Interactive Learning Environments
Vol. 00, No. 00, Month 2009, 1–34

RESEARCH ARTICLE

Constraint Modeling for Curriculum Planning and Validation

Matteo Baldoni[a*], Cristina Baroglio[a], Ingo Brunkhorst[b],
Nicola Henze[b], Elisa Marengo[a], and Viviana Patti[a]

[a]*Dipartimento di Informatica — Università degli Studi di Torino, Italy*;
[b]*L3S Research Center, University of Hannover, Germany*

(*Received 30 April 2008; final version received 22 June 2009*)

Curricula authoring is a complex process, involving different actors and different kinds of knowledge. Learners aim at acquiring expertise about some topic of their own interest, and need to perceive that the curriculum they attend will lead them towards their goal; when this does not happen, they become demotivated. Learners are all different, not only in their aims but also in their background knowledge and skills; curricula must carefully be tailored to the learner's individual traits: when this does not happen, curricula are not effective from a pedagogical perspective. On the other hand, it is not possible to leave learners alone in the design of a curriculum because this activity involves both knowledge about the topics to teach, and knowledge about teaching itself. It is one of the tasks of the school to support curricula authoring so to guarantee the correctness of the result w.r.t. the teaching goals and to pedagogical strategies. In this article we face the problem of authoring personalized curricula and propose a modular, layered architecture that accounts for the representation of learning resources, of the domain model, of the learner, and of pedagogical constraints, with the aim of supporting different validation tasks. The representation combines a Semantic Web approach to annotation with a declarative representation in linear temporal logic. The validation layer of the proposed architecture includes different kinds of inter-conceptual, post-construction verifications, all of which can be realized by means of model checking techniques. The article also reports about a prototype implementation based on the Personal Reader for education, a framework that supplies to its users personalization functionalities implemented in a service-oriented fashion.

Keywords: Semantic technologies for personalization, Semantic-rich service-oriented architectures, Semantic-enhanced learning designs, Linear Time Logic and Model Checking Techniques, Curriculum Sequencing and Validation

1 Introduction

The birth of the Semantic Web (Berners-Lee, 2000, 2002) brought along standard models, languages and tools for representing and dealing with machine-interpretable semantic descriptions of Web resources. The introduction of machine-processable semantics widens the range of the applicable reasoning techniques, giving a strong new impulse to research on personalization. By exploring various representation facets and reasoning techniques it becomes, in fact, feasible to select and deliver contents in a way that is tailored on the specific user.

An interesting research issue for studies on personalization, that requires the development of original solutions, is *curriculum sequencing* (Vassileva, 1992; Henze & Nejdl, 2001; Conlan et al., 2002; Brusilovsky & Vassileva, 2003; Baldoni et al., 2004a), where the term *curriculum* denotes an integrated course of academic studies. Well-designed curricula must both be attractive for students (because they are appealing and challenging) and, at the same time, respect the pedagogical dictates. Usually, however, there is a gap between the learners and the educational offer of schools (Lennon & Maurer, 2003), which is pedagogically sound but not tailored upon the learners' skill and desires, which depend on their previous studies and individual abilities (Samples, 2002; Melia & Pahl, 2007b). On the one hand, the learner needs to perceive that his/her studies will allow him/her to achieve the desired competencies,

*Corresponding author. Email: baldoni@di.unito.it

ISSN: 1049-4820 print/ISSN 1744-5191 online
© 2009 Taylor & Francis
DOI: 10.1080/1049482YYxxxxxxxx
http://www.informaworld.com

otherwise he/she will be *demotivated*. For this reason it is interesting to develop tools for *the design of personalized curricula*, targeted at the acquisition of specific learner's goals. On the other hand, one of the school's functions is to guarantee that *proper pedagogical strategies* are applied in order to make the acquisition of new expertise as smooth and easy as possible (Lennon & Maurer, 2003). The validation of curricula is a difficult and time-consuming task, for this reason it is useful to have *validation tools* that automatically check the learners' curricula against the school pedagogical constraints. The possibility to personalize and to automatically validate curricula reduces the gap between the learner's desires and the school's educational offer.

The automatic support to both these tasks becomes more important, and more difficult, when one considers open scenarios, where the set of learning resources is unrestricted and unknown in advance and resources themselves are *heterogeneous* in their nature. Let us, for instance, consider the *Bologna Process* (European Commission, Education and Training, 1999), promoted by the EU. The Bologna process objective is to foster the mobility of students. The final aim is to give students the possibility of attending *integrated curricula*, whose courses are scattered among different European institutions. The advantage is that, by exploiting this possibility, students will also be enabled to acquire competences which can only be achieved by composing some complementary knowledges that are supplied only by different academic institutes. On the other hand, often learners travel only "virtually" by retrieving and using learning objects made available by different repositories. Also in this case the term "curriculum" is often used, meaning a reading sequence of relevant learning objects. Learners get involved in a mobility program (or in an on-line curriculum) because they are interested in acquiring some desired expertise (the student's *learning goal*); not only they are different in their goals but they also come from different schools and have different backgrounds and skills, and their studies will be effective only if all these ingredients are taken into account. As well as learners, also University courses are different and in various ways even when they teach the same topics. Their heterogeneity includes the level of detail at which topics are taught, and the modalities by which courses are erogated. Learning resources (often called learning objects or media objects) add a further level of heterogeneity. For instance, they can be supplied as any kind of *digital media* (e.g. textual documents as well as videos or interactive applications), they can have *different granularity* (single lessons as well as courses) (Conlan et al., 2002), and they can be gathered from *different repositories*. Such differencies should not be an obstacle when the aim is to understand if a resource is useful in acquiring some *knowledge*. To this aim, it is very important to focus on the knowledge that is delivered and on the knowledge that is required for understanding the delivered concepts.

In order to fill the gap between the learner's desires and the school's pedagogical requirements, it is necessary to develop representations and tools that support learners and schools in practice and, in particular, the means for designing curricula (either offered or desired) that suit learners individually, verifying that they are pedagogically sound. The same representations and tools should be usable to construct or verify personalized reading sequences of heterogeneous learning objects. As such, these representations and tools would achieve a double aim. On the one hand, they would allow to handle learning resources in a homogeneous way, independently from their nature (a course would be just a kind of learning resource). On the other hand, they would meet the requirements given by organizations like ADLNet and IMS (working on learning object specification and standards), which identified *reusability*, *discoverability*, and *interoperability* as some of the main properties that should be granted by such tools. To this purpose Semantic Web comes in handy because it supplies the means for describing courses (more in general, learning resources) by means of *meta-data* (Learning Technology Standards Committee, 2002), i.e. machine-interpretable semantic annotations. In particular, Semantic Web offers standard models and languages for sharing knowledge (Stojanovic et al., 2001; Mohan & Brooks, 2003; W3C, 2004a,b; Antoniou & van Harmelen, 2004). The representations based on Semantic Web languages allow the application of different kinds of automatic reasoning techniques, mostly deriving from studies on Artificial Intelligence: from ontological reasoning (Antoniou & van Harmelen, 2004) to techniques that lay at the logic and proof layers of the Semantic Web tower (Antoniou et al., 2007), and that exploit rule-based representations (Schaffert & Bry, 2002; Paschke & Biletskiy, 2007).

In this article we present the results of a work (carried on in the context of the REWERSE network of excellence (REWERSE, 2008)) that concerns the representation of learning resources, of the domain model, and of pedagogical constraints so as to allow the design of personalized curricula, that are sound w.r.t. the pedagogical constraints given by a school. We call this activity *curricula authoring* and propose a *concept-based* representation of the educational *content* of resources, showing the reasoning tasks that can be performed on top of it. The approach is *cross-repository* and *cross-media*, i.e. verifications can be applied to curricula made of learning resources that are supplied by different sources and through different media. Its modularity allows easy extensions and integrations with further representations and tasks that are either source specific or media related, so to produce more complete implementations. For instance, one could think of integrations that include media processing workflows, where streams produced by different media objects are composed or filtered, e.g. by using the ARIA middleware (Candan et al., 2006; Brunkhorst et al., 2008), or to multi-media synchronization mechanisms, e.g. by adopting object composition Petri net approches (Little & Ghafoor, 1990; Li et al., 1994).

Curricula authoring is a complex process that involves different actors and different kinds of knowledge. Along the lines of Melia & Pahl (2007a,b), in order to simplify curricula authoring we structure this process into a layered architecture (see Figure 1). The adoption of a structured view has the advantage of modularizing the process, making the dependencies between different kinds of knowledge and of tasks clear and explicit. In this way, the upgrades (modifications) become simpler, because they affect single modules, and the effects of such modifications on the overall process can be evaluated more easily. The proposed approach is *general*, in the sense that it allows handling heterogeneous resources in a *uniform way*, independently from their nature and source.

The article is organized as follows. Section 2 gives an overview of our proposal by presenting an architecture for curricula authoring and by introducing our implementations of the various layers. The subsequent sections present in details such realizations. In particular, Section 3 presents a constraint-based representation of curricula models, and the DCML language, including both the graphical and the LTL characterizations. Further constraints that can be used to enrich curricula models are reported in Section 7. Section 4 shows how curricula are represented by means of UML activity diagrams, reporting some examples. Section 5 describes in details the validation layer and the implementation of the validation tasks, with the help of some Promela code. A *prototype implementation* is described in Section 6. This implementation relies on the Personal Reader (PR) framework (Henze & Krause, 2006), a service-oriented framework that supplies to the final user web services for dealing with semantic information. The Personal Reader framework also allows for the combination of services into new personalization functionalities by exploiting the concept of *syndication*. A comparison with relevant works in the literature and some conclusive remarks end the article.

2 Curricula authoring

In order to introduce the aims of our work in an intuitive way, let us consider an Italian student, who needs to attend some courses in Germany: nowadays the student would be guided by an Italian mentor in the definition of a curriculum, which combines the student's interests, the educational goals of the Italian school, the courses offered by the same institute, those offered by the German hosting institute, and also the guidelines and constraints posed by the latter. Quite a complex and time-consuming task (Brady et al., 2008). The work that we propose aims at making the execution of this kind of tasks *automatic*. Along the lines of Melia & Pahl (2007a,b), we structure curricula authoring in a set of representational and activity layers (see Figure 1), that we describe one by one hereafter.

