approach allows *didactic* cultural tours, that is tours that respect logical constraints among places to be visited, works of art, or their explanations. It is currently applicable only to virtual or blended visits, i.e. partially virtual and partially on-site visits. Future works aim to take into account also physical constraints proposing a compromise between physical and logical constraints satisfaction.

The dissertation has also proposed LS-Lab, a system developed with the aim of providing a uniform environment for comparing different *curriculum sequencing* approaches. Future works aim to integrate in the system new algorithms and new metrics for learning sequencing comparison.

Bibliography

[ABCL90] J. R. Anderson, C. F. Boyle, A. T. Corbett, and M. W. Lewis. Cognitive modeling and intelligent tutoring. *Artificial Intelligence*, 42(1):7–49, 1990.

[ACKP95] J. R. Anderson, A. T. Corbett, K. R. Koedinger, and R. Pelletier. Cognitive tutors: Lessons learned. *Journal of the Learning Sciences*, 4(2):167–207, 1995.

[ACM$^+$06] E. Alfonseca, R. M. Carro, E. Martin, A. Ortigosa, and P. Paredes. The impact of learning styles on student grouping for collaborative learning: A case study. *User Modeling and User-Adapted Interaction*, 16(3-4):377–401, 2006.

[AD03] N. A. Abdullah and H. Davis. Is simple sequencing simple adaptive hypermedia? In *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia (HYPERTEXT '03)*, pages 172–173, 2003.

[AGP$^+$02] L. Ardissono, A. Goy, G. Petrone, M. Segnan, and P. Torasso. Ubiquitous user assistance in a tourist information server. In *Proc. of 2-nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002)*, number 2347 in Lecture Notes in Computer Science, pages 14–23. Springer, 2002.

[BAR02] P. De Bra, A. Aerts, and B. Rousseau. Concept relationship types for AHA! 2.0. In *Proc. of World Conference on E-Learning, E-Learn 2002*, pages 1386–1389, 2002.

[BBB$^+$07] M. Baldoni, C. Baroglio, I. Brunkhorst, E. Marengo, and V. Patti. A service-oriented approach for curriculum planning and validation. In *Proceedings of International Workshop on Agents, Web-Services, and Ontologies - Integrated Methodologies, Durham, UK (6th–7th September 2007)*, pages 108–123, 2007.

[BBHP02] M. Baldoni, C. Baroglio, N. Henze, and V. Patti. Setting up a framework for comparing adaptive educational hypermedia: First steps and application on curriculum sequencing. In *In Proc. of ABIS-Workshop 2002: Personalization for the Mobile World, Workshop on Adaptivity and User Modeling in Interative Software Systems*, pages 43–50, 2002.

[BBPT04] M. Baldoni, C. Baroglio, V. Patti, and L. Torasso. Reasoning about learning object metadata for adapting SCORM courseware. In *Proc. of International Workshop on Engineering the Adaptive Web: Methods and Technologies for personalization and Adaptation (EAW'04)*, 2004.

[BCK05] J. Baus, K. Cheverst, and C. Kray. A survey of map-based mobile guides. In L. Meng, A. Zipf, and T. Reichenbacher, editors, *Map-based Mobile Services Theories, Methods and Implementations*, pages 193–209. Springer, 2005.

[BES98] P. Brusilovsky, J. Eklund, and E. Schwarz. Web-based education for all: A tool for developing adaptive courseware. In *Computer Networks and ISDN Systems (Proceedings of Seventh International World Wide Web Conference)*, 1998.

[BFB07] E. Brown, T. Fisher, and T. Brailsford. Real users, real results: examining the limitations of learning styles within AEH. In *Proceedings of the eighteenth conference on Hypertext and hypermedia*, pages 57–66, 2007.

[BG05] A. Berlanga and F. Garcia. IMS-LD reusable elements for adaptive learning designs. *Journal of Interactive Media in Education. (Special Issue Advances in Learning Design)*, 2005.

[BHF03] N. Bajraktarevic, W. Hall, and P. Fullick. Incorporating learning styles in hypermedia environment: Empirical evaluation. In *Proc. Fourteenth Conference on Hypertext and Hypermedia*, pages 41–52, 2003.

[BHW99] P. De Bra, G. J. Houben, and H. Wu. AHAM: A Dexter-based reference model for adaptive hypermedia. In *Proceedings of the 10th ACM Conference on Hypertext and Hypermedia*, pages 147–156, 1999.