Domain Model

The base level is the *domain model*. This layer contains the knowledge about the domain itself expressed as a set of concepts and their relations, independently from any pedagogical concern. Concepts can, for instance, represent the various topics and their organization in sub-topics. Concepts can easily be given a semantic representation by using the tools and the languages supplied by the Semantic Web community. For instance, they could be expressed as part of an ontology (or a taxonomy). The domain model used in this work (see Section 6.1), is a vocabulary of RDF terms, that have been extracted automatically

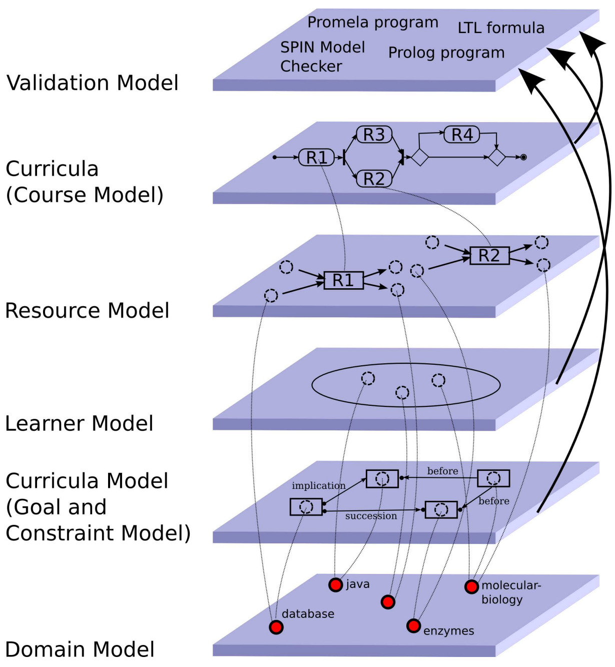


Figure 1. The layered architecture.

by means of the Lixto tool (Baumgartner et al., 2005), applied to the descriptions of the courses held at the University of Hannover, that are supplied by HIS-LSF (http://www.his.de). So far, Lixto returns a flat RDF vocabulary. In the future, it might be interesting to try to derive an ontology from the corpus by structuring the concepts (Pazzi, 1998; Antoniou et al., 2005), e.g. by using Levenshtein distances and a thesaurus, like wordnet. Focussing on a flat vocabulary is not restrictive because, due to the adoption of a modular (layered) architecture, one can think to integrate ontological reasoning in the domain model layer and, as a consequence, to have it included also in the higher levels.

In the literature concerning professional curricula and e-learning, concepts are referred to either as *competency* (plural “competencies”) or as *competence* (plural “competences”), to denote respectively “any form of knowledge, skill, attitude, ability or learning objective that can be described in a context of learning, education or training” and the “effective performance within a domain at some level of proficiency” (De Coi et al., 2007). Our domain model contains competencies. Different courses can supply/require the same competency at different proficiency levels. For the sake of readability, in this article we do not use RDF to represent competences; instead, we use a more intuitive notation of the kind $(database, beginner)$, where the first element denotes a concept while second a proficiency level.

Curricula Model (Goal and Constraint Model)

The subsequent layer (Figure 1) is the *curricula model*, which contains pedagogical constraints defined over the domain model. The curricula model includes a *set of constraints* on competences (i.e. concepts plus a proficiency level). We partition constraints into a set of requirements, a set of goals, and a set of relational constraints. The requirements specify the expected competences that a learner should have before starting the learning process. The goals specify the knowledge that is to be delivered. The relational constraints structure the teaching activity of the various concepts in the domain model, by imposing orderings or presence. Intuitively, they specify the pedagogical constraints – given by some instructional designer –, that rule the order in which concepts are to be taught/learnt. We have identified three kinds of relational constraint (and their negations): *before*, *implication*, and *succession* (Figure 4). The first kind of constraint expresses the fact that in order to acquire a competence, another competence must be owned before. Implication constraints express the fact that if at some point a competence is acquired, also some other competence must already be owned by the learner or it must be learnt in the future. Finally, succession constraints express the fact that when a competence is acquired, then another is to be be acquired in the future. The expressiveness of the proposed representation makes curricula models not only sets of precedence constraints, as done in previous work (e.g. Melia & Pahl (2007a,b)), but it makes them rich schemas – see Figure 2 –, each of which defines a whole class of curricula.

In order to represent curricula models we propose the *Declarative Curricula Modeling Language* (DCML for short, see Section 3). DCML is a graphical language (Figure 2 shows an example), inspired by DecSerFlow, the Declarative Service Flow Language to specify, enact, and monitor web service flows by van der Aalst and Pesic (van der Aalst & Pesic, 2006). DCML, as well as DecSerFlow, is grounded in *Linear Temporal Logic* (Emerson, 1990) and allows a curricula model to be described in an easy way maintaining at the same time a rigorous and unambiguous meaning given by the logic representation. The use of a logic with a clear semantics allows the separation of the representation of constraints from the choice of an algorithm to perform validation tasks. As an effect the curricula model layer and the validation layer result more independent. Moreover, it is possible to detect inconsistencies in the curricula model (e.g. loops in the dependencies), by the application of already existing algorithms.

Learner Model

The *learner model* represents the knowledge of a learner, which evolves during the learning process. Intuitively, by attending a course the learner will gain the knowledge taught in that course. Before starting the learning process, this representation is set to the initial expertise of the learner. So, the learner model contains a representation of the mental state of the learner. The expectation of the learner is that its mental state will evolve so to include some desired expertise, i.e. so to reach a specific learning goal. Also the learning goal is included in the learner model. The learner's knowledge and learning goals are expressed as sets of competences.

Resource Model

The *resource model* describes courses (learning resources), in terms taken from the domain model. Specifically, each course is described in an *action-based* fashion (Baldoni et al., 2002, 2004a,b, 2005, 2006b), by its preconditions and effects. Both preconditions and effects are *competences*. As such, each of them is a pair that combines a competency (any form of knowledge, skill, attitude, ability or learning objective that can be described in a context of learning, education or training) with a proficiency level. This representation is general because, although in this paper we deal with courses, it expresses an information that lays at the knowledge level and, for this reason, it can be used to tackle any kind of learning resource. As an example of resource representation, let us consider a course with name *Biology_II*, which requires beginner's knowledge about *protein_structures* and some deeper (intermediate level) knowledge about *cell_structure*. The course supplies advanced knowledge about *enzymes* and about *molecular_biology*, and intermediate knowledge about *immune_systems*. It will be represented as follows:

resource_name: Biology_II
  preconditions: (cell_structure, intermediate), (protein_structures, beginner)
  effects: (enzymes, advanced), (immune_systems, intermediate), (molecular_biology, advanced)

Preconditions are evaluated against the representation of the learner's mental state, that is contained in the learner model, while effects modify this representation, causing an evolution of the learner's knowledge.

In an open environment, it is likely that resources belonging to different schools are annotated by using different domain models, which are, therefore, to be combined in some way in order to obtain a uniform representation that allows the application of reasoning techniques (Hackelbusch & Appelrath, 2008). Supposing that domain models are given as domain ontologies, it becomes possible to apply techniques for ontology merging, alignment and integration, e.g. Euzenat & Shvaiko (2007). Indeed, these are hot research topics, that we do not tackle in this work. We make the hypothesis that this alignment already occurred and that all resources are represented in terms of a common ontology.

Curriculum (Course Model)

Curricula (see Figure 5 for an example) are represented by means of UML activity diagrams (OMG, 2007), which compose resources, whose representation is supplied by the previous layer. In the simplest case, a curriculum can be seen as a sequence of actions that causes *transitions* from the initial set of competences (possibly empty) of a user up to a final state that will contain also the acquired competences. We assume that concepts can only be added to states and competence level can only grow by executing the actions of attending courses (or more in general reading a learning material). The intuition behind this assumption is that new courses do not erase the concepts acquired previously, thus knowledge grows incrementally. Generally speaking, a curriculum may contain one or several learning pathways to be attended, in alternative or as obligations.

The use of UML activity diagrams seems very natural for representing curricula, and they are used also in other works to this purpose, e.g. Melia & Pahl (2007b). As a difference, we found two principles very useful when representing a curriculum (Section 4): to carefully distinguish courses with distinct duration (in time); to carefully distinguish mandatory courses and optional courses. Modelling a curriculum with these two principles in mind introduces *(i)* a decomposition level and *(ii)* partitions among courses, being these courses from mandatory or from optional partitions.

Validation Model

Finally, we have the *validation model* defines the *pedagogical strategies* that are to be enforced. According to the CAVIAr model (Melia & Pahl, 2007a,b), pedagogical validation can be intra-conceptual (ensuring that each concept is taught in a uniformed manner), inter-conceptual (concerning how the curriculum proceeds from one concept to another), or it can be done w.r.t. pedagogical rules which specify the types of resource that are involved (e.g. resources of type “lecture”). With reference to this classification, the kinds of validation that we perform are *inter-conceptual*. In particular, for what concerns curricula authoring three are the main checks to perform (Brusilovsky & Vassileva, 2003): (a) verify that the curriculum brings to the acquisition of the desired knowledge; (b) verify that at every point the learner already has the knowledge, that is necessary for understanding the next taught concept; (c) verify that the curriculum respects the goal and constraint model. Task (a) allows checking whether a curriculum brings to the achievement of the declared user’s *learning goal*. In this case, the verification process uses the learner model (that, in turn, uses the domain model) and the course model (that, in turn, uses the resource model, which uses the domain model). Task (b) consists in checking that the curriculum shows no *competence gaps* (De Coi et al., 2007), i.e. the knowledge that is necessary to fully understand a course is introduced or available before it is attended. In this case, the verification process uses the course model, which uses the resource model, which, in turn, uses the domain model. Task (c) verifies the compliance of a given curriculum against the guidelines (pedagogical strategies) given by the institution (Brusilovsky & Vassileva, 2003), a tedious task that is currently performed manually. It uses the course model, and the curricula model, and verifies that the curriculum represented by the course model satisfies all the constraints imposed by the other model.

To perform all these tasks, we use *model checking techniques* (Clarke & Peled, 2001) (Section 5). By means of a *model checker*, it is possible to generate and analyze all the possible states of a model (represented as an automaton) exhaustively, to verify that all execution paths satisfy a certain property, usually expressed by a temporal logic, such as LTL. In our case, the model is a curriculum (i.e. it is the course model), suitably translated into an automaton. The LTL formula is, instead, obtained in different ways depending on the task to accomplish. For Task (a), it is obtained by translating the learning goal (which is part of the learner model) into a set of assertions that are to be satisfied by the final states of the automaton. For Task (b), the resource model is translated into a set of assertions that are to be satisfied by the states in the automaton. For Task (c), it is the goal and constraint model, which is translated into a set of LTL formulas that the automaton must satisfy.

When a model checker finds that a model does not satisfy some LTL formula (or assertion), it produces a *counterexample* that shows the violation. Even though it is possible to argue that model checking techniques are not efficient because they analyze the whole space of possible executions, the fact of having counterexamples is extremely important because it supplies an *explanation* of the failure, that can be supplied as feedback for the user. It is well-known that the user will not be confident in the answer unless he/she can trust the *reasons* why the answer was produced. As a model checker, in this work we have used SPIN, by G. J. Holzmann (Holzmann, 2003), which is the most representative tool of this kind.

Implementation and planning

All the kinds of validation mentioned above are post-construction course validations (Persico, 1996; Rosmalen et al., 2006), that can be used to produce personalization functionalities. To this aim, we have developed a prototype implementation (Section 6), based on the PR (Personal Reader) framework (Henze & Krause, 2006) – see Figure 7 for an overview –. The PR relies on a *service-oriented architecture* enabling personalization, via the use of *Personalization Services*, that communicate solely based on RDF documents. Each service offers a different personalization functionality, e.g. recommendations tailored to the needs of specific users, pointers to related (or interesting or more detailed/general) information, and so on. Besides the verification functionalities, the PR offers a service for automatically composing a curriculum, based on the learner model and on the resource model, that exploits automatic planning techniques (Russell & Norvig, 2003). In fact, by interpreting resources as actions, a curriculum can, in turn, be interpreted as a *plan*, whose execution causes transitions from a state to another, until some final state is reached. A transition between two states is due to the application of an action, that corresponds to attending a course. To apply an action, its preconditions must hold in the state to which it is applied, and its application consists in an *update* of the state. We assume that facts can only be added to states. The intuition behind this assumption is that the act of attending a course (or, more in general, of using a learning resource) will not erase from the learner's memory the concepts acquired so far. The service that we have developed, that is described in details in Section 6.3, is specifically aimed at building *linear curricula*, i.e. curricula which consist of a sequence of courses only. Such curricula can be tailored to a specific user by starting from an initial state that contains *that* user's knowledge. This perspective is taken in several works that face the so called *curriculum sequencing* problem (Vassileva, 1992; Henze & Nejdl, 2001; Brusilovsky & Vassileva, 2003), that is the problem of generating a personalized learning path for each student, by dynamically selecting at any moment a resource that, in the context of other available resources, brings the student closer to the learning goal. In particular, our representation relies on a semantic annotation supplied by the resource model and based on the domain model.

3 Curricula Models and DCML

In order to present *curricula models*, we propose a *constraint-based representation* (Marengo, 2008). The instructional designer can express various kinds of constraints by means of a simple graphical language, that we call DCML (*Declarative Curricula Model Language*). DCML constraints can be automatically turned into formulas in a temporal logic (LTL, linear temporal logic (Emerson, 1990)). This logic allows the verification of properties of interest for all the possible executions of a model, which in our case corresponds to a curriculum. LTL includes temporal operators such as next-time ($\bigcirc\varphi$, the formula φ holds in the immediately following state of the run), eventually ($\Diamond\varphi$, φ is guaranteed to eventually become true), always ($\Box\varphi$, the formula φ remains invariably true throughout a run), until ($\alpha \cup \beta$, the formula α remains true until β), see also Holzmann (2003, Chapter 6). The adoption of a *graphical* language with a *logical* grounding allows designers, who cannot be expected to feel comfortable with the logical notation, to take anyway advantage of automatic verification tools for accomplishing the tasks foreseen by the validation model. On the other hand, relying on a logic with a rigorous semantics allows a separation of concerns between the constraint model and the choice of an algorithm (tool) for performing the validation tasks. Constraints have a declarative semantics which is independent from the implementation.

Constraint-based representations are very compact because they require to express only those *necessary conditions* that characterize a correct curriculum, avoiding overspecification. To make an example, by means of constraints we can request that a certain knowledge is acquired before some other knowledge, without expressing what else is to be done in between. This is an advantage w.r.t. other approaches, like the procedural approach (e.g. Baldoni et al. (2004a)), having a *prescriptive* nature because in the latter it is necessary to express exhaustively all the pathways that are correct. If, on the one hand, it makes sense that some topics have a specific ordering, it is also intuitive that this does not happen for all the involved competences. For example, it makes sense that the competence $(database, beginner)$ (knowl-

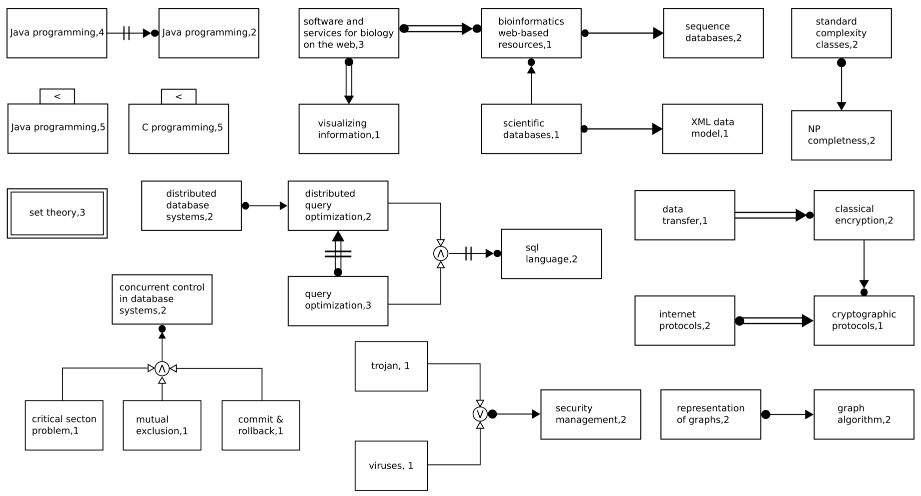


Figure 2. An example of curricula model in DCML.

edge about databases with proficiency level "beginner") is to be acquired before $(database, advanced)$ but what about a competence $(English, advanced)$? Should it be acquired before or after the other two mentioned competences? Intuitively, it is not necessary to impose that $(English, advanced)$ is, for instance, the first competence to acquire because it is not related to the other topics. The problem is that an ordering that is not explicitly mentioned in the model is not legal, so the designer must foresee many variants within the schema. By using constraints, in the previous case, we only need to express that $(database, beginner)$ must be acquired before $(database, advanced)$ and that $(English, beginner)$ is to be acquired sooner or later. For this reason, we can say that the constraints-based approach is more flexible and more suitable in an open environment, due to the fact that in an open environment resources are many, they are various, and they are added and removed dynamically.

As an example, Figure 2 shows a curricula model expressed in DCML (notice that double arrows are defined in Section 7). Every box contains at least one competence. Boxes/competences are connected by arrows, which represent (mainly) temporal constraints among the times at which they are to be acquired. Altogether the constraints describe a curricula model. Let us now introduce DCML in details.

3.1 *Expressing competences and basic constraints*

DCML is used to define a set of constraints on top of the domain model. Competencies are associated to a representation of the *proficiency level* at which a competency is owned or to be supplied. So, for instance, we express the fact that a competency $database$ is to be owned at level $beginner$. To simplify the management of proficiency levels, such levels are represented as numerical values. To this aim, we associate to each competency a variable k, having the same name as the competency, which can be assigned natural numbers as values. The value of *k* denotes the proficiency level; zero means absence of knowledge. Therefore, k encodes a *competence*, Figure 3. On top of competences, in DCML it is possible to define two basic *constraints*:

- *Goal constraints*: A goal constraint – Figure 3, third row – imposes that a certain competency k must be acquired at least at level l. It is represented by the LTL formula $\Diamond(k \geq l)$. Similarly, a course designer can impose that a competency must never appear in a curriculum with a proficiency level higher than l. This is possible by means of the "*always less than level*" constraint, shown in Figure 3 fourth row. The LTL formula $\Box(k < l)$ expresses this fact (it is the negation of the previous one). As a special case, when the level l is one ($\Box(k < 1)$), the competency k must never appear in a curriculum.
- *Requirements*: A requirement, represented by a double box – see Figure 3, second row – specifies that before starting the learning process, the learner must own the competence k at least at level l. This is represented by the logic formula $(k \geq l)$.

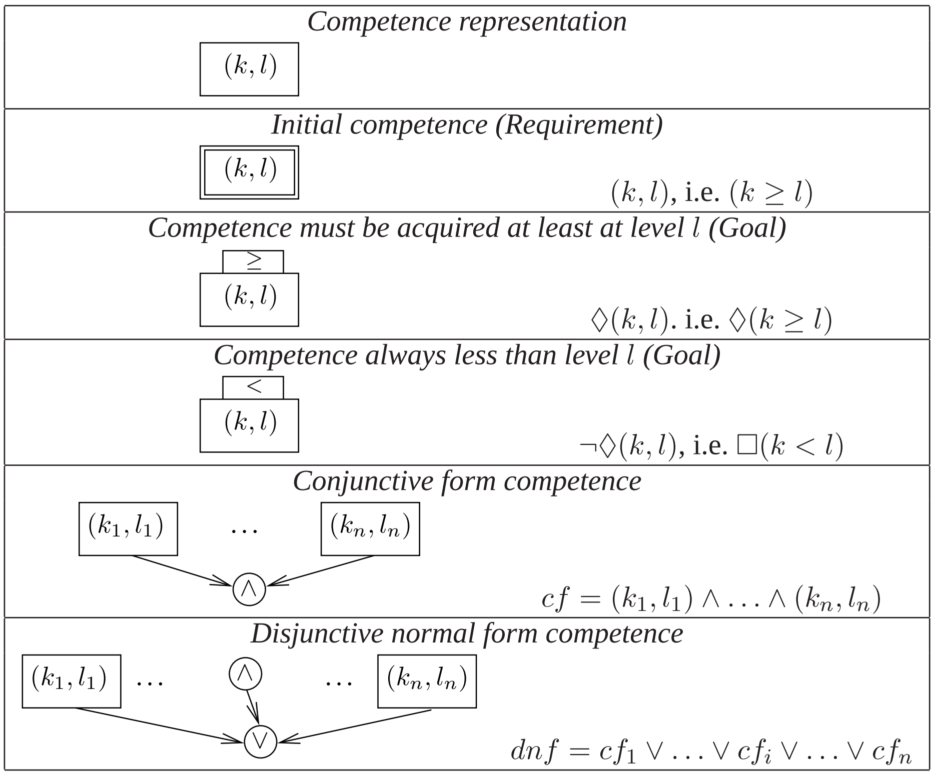

Competence representation
(k, l)
Initial competence (Requirement)
(k, l)
(k, l), i.e. $(k \geq l)$
Competence must be acquired at least at level l (Goal)
$\geq$
(k, l)
$\Diamond(k, l)$. i.e. $\Diamond(k \geq l)$
Competence always less than level l (Goal)
$<$
(k, l)
$\neg\Diamond(k, l)$, i.e. $\Box(k < l)$
Conjunctive form competence
(k_1, l_1) … (k_n, l_n)
$\wedge$
$cf = (k_1, l_1) \wedge \ldots \wedge (k_n, l_n)$
Disjunctive normal form competence
(k_1, l_1) … $\wedge$ … (k_n, l_n)
$\vee$
$dnf = cf_1 \vee \ldots \vee cf_i \vee \ldots \vee cf_n$

Figure 3. DCML representation of competences, conjunctions and disjunctions of competences.

In DCML it is possible to represent *Disjunctive Normal Form* (DNF) formulas as *conjunctions* and *disjunctions* of competences[1]. Graphically, a conjunction of basic constraints is represented by a circle with a "$\wedge$" symbol inside, having as many incoming arrows as conjuncts. A disjunction, instead, can compose both basic constraints and conjuncts of constraints. Graphically – Figure 3, last two rows – it is represented by a circle with a "$\vee$" symbol inside, the disjuncts are connected to it by means of arrows.

Let k be a competence, we denote by (k, l) the constraint $k \geq l$ and by $\neg(k, l)$ the constraint $k < l$. A *conjunctive competence formula* cf is a conjunction of atomic competence constraints $cf = (k_1, l_1) \wedge \cdots \wedge (k_n, l_n)$. A conjunction can also be interpreted as the set of constraints $cf = \{(k_1, l_1), \ldots, (k_n, l_n)\}$. To help the expression of LTL formulas we introduce the functions:

- $\mathsf{negation}(cf) = \bigwedge_{(k_i,l_i) \in cf} \neg(k_i, l_i)$;
- $\mathsf{existence}(cf) = \bigwedge_{(k_i,l_i) \in cf} \Diamond(k_i, l_i)$;
- $\mathsf{absence}(cf) = \bigwedge_{(k_i,l_i) \in cf} \Box\neg(k_i, l_i)$.