[BKS04] P. Brusilovsky, C. Karagiannidis, and D. Sampson. Layered evaluation of adaptive learning systems. *International Journal of Continuing Engineering Education and Lifelong Learning*, 14(4/5):402–421, 2004.

[Blo64] B.S. Bloom. *Taxonomy of Educational Objectives.* David McKay Comp. Inc., 1964.

[BM07] P. Brusilovsky and E. Millan. User models for adaptive hypermedia and adaptive educational systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, pages 3–53. Springer-Verlag, Berlin Heidelberg New York, 2007.

[Bru94] P. Brusilovsky. The construction and application of student models in intelligent tutoring systems. *Journal of Computer and Systems Sciences International*, 32(1):70–89, 1994.

[Bru96] P. Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction (Special issue on adaptive hypertext and hypermedia)*, 6(2–3):87–129, 1996.

[Bru00] P. Brusilovsky. Adaptive hypermedia: From intelligent tutoring systems to web-based education (invited talk). In *Proc. of 5-th International Conference on Intelligent Tutoring Systems (ITS 2000)*, number 1839 in Lecture Notes in Computer Science, pages 1–7. Springer Berlin/Heidelberg, 2000.

[Bru01] P. Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11(1-2):87–110, 2001.

[Bru03] P. Brusilovsky. Developing adaptive educational hypermedia systems: from design models to authoring tools. In T. Murray, S. Blessing, and S. Ainsworth, editors, *Authoring Tools for Advanced Technology Learning Environment*, pages 377–410. Kluwer Academic Publishers, 2003.

[BSS06] P. De Bra, D. Smits, and N. Stash. Creating and delivering adaptive courses with AHA! In *EC-TEL*, pages 21–33, 2006.

[BSW96] P. Brusilovsky, E. Schwarz, and G. Weber. ELM-ART: An intelligent tutoring system on world wide web. In *Proceedings of Third International Conference on Intelligent Tutoring Systems (ITS96)*, pages 261–269, 1996.

[BTK06] D. Burgos, C. Tattersall, and R. Koper. Representing adaptive elearning strategies in IMS Learning Design. In *TENCompetence Conference*, 2006.

[BV03] P. Brusilowsky and J. Vassileva. Course sequencing techniques for large-scale web-based education. *International Journal of Continuing Engineering Education and Life-long Learning*, 13:75–94, 2003.

[Byl94] T. Bylander. The computational complexity of propositional strips planning. *Artif. Intell.*, 69(1-2):165204, 1994.

[CB06] A. Cristea and D. Burgos. Authoring adaptive hypermedia and IMS Learning Design: A possible understanding? In *Proceedings of the Sixth International Conference on Advanced Learning Technologies (ICALT'06)*, 2006.

[CC03] A. Cristea and L. Calvi. The three layers of adaptation granularity. In *Proc. of UM 2003*, number 2702 in LNAI, pages 4–14, 2003.

[CDMF00] K. Cheverst, N. Davies, K. Mitchell, and A. Friday. Experiences of developing and deploying a context-aware tourist guide: the GUIDE project. In *Proc. of the 6-th International Conference on Mobile computing and networking (MobiCom 2000)*, pages 20–31, 2000.

[CGV02] C. Conati, A. Gertner, and K. Vanlehn. Using bayesian networks to manage uncertainty in student modeling. *User Modeling and User-Adapted Interaction*, 12(4):371–417, 2002.

[Chi01] D. N. Chin. Empirical evaluation of user models and user-adapted systems. *User Modeling and User-Adapted Interaction*, 11(1):181–194, 2001.

[CHL99] C. A. Carver, R. A. Howard, and W. D. Lane. Enhancing student learning through hypermedia courseware and incorporation of student learning styles. *IEEE Trans. on Education*, 42(1):33–38, 1999.

[CM03a] A. Cristea and A. De Mooij. Adaptive course authoring: My Online Teacher. In *Proc. of ICT'03*, volume 2, pages 1762–1769, 2003.

[CM03b] A. Cristea and A. De Mooij. LAOS: Layered WWW AHS authoring model and its corresponding algebraic operators. In *Proc. of WWW'03*, 2003.