A *disjunctive normal competence formula* dnf is a disjunction of conjunctive competence formulas, $dnf = cf_1 \vee \cdots \vee cf_n$. Again, we also denote a disjunctive normal competence formula as a set of conjunctive competence formulas $dnf = \{cf_1, \ldots, cf_n\}$. Therefore, a disjunctive normal competence formula is a set of sets of atomic competences.

3.2 *Constraints among competences*

Besides the representation of competences and of constraints on competences, DCML allows the representation of *relations* among competences. These relations are represented as different kinds of arrows. It is also possible to represent "negative relations" by using two vertical lines to break the arrow that represents the constraint.

There are three main kinds of relation:

[1]A DNF formula is an expression having the form $(a_{1,1} \wedge a_{1,2} \wedge \ldots) \vee (a_{2,1} \wedge a_{2,2} \wedge \ldots) \vee \ldots$.

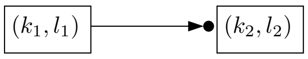
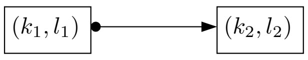
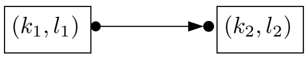
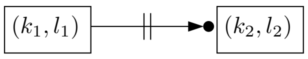
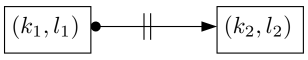
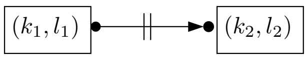
Before  $\neg(k_2, l_2)\ \mathsf{U}\ (k_1, l_1)$	(k_1, l_1) *before* (k_2, l_2) Competence (k_1, l_1) must be acquired before or at the same time of competence (k_2, l_2)
Implication  $\Diamond(k_1, l_1) \supset \Diamond(k_2, l_2)$	(k_1, l_1) *implies* (k_2, l_2) If competence (k_1, l_1) is acquired, then competence (k_2, l_2) must be acquired sooner or later
Succession  $\Diamond(k_1, l_1) \supset (\Diamond(k_2, l_2) \wedge (\neg(k_2, l_2)\ \mathsf{U}\ (k_1, l_1)))$	(k_2, l_2) *succeeds* (k_1, l_1) If competence (k_1, l_1) is acquired, then competence (k_2, l_2) must be acquired after or at the same time
Negative before  $\neg(k_1, l_1)\ \mathsf{U}\ ((k_2, l_2) \wedge \neg(k_1, l_1))$	(k_1, l_1) *not before* (k_2, l_2) Competence (k_1, l_1) cannot be acquired before or at the same time of (k_2, l_2)
Negative implication  $\Diamond(k_1, l_1) \supset \Box\neg(k_2, l_2)$	(k_1, l_1) *not implies* (k_2, l_2) If competence (k_1, l_1) is acquired, then competence (k_2, l_2) cannot be acquired
Negative succession  $\Diamond(k_1, l_1) \supset (\Box\neg(k_2, l_2) \vee$ "(k_1, l_1) *not before* (k_2, l_2)")	(k_2, l_2) *not succeeds* (k_1, l_1) If competence (k_1, l_1) is acquired, then either competence (k_2, l_2) already holds when (k_1, l_1) is acquired or it is never achieved

Figure 4. DCML representation of basic constraints among competences. Arrows can connect not only competences but also conjunctions and disjunctions of competences, see Figure 2.

- *before*: this relation expresses the fact that in order to acquire a competence, another competence is to be owned in advance;
- *implication*: this relation means that if at some point a competence is acquired, also some other competence must already be owned by the learner or it must be learnt in the future;
- *succession*: means that when a competence is acquired, then another is to be be acquired in the future.

Before

Arrows ending with a little-ball, Figure 4, express the *before* temporal constraint between two competences. "(k_1, l_1) *before* (k_2, l_2)" requires that (k_1, l_1) holds *before* (k_2, l_2). This constraint can be used to express that to understand some topic (e.g., k_2 at least at the level l_2), some proficiency of another is required as precondition (in the example, k_1 at least at the level l_1). It is important to underline that if the antecedent never becomes true, also the consequent must be invariably false; this is expressed by the LTL formula $\neg(k_2, l_2)\ \mathsf{U}\ (k_1, l_1)$. More generally, in presence of DNF formulas as antecedent and consequence of a "before" relation, we have the following definition for "dnf_1 *before* dnf_2":

$$\bigvee_{cf_i \in dnf_1, cf_j \in dnf_2} \mathsf{negation}(cf_j)\ \mathsf{U}\ cf_i$$

With reference to Figure 2 – bottom-left side of the picture, the competences (*critical_section_problem*, 1), (*mutual_exclusion*, 1) and (*commit&rollback*, 1) are necessary for the user to understand the competence (*concurrent_control_in_db_systems*, 2). Hence, the curricula model contains a constraint: (*critical_section_problem*, 1) ∧ (*mutual_exclusion*, 1) ∧ (*commit&rollback*, 1) *before* (*concurrent_control_in_db_systems*, 2).

A constraint of kind "(k_1, l_1) *not before* (k_2, l_2)" specifies that k_1 cannot be acquired up to level l_1 before or in the same state when (k_2, l_2) is acquired. The corresponding LTL formula is $\neg(k_1, l_1)\ \mathsf{U}\ ((k_2, l_2) \wedge \neg(k_1, l_1))$. Notice that this is not obtained by simply negating the before relation but it is weaker; the negation of *before* would *impose the acquisition* of the concepts specified as consequents (in fact, the formula would contain a strong until instead of a weak until), the *not before* does not. More generally, in presence of DNF formulas, "dnf_1 *not before* dnf_2" is:

$$\bigvee_{cf_i \in dnf_1, cf_j \in dnf_2} \mathsf{negation}(cf_i)\ \mathsf{U}\ (cf_j \wedge \mathsf{negation}(cf_i))$$

As an example, the constraint (*distributed_query_optimization*, 2) $\wedge$ (*query_optimization*, 3) *not before* (*sql_language*, 2) represents the fact that competences (*distributed_query_optimization*, 2) and (*query_optimization*, 3) cannot be acquired before or in the same state in which the competence (*sql_language*, 2) is acquired. The reason could be that *distributed_query_optimization* and *query_optimization* are more difficult topics and learners need time to get acquainted with them.

Another interesting way to use this constraint is described by the following example: (*Java_programming*, 4) *not before* (*Java_programming*, 2). In this way we express that the competency *Java_programming* is too important (in a certain curricula model) and from a pedagogical point of view, it cannot be acquired immediately at advanced level. Instead, the student has to acquire it in two subsequent steps: intermediate (proficiency level 2) and then advanced (proficiency level 4 in the example).

Implication

Before relation represent temporal constraint between competences. The *implication* relation, for example "(k_1, l_1) *implies* (k_2, l_2)", denoted by arrows starting with a little-ball (see Figure 4), specifies, instead, that if a competency k_1 holds at least at the level l_1, some other competency k_2 must be acquired, *sooner* or *later*, at least at the level l_2. The main characteristic of the implication, is that the acquisition of the consequent is imposed by the truth value of the antecedent, but, in case this one is true, it does not specify when the consequent must be achieved (it could be before, after or in the same state of the antecedent). This is expressed by the LTL formula $\Diamond(k_1, l_1) \supset \Diamond(k_2, l_2)$. More generally, in presence of DNF formulas as antecedent and consequence of a "implication" relation, we have the following definition for "dnf_1 *implies* dnf_2":

$$\bigvee_{cf_i \in dnf_1, cf_j \in dnf_2} \mathsf{existence}(cf_i) \supset \mathsf{existence}(cf_j)$$

As an example, let us consider the constraint, Figure 2, (*distributed_database_systems*, 2) *implies* (*distributed_query_optimization*, 2). This constraint imposes that if the competence (*distributed_database_systems*, 2) is acquired, also (*distributed_query_optimization*, 2) must be acquired sooner or later. Generally, the implication constraint is useful to express that when a learner acquires a certain competence, for the sake of completeness, the learner should also have another competence; if he/she does not have it yet he/she will have to acquire it by the end of the curriculum. To have a better intuition, this is the case of the two algorithms *breadth_first_search* and *depth_first_search*. Of course, it is not necessary to know one of them in order to understand the other but with the implication we can impose that if one of them is acquired, also the other one must be owned by the student before the end of the curriculum: (*breadth_first_search*, *l1*) *implies* (*depth_first_search*, *l2*).

"(k_1, l_1) *not implies* (k_2, l_2)" expresses that if (k_1, l_1) is acquired k_2 cannot be acquired at level l_2; as an LTL formula: $\Diamond(k_1, l_1) \supset \Box\neg(k_2, l_2)$. Again, we choose to use a weaker formula than the natural negation of the implication relation because the simple negation of formulas, $\Diamond(k_1, l_1) \wedge \Box\neg(k_2, l_2)$, would impose the presence of certain concepts (k_1 at least at level l_1, in the example). More generally, in presence of DNF formulas, "dnf_1 *not implies* dnf_2" is:

$$\bigvee_{cf_i \in dnf_1, cf_j \in dnf_2} \mathsf{existence}(cf_i) \supset \mathsf{absence}(cf_j)$$

This kind of constraint is useful to express the fact that two competences A and B, considered as equivalent to one another, should be mutually exclusive. We can capture this by using the pair of constraints:

A *not implies* B, and B *not implies* A. This is analogous, though at the level of competences and not at the level of courses, to a kind of constraint proposed in Wu & Havens (2005), expressing the fact that equivalent courses are not to be used in a same curriculum.

Succession

The last constraint is *succession* (arrows starting and ending with a little-ball, Figure 4). "(k_2, l_2) *succeeds* (k_1, l_1)" specifies that if (k_1, l_1) is acquired, afterwards (k_2, l_2) is also achieved; otherwise, the level of k_2 is not important. This is a difference w.r.t. the *before* constraint where, when the antecedent is never acquired, the consequent must be invariably false. Indeed, the *succession* specifies a condition of the kind *if* $k_1 \geq l_1$ *then* $k_2 \geq l_2$, while *before* represents a constraint without any conditional premise. Instead, the fact that the consequent must be acquired after the antecedent is what differentiates *implication* from *succession*. Succession constraint is expressed by the LTL formula $\Diamond(k_1, l_1) \supset (\Diamond(k_2, l_2) \wedge (\neg(k_2, l_2)\ \mathsf{U}\ (k_1, l_1)))$. More generally, in presence of DNF formulas as antecedent and consequence of a "succession" relation, we have the following definition for "dnf_2 *succeeds* dnf_1":

$$\bigvee_{cf_i \in dnf_1, cf_j \in dnf_2} \mathsf{existence}(cf_i) \supset (\mathsf{existence}(cf_j) \wedge \text{“}cf_i \textit{ before } cf_j\text{”})$$

This constraint specifies an ordering on the acquisition of two competences. It can be used, for instance, to express a pedagogical constraint stating that after a learner has acquired some theoretical competence about a specific topic, the same learner must also acquire some practical skill about the same topic. An example, which is not included in Figure 2 is the following: (*Oracle_DBMS*, *l1*) *succeeds* (*DBMS*, *l2*). The constraint specifies that after the acquisition of knowledge about DBMS (at the proficiency level $l2$), also skills about the Oracle DBMS is to be acquired (at the proficiency level $l1$).