[CMHE04] F. Coffield, D. Moseley, E. Hall, and K. Ecclestone. Should we be using learning styles? What research has to say to practice. *Learning & Skills Research Centre*, 2004.

[Fis01] G. Fischer. User modeling in human–computer interaction. *User Modeling and User-Adapted Interaction*, 11(1-2):65–86, 2001.

[FKV$^+$90] J.C. Falmagne, M. Koppen, M. Villano, J.P. Doignon, and L. Johannesen. Introduction to knowledge spaces: How to build, test, and search them. *Psychological Review*, 97(2):201–224, 1990.

[FS88] R. M. Felder and L. K. Silverman. Learning and teaching styles in engineering education. *Engineering Education*, 78(7):674–681, 1988.

[FS05] R. M. Felder and J. Spurlin. Application, reliability and validity of the index of learning styles. *Int. Journal of Engineering Education*, 21(1):103–112, 2005.

[GBGPZ06] D. Goren-Bar, I. Graziola, F. Pianesi, and M. Zancanaro. The influence of personality factors on visitor attitudes towards adaptivity dimensions for mobile museum guides. *User Model. User-Adapt. Interact*, 16(1):31–62, 2006.

[Gen05] C. Gena. Methods and techniques for the evaluation of user-adaptive systems. *Knowl. Eng. Rev.*, 20(1):1–37, 2005.

[GK07] S. Graf and Kinshuk. Providing adaptive courses in learning management systems with respect to learning styles. In *Proc.*

of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (eLearn 2007), pages 2576–2583, 2007.

[GN00] E.R. Gansner and S.C. North. An open graph visualization system and its applications to software engineering. *Software Practice and Experience*, 30(11):1203–1233, 2000.

[GS02] A. Gerevini and I. Serina. LPG: A planner based on local search for planning graphs with action costs. In *Proceedings of AIPS'02*, 2002.

[HL56] J. Hodges and E. Lehmann. The efficiency of some non parametric competitors of the t-test. *Ann. Math. Statist*, 35:324–335, 1956.

[HL63] J. Hodges and E. Lehmann. Estimates of location based on rank tests. *Ann. Math. Statist*, 34:598–611, 1963.

[HN01] N. Henze and W. Nejdl. Adaptation in open corpus hypermedia. *International Journal of Artificial Intelligence in Education*, 12(4):325–350, 2001.

[HWC73] M. Hollander, D. Wolfe, and R. Cohen. *Nonparametric Statistical Methods*. John Wiley and Sons, 1973.

[Jam96] A. Jameson. Numerical uncertainty management in user and student modeling: An overview of systems and issues. *Journal User Modeling and User-Adapted Interaction*, 5(3-4):193–251, 1996.

[Kav04] A. Kavcic. Fuzzy user modeling for adaptation in educational hypermedia. *IEEE Transactions on Systems, Man, and Cybernetics*, 34(4):439–449, 2004.

[KB05] R. Koper and D. Burgos. Developing advanced units of learning using IMS Learning Design level b. *International Journal on Advanced Technology for Learning*, 2(4):252–259, 2005.

[KDB04] E. Kosba, V. Dimitrova, and R. Boyle. Using fuzzy techniques to model students in web-based learning environments. *Int. J. Artif. Intell. Tools Special Issue on AI Techniques in Web-Based Educational Systems*, 13(2):279–297, 2004.

[KDB07] E. Kosba, V. Dimitrova, and R. Boyle. Adaptive feedback generation to support teachers in web-based distance education. *User Modeling and User-Adapted Interaction*, 17(4):379–413, 2007.

[KM04] R. Koper and J. Manderveld. Educational modelling language: modelling reusable, interoperable, rich and personalised units of learning. *British Journal of Educational Technology*, 35(5):537–551, 2004.

[Kol84] D. A. Kolb. *Experiential Learning: Experience as the Source of Learning and Development.* Prentice-Hall, 1984.

[KS05a] P. Karampiperis and D. Sampson. Adaptive learning resources sequencing in educational hypermedia systems. *Educational Technology and Society*, 8(4):128147, 2005.

[KS05b] P. Karampiperis and D. Sampson. Automatic learning object selection and sequencing in web-based intelligent learning systems. In Zongmin Ma, editor, *Web-Based Intelligent e-Learning Systems: Technologies and Applications*, page 5671. Information Science Publishing, 2005.