"(k_2, l_2) *not succeeds* (k_1, l_1)" imposes that a certain competence, (k_2, l_2), cannot be acquired after another, (k_1, l_1), either it was acquired before, or it will never be acquired. As LTL formula, it is $\Diamond(k_1, l_1) \supset (\Box\neg(k_2, l_2) \vee$ "(k_1, l_1) *not before* (k_2, l_2)"). Similarly to the previous negative constraints, we choose to use a weaker formula than the natural negation of the succession relation because the simple negation of formulas, $\Diamond(k_1, l_1) \wedge (\Box\neg(k_2, l_2) \vee \neg$ "(k_1, l_1) *before* (k_2, l_2)"), would impose the presence of certain concepts (k_1 at least at level l_1, in the example). More generally, in presence of DNF formulas, "dnf_2 *not succeeds* dnf_1" is:

$$\bigvee_{cf_i \in dnf_1, cf_j \in dnf_2} \mathsf{existence}(cf_i) \supset (\mathsf{absence}(cf_j) \vee \text{“}cf_i \textit{ not before } cf_j\text{”})$$

4 Representation of Curricula

In general, a curriculum may be represented with one or several learning paths to be attended, *optionally* or as an *obligation*. We represent curricula by means of UML *activity diagrams* (OMG, 2007; Marengo, 2008). The diagram captures essentially the "student personal process" to achieve the final degree. In the simplest case, a curriculum is a *sequence of courses*, that we represent as *activities* in the resource model (see Section 2, Resource Model), causing *transitions* from the initial set of competences (possibly empty) of a learner (included in the learner model) up to a final state that will contain the acquired competences. As mentioned, we assume that, as an effect of attending courses, competences are only added to states and that competence levels can only grow. Along the line of Baldoni et al. (2004a, 2005, 2006b), we interpret courses (activities) as actions, that can be executed given that their preconditions hold. By executing an activity, a set of post-conditions, the effects, will become true (the learner will acquire new competences).

Example 4.1 Let us consider again the example in Section 2:

resource_name: Biology_II

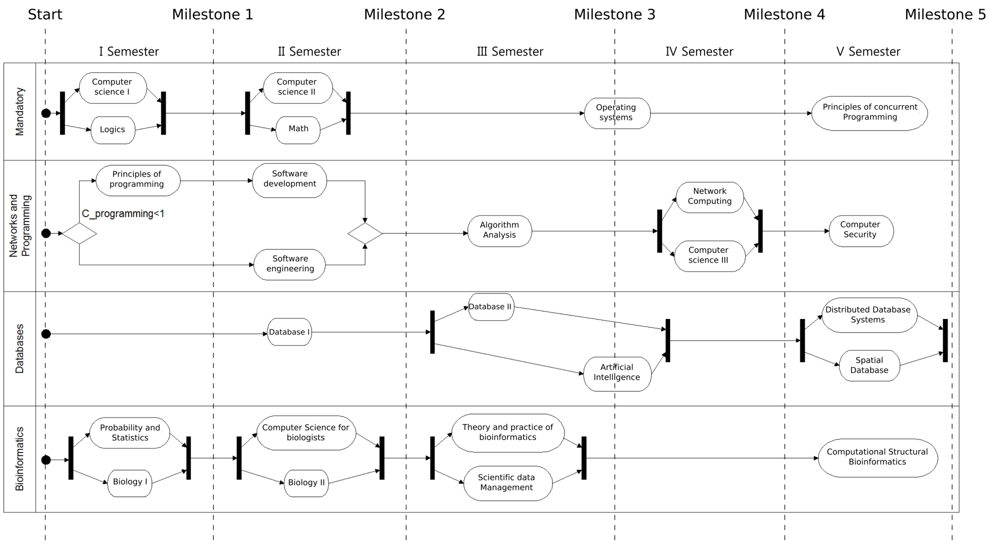


Figure 5. Activity diagram representing a curriculum with mandatory and additional, student chosen, courses. Swimlanes represent the sequencings of courses. Vertical divisions capture the different milestones (semesters).

preconditions: (cell_structure, intermediate), (protein_structures, beginner)
effects: (enzymes, advanced), (immune_systems, intermediate), (molecular_biology, advanced)

The activity *Biology_II* is executable at some point of a learning path, if at that point the learner has already acquired the competences (*cell_structure*, *intermediate*), (*protein_structures*, *beginner*). After the transition, caused by the execution of the activity *Biology_II*, the learner will be in a state that also includes the competences: (*enzymes*, *advanced*), (*immune_systems*, *intermediate*), (*molecular_biology*, *advanced*).

UML activity diagrams are well suited for representing curricula for many reasons. They may contain activities with pre- and post- conditions, combined in complex paths and possibly aggregated. Activity diagrams are rich enough to represent alternative, intermediate states and conditional paths. Moreover, they allow the distinction of courses with different duration (in time), the distinction of mandatory learning paths and additional (optional) learning paths, and the decomposition of an activity in sub-activities up to the desired granularity.

Figure 5 reports an example. The *horizontal partition* (*swimlanes* in UML) allows the distinction of mandatory and additional learning paths. With reference to the example, the curriculum includes three additional pathways (*Networks and Programming*, *Databases*, and *Bioinformatics*), that can alternatively be used to complement the mandatory pathway. *Vertical partitions* provide information about activities with different duration. In this case, we have used as time references the usual distinction, in years and semesters, implemented in universities. The beginning/ending points of the semesters correspond to a set of *milestones*: this temporal organization will be used to identify those states, at which the verification will be applied (see below for an example). In previous works (Melia & Pahl, 2007b; Baldoni et al., 2007b), instead, courses were *atemporal* and each state was tied to the simulation of a single course. The introduction of durations allows a more realistic representation of the curricula and, especially, of the dependencies between competences. Therefore, we can easily see that the course *Logics* (in the mandatory pathway) is delivered during the first semester, while the course *Operating Systems* (in the same pathway) is delivered in the third and fourth semesters.

The swimlanes that represent additional learning paths can be used to capture *one-time* choices of the learners. Due the semantics of UML activity diagrams, each learner has to choose one of the optional learning paths in order to complete its curriculum. However, as a particular case, it is possible to include an empty swimlane when the learner has the right of ignoring all of the optional pathways. For instance, once the learner has decided to become “database specialist”, he/she has to complete the mandatory pathway, by merging it with the process represented in the swimlane *Databases*. Figure 6

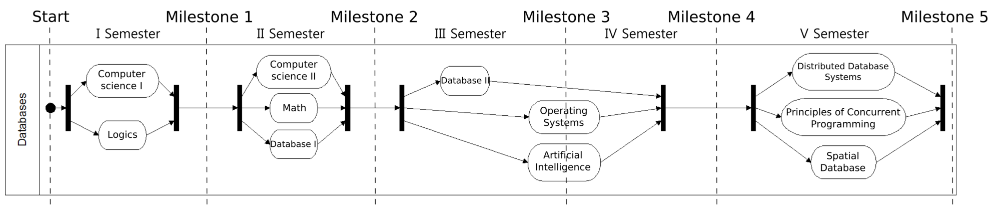


Figure 6. Example of a possible pathway obtained from the curriculum in Figure 5 by merging the mandatory pathway with the Databases additional pathway.

shows the curriculum resulting by merging the mandatory learning path and the *Databases* learning path in Figure 5. It should also be noted that processes representing a curriculum are only *views* combining activities. This is very compatible with UML activity diagrams where it is possible to reuse, in distinct contexts, activities defined once.

Another case to consider is when some two-semester course overlaps in time with another course. With reference to our example, Figure 5 Databases swimlane, the course *Artificial Intelligence* spans over the third and the fourth semesters, partially overlapping with the *Database II* course (held in parallel in the third semester). We cannot expect that *Database II* supplies competences that are preconditions to *Artificial Intelligence*. This issue can be solved once again by exploiting UML *partitions*. In fact, time overlapping can be handled by regulating the size and the relative position of the vertical partitions, i.e. by using *milestones*. The “timed semantics” remains underspecified and may be approached in the classical way by introducing time-dependent constraints on activity edges (or, on top of the interpretation of the UML superstructure specification – that often does not provide a sufficient level of detail – constraints attached to the partitions themselves).

5 Validation of curricula

The kinds pedagogical validation foreseen by the proposed authoring architecture are inter-conceptual and performed after the construction of a curriculum. In other terms, given a curriculum expressed as a UML activity diagram, it is possible to verify that all of its possible executions:

(1) allow the achievement of the user’s *learning goal*, contained in the *learner model*;
(2) do not show any *competence gap* w.r.t the *resource model*;
(3) satisfy the constraints in the *curricula model*.

All these validation tasks have been implemented by exploiting model checking techniques. Model checking is normally use to verify the properties of system. Two elements are needed: the set of facts that one wants to verify and the relevant aspects of the system that are needed to verify those facts (Holzmann, 2003). In our case, the system is the curriculum, while the facts to verify are respectively given by the learner model, the resource model or the curricula model, depending on the validation at issue. SPIN, by G. J. Holzmann (Holzmann, 2003), is one of the best known tools of this kind, for LTL formulas. In order to perform the validation tasks, our proposal is to use SPIN, to translate the UML activity diagram, that encodes the curriculum, in a Promela program (where Promela is the language used by SPIN), and, then, to verify whether this program satisfies the LTL formulas which encode the facts to prove. We perform the three validation tasks at the same time.

The use of SPIN is orthogonal to the semantics of LTL. Any model checker for this logic or any optimized algorithm that allows the same verifications could be used instead of it. This is the advantage of adopting a declarative semantics for the curricula model. The important thing is that the implementation is proved sound w.r.t. the LTL semantics. The choice of SPIN is due to the fact that SPIN can tackle *any* LTL formula, and this gave us some freedom while we were defining the semantics of the various DCML operators because we did not have to modify in any way the model checker. Moreover, when a model does not satisfy some LTL formula (or assertion), SPIN produces a *counterexample* that shows the violation. This is very important in order to return feedbacks to the users, that help in correcting curricula (or curricula models).

Let us now introduce how a UML activity diagram, that expresses a curriculum, can be translated into a Promela program. In the literature, we can find some proposals to translate UML activity diagrams into Promela programs, such as Gallardo et al. (2002); Guelfi & Mammar (2005). These proposals have a different purpose than ours and they cannot directly be used to perform the translation that we need to perform the verifications we list above, however, it is possible to follow them as guidelines to perform our translation. Generally, their aim is debugging UML designs, by helping UML designers to write sound diagrams. The translation proposed in the following, instead, aims to simulate, by means of a Promela program, the acquisition of competencies by attending courses contained in an activity diagram. The verification process is, in a way, a simulation of the learning paths that can be obtained by combining the mandatory and the optional swimlanes in the UML activity diagram. Every milestone corresponds to a *state*, containing specific competences – all those acquired up to that point, partly because belonging to the learner from the very beginning and partly because achieved by attending courses. The *initial state* contains all the facts concerning the user at issue, which are contained in the learner model. The *final state* contains the competences that are gained by using the various learning resources which lay upon the followed learning path.