[KS05c] P. Karampiperis and D. Sampson. Designing learning services for open learning systems utilizing IMS Learning Design. In *Proceedings of the 4th IASTED International Conference on Web-Based Education (WBE 2005)*, pages 279–283, 2005.

[LLWF05] T. A. Litzinger, S. H. Lee, J. C. Wise, and R. M. Felder. A study of the reliability and validity of the Felder-Soloman Index of Learning Styles. In *Proceedings of the 2005 American Society for Engineering Education Annual Conference and Exposition*, 2005.

[LSTV] C. Limongelli, F. Sciarrone, M. Temperini, and G. Vaste. The Lecomps5 framework for personalized web-based learning: a teacher's satisfaction perspective. *Computers in Human Behavior (to appear).*

[LSTV08a] C. Limongelli, F. Sciarrone, M. Temperini, and G. Vaste. Lecomps5: a framework for the automatic building of personalized learning sequences. In *Proceedings of the 1st World Summit on the Knowledge Society (WSKS 2008)*, number 5288 in LNCS/LNAI, pages 296–303. Springer Berlin/Heidelberg, 2008.

[LSTV08b] C. Limongelli, F. Sciarrone, M. Temperini, and G. Vaste. Lecomps5: a web-based learning system for course personalization and adaptation. In *Proceedings of the IADIS Multi Conference on Computer Science and Information Systems, e-Learning 2008*, pages 325–332, 2008.

[LSTV09a] C. Limongelli, F. Sciarrone, M. Temperini, and G. Vaste. Adaptive learning with the LS-Plan system: A field evaluation. *IEEE Transactions on Learning Technologies*, 2009.

[LSTV09b] C. Limongelli, F. Sciarrone, M. Temperini, and G. Vaste. Virtual cultural tour personalization by means of an adaptive e-learning system: A case study. In *Proceedings of the 2nd World Summit on the Knowledge Society (WSKS 2009)*, LNCS/LNAI, pages 40–49. Springer Berlin/Heidelberg, 2009.

[LSV08] C. Limongelli, F. Sciarrone, and G. Vaste. LS-Plan: An effective combination of dynamic courseware generation and learning styles in web-based education. In *Proc. of 5-th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2008)*, number 5149 in Lecture Notes in Computer Science, pages 133–142. Springer Berlin/Heidelberg, 2008.

[LSV09a] C. Limongelli, F. Sciarrone, and G. Vaste. An Application of the LS-Plan System to an Educational Hypermedia. *Int. J. of Web-Based Learning and Teaching Technologies*, 4(1):16–34, 2009.

[LSV09b] C. Limongelli, F. Sciarrone, and G. Vaste. LS-Lab: A framework for comparing curriculum sequencing algorithms. In *Proc. of 9th International Conference of intelligent Systems, Design and Application (ISDA 09)*, 2009.

[Mas03] J. Masthoff. The evaluation of adaptive systems. In N. V. Patel, editor, *Adaptive evolutionary information systems*, pages 329–347. Idea Group publishing, 2003.

[MCFOGF08] L. Morales, L. Castillo, J. Fernandez-Olivares, and A. Gonzalez-Ferrer. Automatic generation of user adapted learning designs: An AI-Planning proposal. In *Proc. of 5-th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*

(AH 2008), number 5149 in Lecture Notes in Computer Science, pages 324–328. Springer Berlin/Heidelberg, 2008.

[MG95] R. J. Mislevy and D. H. Gitomer. The role of probability-based inference in an intelligent tutoring system. *Journal User Modeling and User-Adapted Interaction*, 5(3-4):253–282, 1995.

[MLOP07] M. Cialdea Mayer, C. Limongelli, A. Orlandini, and V. Poggioni. Linear temporal logic as an executable semantics for planning languages. *J. of Logic, Lang. and Inf.*, 1(16):63–89, 2007.

[NCMW08] L. Noel, O. Carloni, N. Moreau, and S. Weiser. Designing a knowledge-based tourism information system. *Int. J. of Digital Culture and Electronic Tourism*, 1(1):1–17, 2008.

[Pas76] G. Pask. Styles and strategies of learning. *British Journal of Educational Psychology*, 46:128–148, 1976.

[PN05] D. Petrelli and E. Not. User-centred design of flexible hypermedia for a mobile guide: Reflections on the hyperaudio experience. *User Model. User-Adapt. Interact*, 15(3-4):303–338, 2005.