We represent all the competences involved by a UML activity diagram as *integer variables*. In the beginning, only those variables that represent the initial knowledge owned by the learner are set to a value greater than zero. *Courses* are represented as actions that can modify the value of such variables. Since we suppose that the learner's knowledge can only grow, also the value of variables can only grow.

The Promela program produced by the translation simulates the way competences are acquired, updating the set of the achieved competences at every step. Steps correspond to the various milestones into which the curriculum is organized. For instance, in Figure 5 we identify the initial state, a second state corresponding to the end of the first semester, another corresponding to the end of the second semester, and so on, up to a final state, corresponding to the end of the curriculum[1]. Hereafter, we report the top level of the program corresponding to Figure 5:

```
proctype CurriculumVerification() {
    milestone_1();
    milestone_2();
    milestone_3();
    milestone_4();
    milestone_5();
    LearningGoal();
}
```

In order for the curriculum to contain no *competence gap*, it is sufficient that all the *milestone* procedures are completed successfully. In order to understand how milestones are implemented, let us see, first, how courses are encoded. Each *course* is represented by its preconditions and its effects. For example, the course *Database_II* is encoded as follows:

```
inline preconditions_course_database_II() {
        assert(indexes>=2 && relational_model>= 2 &&
               entity_relationship_model  >=2);
}

inline effects_course_database_II() {
        SetCompetenceState(object_database ,3);
        SetCompetenceState(object_definition_language ,2);
        SetCompetenceState(object_query_language ,2);
        SetCompetenceState(object_relational ,3);
        SetCompetenceState(query_optimization ,3);
}
```

The SPIN instruction *assert* verifies the truth value of its condition, which in our case is the precondition to the course. If violated, SPIN interrupts its execution and reports about it. *SetCompetenceState* is used in the implementation of the effects of a course: it affects the current state, by adding new competences

[1]In this article we report only some parts of the Promela program obtained by translating the example in Figure 5. The whole program, as well as the LTL formulas corresponding to the whole curricula model in Figure 2, are available at `http://www.di.unito.it/~emarengo/DCMLexamples`.

or by increasing the level of proficiency of some competence. If all the learning paths represented by the translated UML activity diagram have *no competence gap*, no assertion violation will be detected. Otherwise, a counterexample will be returned that corresponds to a sequence of courses bringing about the violation, thus, giving a precise feedback to the student/teacher/instructional designer.

Generally speaking, a milestone implements a transition in the learning process where: (1) all the preconditions of the courses attended in that phase are checked; (2) all the effects of courses attended in that phase are added to the state. If some preconditions are not satisfied a competence gap is identified. If the curriculum contains various possible learning paths, every milestone verifies the mandatory courses and simulates all of the different alternatives concerning optional courses. This is done by means of the introduction of an array of variables, that is used to discriminate among the alternative paths.

```
inline milestone_1(){
    atomic{
        preconditions_course_computer_science_I();
        preconditions_course_logics()

        if
        :: true ->
                if
                ::(C_programming < PATH_1) -> path_additional[1]=1;
                      preconditions_course_principles_of_programming();
                :: else -> path_additional[1]=PATH_2;
                fi;
                path_additional[0] = PATH_1;
        :: true ->
                path_additional[0] = PATH_2;
        :: true ->
                preconditions_course_probability_and_statistics();
                preconditions_course_biology_I();
                path_additional[0] = PATH_3;
        fi;

        effects_course_computer_science_I();
        effects_course_logics();

        if
        :: (path_additional[0]==PATH_1) ->
                if
                ::(path_additional[1]==PATH_1) ->
                      effects_course_principles_of_programming();
                ::(path_additional[1]==PATH_2) -> skip;
                fi;
        :: (path_additional[0]==PATH_2) -> skip;
        :: (path_additional[0]==PATH_3) ->
               effects_course_probability_and_statistics();
               effects_course_biology_I();
        fi;
    }
}
```

The last instruction of the process *CurriculumVerification*, which is applied only if the curriculum can be executed to its end, is *LearningGoal*. *LearningGoal* checks whether the *user's learning goal* is satisfied, by applying a test on the knowledge in the final state. For example, a learner interested in event-driven programming, concurrent programming, and file systems could have the goal:

```
inline LearningGoal() {
   assert(
        event_driven_programming >= 2 &&
        concurrent_programming >= 2 &&
        file_systems >= 1
   );
}
```

To check if the curriculum complies to a curricula model, first, the curricula model is turned into the corresponding LTL formulas, by applying the translations given in Section 3. Notice that SPIN uses *strong until* "U" and not *weak until* "U", as we have done, therefore, we use the equivalence relation $\phi\ \mathsf{U}\ \psi \equiv (\phi U \psi) \vee \Box\phi$. Then, it is possible to apply SPIN and check if every possible learning path satisfies them. Since all the DCML constraints are in conjunction with each other, it is possible to check them one at a time, thus reducing the complexity of the problem drastically, because the automaton resulting for a single constraint is small. As soon as a constraint verification fails, the procedure stops.

6 Implementation in the Personal Reader Framework

We designed the Personal Reader (PR) Framework (`http://www.personal-reader.de/`) as a tool and test-bed for creating applications in the Semantic Web. It offers an environment for designing, implementing and realizing Web content readers using a service-oriented approach, for a more detailed description, see Henze & Krause (2006). In applications based on the Personal Reader Framework, users can select and combine – plug together – the personalization support they want to receive. The framework has already been used for developing Web Content Readers that present online materials in an embedded context (Baumgartner et al., 2005; Henze, 2005; Abel et al., 2006).

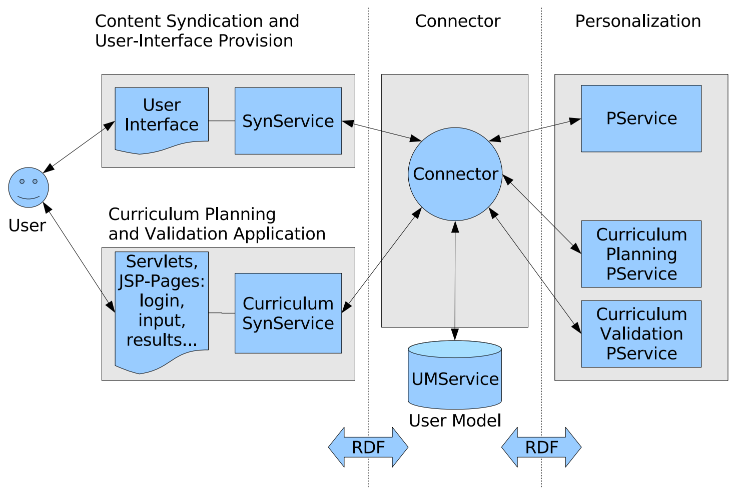

Content Syndication and User-Interface Provision
Connector
Personalization
User Interface
SynService
User
Connector
PService
Curriculum Planning and Validation Application
Servlets, JSP-Pages: login, input, results...
Curriculum SynService
Curriculum Planning PService
Curriculum Validation PService
UMService
User Model
RDF
RDF

Figure 7. Personal Reader Framework Overview.

Figure 7 gives a brief overview of the main components of the PR architecture. A typical PR application consists of three types of services. *Personalization services* (PService) provide personalization functionalities: they deliver personalized recommendations for content, as requested by the user and obtained or extracted from the Semantic Web. The *Syndication Services* (SynService) realize the user interface and facilitate interoperability with the other services in the framework, e.g. it allows for the discovery of the applications' interfaces by a portal. The *Connector* is a single central instance responsible for controlling the communication between user interface and personalization services. It selects services based on their semantic description and the requirements by the SynService. The Connector protects – by means of a public-key-infrastructure (PKI) – the communication among the involved parties. It also supports the customization and invocation of services and interacts with a user modelling service, called the *UMService*, which maintains a central user model.

The Personal Reader architecture requires web services to be described in OWL-S, and also exploits taxonomies (Abel et al., 2006) for describing services, their capabilities and their configurable options (required for personalization). This information is used for accomplishing the service selection task by the Connector (see Figure 7). Notice that the aims of the PR are, however, different than those of frameworks like WSMO (Roman et al., 2005) and IRS-III (Cabral et al., 2006): the focus, in fact, is not to provide a generic framework to create web service compositions but to configure web services according to user and application requirements, and select services based on their personalization functionality.

For the implementation of the Curriculum Planning and Validation Application, the User Interface (SynService) was realized using Servlets and JSP Pages. Other applications, like the MyEar system (Henze & Krause, 2006) also use active scripting approaches (Java Script and Ajax) for creating adaptive user interfaces. For the actual data processing and personalization, two PServices were implemented, one for performing the *curriculum sequencing* task, the other hosting the *validation engine*.

6.1 *Metadata Description of Courses*

To apply our ideas to a real world scenario, we created the resource model (a corpus of courses) by extracting information from an existing database. We used the Lixto (Baumgartner et al., 2001) tool to extract the needed data from the web-pages provided by the HIS-LSF (`http://www.his.de/`) system of the University of Hannover. This approach was chosen based on our experience with Lixto in the *Personal Publication Reader* (Baumgartner et al., 2005) project, where we used Lixto for creating the bibliographic database by crawling the publication pages of all the REWERSE partners. The use of Lixto is motivated by the fact that a lot of information about courses and academic activities is already on-line, though its representation does not include a semantic annotation and it is unlikely that such an annotation will be added manually in the future. Hence the need of adopting a tool that is capable of extracting such a semantic annotation automatically. The effort to adapt our existing tool for the new data source was only small. From the extracted metadata we created the RDF knowledge base. For each of the courses, the course name, catalog identifier, semester, the number of credit points, effects and preconditions, and the type, e.g. laboratory, seminar, or regular course with examinations in the end, was recorded. Figure 8 shows the metadata properties of the course *Digital Image Processing*.

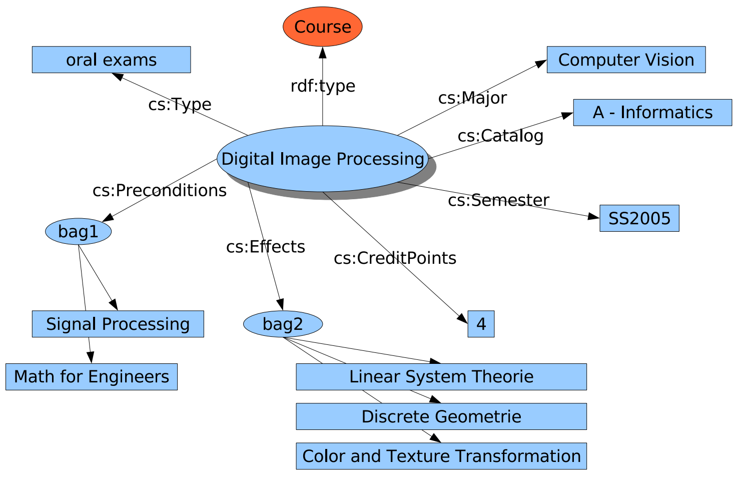


Figure 8. Metadata for the course *Digital Image Processing* from the Hannover course database.

As it turned out, the biggest problem was that the quality of most of the information in the database is insufficient, largely because of inconsistencies in the description of prerequisites and effects of the courses. Additionally the corpus was not annotated using a common set of terms, but each author and department secretary used a slightly varying vocabulary in the descriptions that they edited, instead of relying on a common classification system, like e.g. the ACM Computing Classification System (ACM CCS `http://www.acm.org/class/`).

As a consequence, we focussed only on a subset of the courses (computer science and engineering courses), and manually post-processed the harvested data. In our system, courses are annotated with prerequisites and effects, which correspond to either required or supplied competences. After the automatic extraction of effects and preconditions, the collected terms were spell-checked and harmonized, synonyms were removed and annotations were corrected where necessary. The resulting corpus had a total of 65 courses left, with 390 effects and 146 preconditions.

6.2 *The User Interface and Syndication Service*

The Syndication service (SynService) is responsible for creating the user interface to present to the learner, for requesting the necessary personalization functions from the connector, as well as all for the communication between the different Web services. In particular, it identifies the user and presents him/her an interface that allows the selection of the competences to be acquired. Furthermore, it has to display the results of the planning and validation processes (see Figure 9), allowing for further refinement of the created plans.

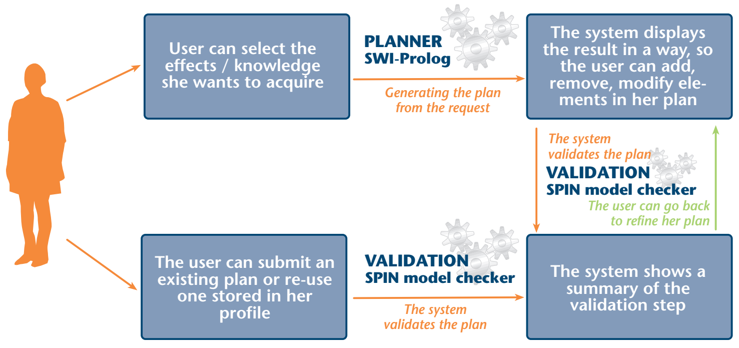

User can select the effects / knowledge she wants to acquire
PLANNER
SWI-Prolog
Generating the plan from the request
The system displays the result in a way, so the user can add, remove, modify elements in her plan
The system validates the plan
VALIDATION
SPIN model checker
The user can go back to refine her plan
The user can submit an existing plan or re-use one stored in her profile
VALIDATION
SPIN model checker
The system validates the plan
The system shows a summary of the validation step

Figure 9. The Actions supported by the User Interface.

The curriculum sequencing and the validation tasks are delegated to two independent Personalization Services, the *Curriculum Planning PService*, and the *Curriculum Validation PService*. Because of the Plug & Play nature of the infrastructure, the two PServices can be used by other applications (SynServices) as well (Figure 9). It is also possible to use additional PServices by extending the SynService so to provide additional planning and validation capabilities to our application. The current implementation of the Curriculum Planning and Validation Prototypes can be reached via the Projects page of the Personal Reader Homepage `http://personal-reader.de`. The initial learning goal selection page contains fields for up to three required goals, as well as the number of credit points to achieve and the number of plans to generate.

6.3 *Automatic construction of curricula: the Curriculum Planning PService*

We have developed a simple service for building personalized curricula, which has been integrated as a Plug & Play personalization service in the Personal Reader architecture. The curriculum is *personalized* in the sense that it allows a user to reach his/her learning goals, starting from the current competences the user has, which are included in the user model.

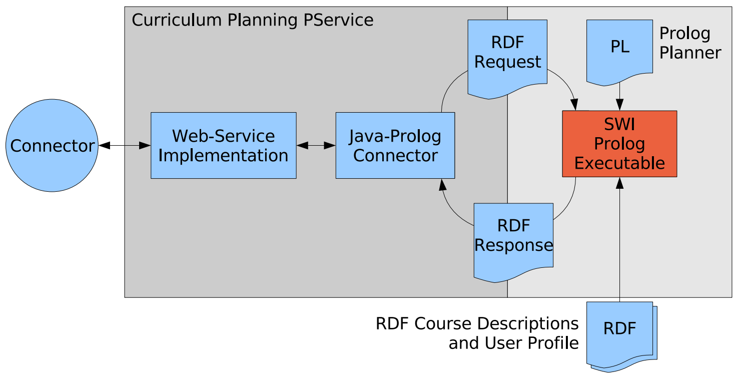

Curriculum Planning PService
RDF Request
PL
Prolog Planner
Connector
Web-Service Implementation
Java-Prolog Connector
SWI Prolog Executable
RDF Response
RDF Course Descriptions and User Profile
RDF

Figure 10. Curriculum Planning Web Service.

The planner was implemented as a Prolog program, so we had to embed a Prolog reasoner into the Web service. The planner executes a simple *depth-first forward planning* (an early prototype was presented in (Baldoni et al., 2006a)), where actions cannot be applied more than once. The algorithm is simple:

(1) starting from the initial state, the set of *applicable* actions (those whose preconditions are contained in the current state) is identified;
(2) one of such actions is selected and its application is simulated leading to a new state;
(3) the new state is obtained by adding to the previous one the competencies supplied as effects of the selected action;
(4) the procedure is repeated until either the goal is reached or a state is reached, in which no action can be applied and the learning goal is not satisfied;
(5) in the latter situation, backtracking is applied to look for another solution.

The procedure will eventually end because the set of possible courses is finite and each is applied at most once (the trivial assumption is that a course can be attended only once). If the goal is achieved, the sequence of courses that label the transitions leading from the initial to the final state is returned as the resulting *curriculum*. If desired, the backtracking mechanism allows the collection of a set of alternative solutions to present to the user.

Figure 10 gives an overview over the components in the current implementation. The Web service implements the Personalization Service (*PService* for short, see Henze & Krause (2006)) interface, defined by the Personal Reader framework, which allows for the processing of RDF documents and for inquiring about the services capabilities. The *Java-to-Prolog Connector* runs the SWI-Prolog executable in a sub-process; essentially it passes the RDF document containing the request *as-is* to the Prolog system, and collects the results, already represented as RDF.

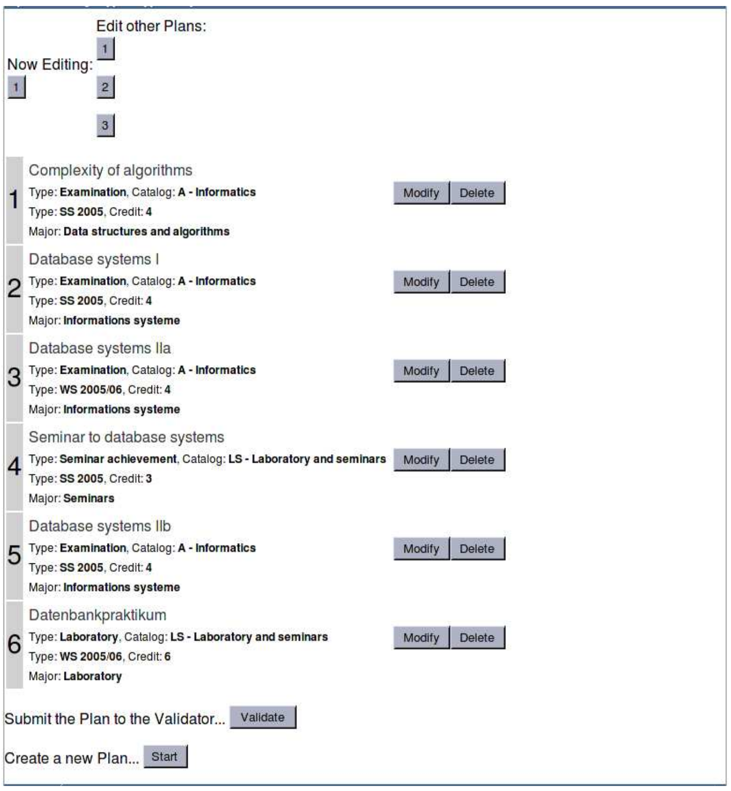
The curriculum planning task itself is accomplished by a reasoning engine, which has been implemented in SWI-Prolog (`http://www.swi-prolog.org/`). The interesting thing of using SWI-Prolog is that it contains a semantic web library allowing to deal with RDF statements. Since all the inputs are sent to the reasoner in a *RDF request document*, it actually simplifies the process of interfacing the planner with the Personal Reader. In particular the request document contains: a) links to the RDF document containing the database of courses, annotated with metadata (the resource model), b) a reference to the learner model, including the user's actual learning goal, i.e. a set of competences that the learner would like to acquire, and that concern competencies belonging to the *domain model*. The reasoner can also deal with information about credits provided by the courses, when the user sets a credit constraint together with the learning goal.

At the end of the process, a *RDF response document* is returned as an output. It contains a list of plans (learning paths) that fulfill the learner model. The maximum number of possible solutions can be set by the learner in the request document. Notice that further information stored in the learner model is used at this stage for adapting the presentation of the solutions, here simple hints are used to *rank higher* those plans that include topics for which the user has expressed a special interest. Figure 11 shows the output generated by the syndication service for the generated plan, decoded from the RDF response document. The interface also provides the means for the learner to modify the plan and submit it for further validation.

6.4 *The Curricula Validation PService: current state of the implementation*

Given a curriculum, it is possible to perform post-construction validation tasks in order to verify if the curriculum allows the achievement of the learning goal, if it contains competence gaps, and if it satisfies the constraints of a given curricula model, by following the process described in Figure 12. Once a curricula model is defined, it can be translated into a LTL formula by following the process described in Section 5.

For what concerns the interaction of the learner with the validation system, there is the need of allowing the learner to insert the curriculum to be validated into the system. The easiest way for allowing a *naïve*



Edit other Plans: 1 2 3

Now Editing: 1

1 Complexity of algorithms
Type: **Examination**, Catalog: **A - Informatics**
Type: **SS 2005**, Credit: **4**
Major: **Data structures and algorithms**
Modify Delete

2 Database systems I
Type: **Examination**, Catalog: **A - Informatics**
Type: **SS 2005**, Credit: **4**
Major: **Informations systeme**
Modify Delete

3 Database systems IIa
Type: **Examination**, Catalog: **A - Informatics**
Type: **WS 2005/06**, Credit: **4**
Major: **Informations systeme**
Modify Delete

4 Seminar to database systems
Type: **Seminar achievement**, Catalog: **LS - Laboratory and seminars**
Type: **SS 2005**, Credit: **3**
Major: **Seminars**
Modify Delete

5 Database systems IIb
Type: **Examination**, Catalog: **A - Informatics**
Type: **SS 2005**, Credit: **4**
Major: **Informations systeme**
Modify Delete

6 Datenbankpraktikum
Type: **Laboratory**, Catalog: **LS - Laboratory and seminars**
Type: **WS 2005/06**, Credit: **6**
Major: **Laboratory**
Modify Delete

Submit the Plan to the Validator... Validate

Create a new Plan... Start

Figure 11. Output of the Planning Step.