[RTA05] D. Raptis, N. Tselios, and N. Avouris. Context-based design of mobile applications for museums: a survey of existing practices. In *Proc. of 7-th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 153–160, 2005.

[SBB04] O. C. Santos, C. Barrera, and J. Boticario. An overview of Alfanet: An adaptive iLMS based on standards. In *Proc. of International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2004)*, number 3137 in Lecture Notes in Computer Science, pages 429–432. Springer Berlin/Heidelberg, 2004.

[SC88] S. Siegel and N. J. Castellan. *Nonparametric statistics for the behavioral sciences.* McGraw-Hill Book Company, New York, USA, 1988.

[SCGM08] E. Sangineto, N. Capuano, M. Gaeta, and A. Micarelli. Adaptive course generation through learning styles representation. *Universal Access in the Information Society*, 7(1):1–23, 2008.

[SS98] R. Sison and M. Shimura. Student modeling and machine learning. *Int. J. of Artificial Intelligence in Education*, 9:128–158, 1998.

[ST03] A. Sterbini and M. Temperini. A logical framework for course configuration in elearning. In *Proc. of ITHET03*, July 2003.

[SZB$^+$08] O. Stock, M. Zancanaro, P. Busetta, C.B. Callaway, A. Krüger, M. Kruppa, T. Kuflik, E. Not, and C. Rocchi. Adaptive, intelligent presentation of information for the museum visitor in PEACH. *User Model. User-Adapt. Interact*, 17(3):257–304, 2008.

[TH05] B. Towle and M. Halm. Designing adaptive learning environments with learning design. In R. Koper and C. Tattersall, editors, *Learning Design. A Handbook on Modelling and Delivering Networked Education and Training*, pages 215–226. Springer, 2005.

[TV07] M. Temperini and U. Vitti. A web application for automated course configuration. In *Proc. of ITHET 2007*, pages 536–540, 2007.

[Vas92] J. Vassileva. Dynamic CAL-courseware generation within an ITS-shell architecture. In *Proc. of the 4th International Conference on Computer Assisted Learning (ICCAL ’92)*, volume 602 of *Lecture Notes in Computer Science*, pages 581–591. Springer-Verlag, 1992.

[WASR07] Y. Wang, L. Aroyo, N. Stash, and L. Rutledge. Interactive user modeling for personalized access to museum collections: The Rijksmuseum case study. In *Proc. of 11-th International Conference on User Modeling (UM 2007)*, number 4511 in Lecture Notes in Computer Science, pages 385–389. Springer, 2007.

[WB01] G. Weber and P. Brusilovsky. ELM-ART: An adaptive versatile system for web-based instruction. *International Journal of Artificial Intelligence in Education*, 12:351–384, 2001.

[Wil47] F. Wilcoxon. Probability tables for individual comparisons by ranking methods. *Biometrics*, 3:119–122, 1947.

[WKW01] G. Weber, H. C. Kuhl, and S. Weibelzahl. Developing adaptive internet based courses with the authoring system NetCoach. In *Proc. of Third workshop on Adaptive Hypertext and Hypermedia*, pages 35–48, 2001.

[WL01] S. Weibelzahl and C. U. Lauer. Framework for the evaluation of adaptive CBR-systems. In Ivo Vollrath, Sascha Schmitt, and Ulrich Reimer, editors, *Experience Management as Reuse of Knowledge. Proceedings of the 9th German Workshop on Case Based Reasoning, GWCBR2001*, pages 254–263, Baden-Baden, Germany, 2001.

[Wol85] P. Wolper. The tableau method for temporal logic: an overview. *Journal of Logique et Analyse.*, 28:119152, 1985.

[Wu02] H. Wu. *A Reference Architecture for Adaptive Hypermedia Applications.* PhD thesis, Eindhoven University of Technology, 2002.

[ZW01] M. S. Zywno and J. K. Waalen. The effect of hypermedia instruction on achievement and attitudes of students with different learning styles. In *Proc. Annual ASEE Conf*, 2001.

[ZYB08] V. Zadorozhny, M. Yudelson, and P. Brusilovsky. A framework for performance evaluation of user modeling servers for web applications. *Web Intelli. and Agent Sys.*, 6(2):175–191, 2008.