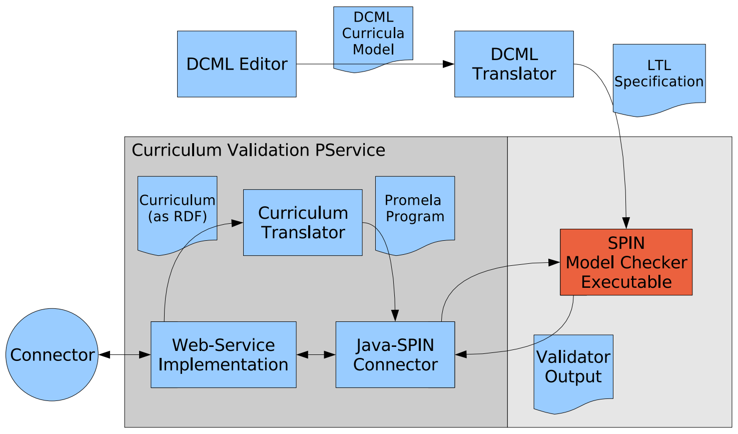
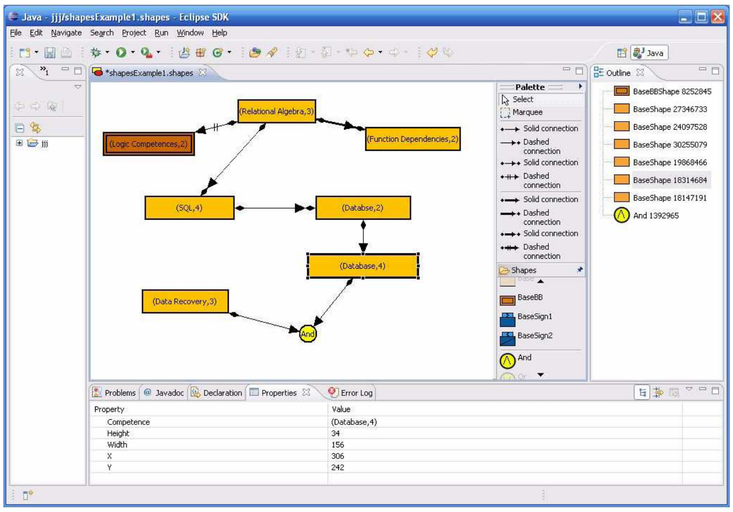


Figure 12. The validation w.r.t. a curricula model: process workflow.

user, like the average student, to perform this task is to ask him/her to enter a sequence of courses, corresponding to the desired curriculum, by using a web interface (as it was done, for instance, in the WLog system (Baldoni et al., 2004a)). A linear plan is the simplest kind of curriculum that can be captured by means of an activity diagram, however, this choice makes sense because it is unlikely that the student learns to exploit the full potential of an activity diagram-based representation in order to express curricula. A good handling of activity diagrams requires, in fact, some expertise that the naïve user does not have. Moreover, since the planning PService produces linear plans, it is also possible to compose the effect of the planning process with the validation service, in order to check the compliance of an automatically generated curriculum with a given curricula model.

The full potential of activity diagrams can, instead, be exploited by the instructional designer, when he/she faces the task of building and proposing new educational paths, which may, for instance, include



Property	Value
Competence	(Database,4)
Height	34
Width	156
X	306
Y	242

Figure 13. A screenshot of the DCML designer.

a mandatory part and one or more (alternative) options. In this case, the interaction will not necessarily be performed via the browser. The designer can, in fact, write activity diagrams by means of standard UML design system. Moreover, for helping the instructional designer in the construction of curricula models, we have developed an Eclipse plugin (the *DCML Designer*, see Figure 13 (Pistamiglio, 2007)). This tool is thought for being used by the *instructional designers* of an academic institute. The task of an instructional designer is to define the educational offer of the institute and the curricula models that must be respected by the curricula defined by the students.

In order to perform the check that a curriculum does not show competence gaps and supplies the user's learning goal, it is required to interface the Validation PService with the RDF course descriptions (contained in the resource model) and with the learner model. This can be done along the lines of what we have described in the previous section by translating a curriculum into Promela code.

7 Extending DCML to deal with time proximity

The constraints presented in Section 3 express temporal relations which do not capture the time proximity between the acquisitions of two competences. It is often the case, however, when by saying that a competence is to be acquired, for instance, before another competence, the designer actually means "immediately before", e.g. in the previous semester. In order to express this stronger kind of relations we have extended the core of DCML by adding the notions of *immediateness*. All the relations that we have introduced (see Figure 4) have a correspondent stronger version: *immediate before*, *immediate implication*, *immediate succession* and their negations. Graphically, the notion of immediateness is represented by double arrows (see Figure 14). In the following we briefly introduce these new relations, whose definitions exploit the temporal logic operator *next-time*: $\bigcirc\varphi$ means that the formula φ holds in the next state of the run.

Immediate before

Immediate before (see Figure 14) is represented by means of a double line arrow that ends with a little-ball. The constraint (k_1, l_1) *immediate before* (k_2, l_2) imposes that (k_1, l_1) holds before (k_2, l_2) and the latter either is true in the next state w.r.t. the one in which (k_1, l_1) becomes true or k_2 *never* reaches the level l_2. The difference w.r.t the *before* constraint is that it imposes that the two competences are acquired *in sequence*. The corresponding LTL formula is "(k_1, l_1) *before* (k_2, l_2)" $\wedge\Box((k_1, l_1) \supset (\bigcirc(k_2, l_2) \vee \Box\neg(k_2, l_2)))$. More generally, in presence of DNF formulas as antecedent and consequence

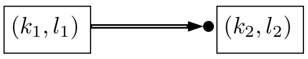
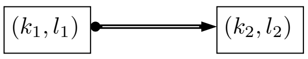
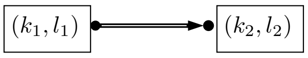
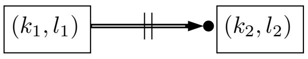
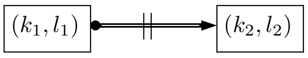
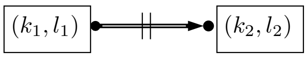
Immediately before: (k_1,l_1) *immediately before* (k_2,l_2)	
	“(k_1,l_1) *before* (k_2,l_2)” $\wedge\Box((k_1,l_1) \supset (\bigcirc(k_2,l_2) \vee \Box\neg(k_2,l_2)))$ If the competence (k_1,l_1) is acquired, then it must be acquired in the state that precedes the one in which competence (k_2,l_2) is acquired
Immediately implication: (k_1,l_1) *immediately implies* (k_2,l_2)	
	$\Diamond(k_1,l_1) \supset \Diamond(k_2,l_2) \wedge \Box((k_1,l_1) \supset \bigcirc(k_2,l_2))$ If the competence (k_1,l_1) is acquired in a certain state, in the following one the competence (k_2,l_2) must hold (not necessarily acquired)
Immediately succession: (k_2,l_2) *immediately succeeds* (k_1,l_1)	
	“(k_2,l_2) *succeeds* (k_1,l_1)” $\wedge\Box((k_1,l_1) \supset \bigcirc(k_2,l_2))$ If (k_1,l_1) is acquired, then (k_2,l_2) must be acquired in the same state or in the following one
Negative immediately before: (k_1,l_1) *not immediately before* (k_2,l_2)	
	$\neg(k_1,l_1)\ \mathsf{U}\ ((k_2,l_2) \wedge \neg(k_1,l_1))$ Competence (k_1,l_1) cannot hold in the state that precedes the one in which (k_2,l_2) is acquired
Negative immediately implication: (k_1,l_1) *not immediately implies* (k_2,l_2)	
	$\Diamond(k_1,l_1) \supset (\Box\neg(k_2,l_2) \vee \Diamond((k_1,l_1) \wedge \bigcirc\neg(k_2,l_2)))$ If competence (k_1,l_1) is acquired, then in the following state (k_2,l_2) cannot hold (i.e. $k_2 < l_2$).
Negative immediately succession: (k_2,l_2) *not immediately succeeds* (k_1,l_1)	
	$\Diamond(k_1,l_1) \supset (\Box\neg(k_2,l_2) \vee$ “(k_1,l_1) *not before* (k_2,l_2)” $\vee\Diamond((k_1,l_1) \wedge \bigcirc\neg(k_2,l_2)))$ If (k_1,l_1) is acquired then (k_2,l_2) cannot be acquired in the same state or in the following one (thus, it can be acquired before the acquisition of (k_1,l_1), after the state that follows it or it could never be acquired)

Figure 14. DCML notations for immediate before, immediate implication, immediate succession, and their negations.

of a “immediate before” relation, we have the following definition for “dnf_1 *immediate before* dnf_2”:

$$\bigvee_{cf_i \in dnf_1, cf_j \in dnf_2} \text{“}cf_i \textit{ before } cf_j\text{”} \wedge \Box(cf_i \supset (\mathsf{next}(cf_j) \vee \mathsf{absence}(cf_j)))$$

where $\mathsf{next}(cf) = \bigwedge_{(k_i,l_i)\in cf} \bigcirc(k_i,l_i)$.

Example 7.1 As an example, with reference to Figure 2 (right), let us consider the constraint: (*data_transfer*, 1) *immediately before* (*classical_encryption*, 2). If the competence (*data_transfer*, 1) is acquired, then this must happen in the state that precedes the one in which (*classical_encryption*, 2) is acquired.

This kind of constraint is useful because, even though we assume that once a competence is acquired, then, it cannot be removed from the set of competences owned by the student, in real life students tend to forget part of what they learnt. In some cases, however, it is important to be sure that certain competences are owned in (almost) their integrity in order to allow the acquisition of some other competence. In this case, it is better to use the immediately before constraint rather than the before constraint.

The *not immediate before* is translated exactly in the same way as the *not before*. Indeed, it is a special case because we assume that a competence cannot be forgotten. More generally, in presence of DNF formulas, "dnf_1 *not immediate before* dnf_2" is:

$$\bigvee_{cf_i \in dnf_1, cf_j \in dnf_2} \mathsf{negation}(cf_i)\ \mathsf{U}\ (cf_j \wedge \mathsf{negation}(cf_i))$$

Immediate implication

The *immediate implication*, instead, specifies that the consequent must *hold* in the state right after the one in which the antecedent is acquired. Note that this does not mean that it must be *acquired* in that state but only that it cannot be acquired afterwards. This is expressed by the LTL implication formula in conjunction with the constraint that whenever $k_1 \geq l_1$ holds, $k_2 \geq l_2$ holds in the next state: $\Diamond(k_1, l_1) \supset \Diamond(k_2, l_2) \wedge \Box((k_1, l_1) \supset \bigcirc(k_2, l_2))$. More generally, in presence of DNF formulas as antecedent and consequence of a "immediate implication" relation, we have the following definition for "dnf_1 *immediate implication* dnf_2":

$$\bigvee_{cf_i \in dnf_1, cf_j \in dnf_2} \text{“}cf_i \textit{ implies } cf_j\text{”} \wedge \Box(cf_i \supset \mathsf{next}(cf_j))$$

Example 7.2 Figure 2 (top) reports the following example of immediate implication: (*software_and_services_for_biology_on_the_web*, 3) *immediately implies* (*visualizing_informations*, 1). In this case if the learner does not own the competence (*visualizing_informations*, 1) yet, he/she must acquire it, and this acquisition must be performed immediately after *software_and_services_for_biology_on_the_web*. The reason is that in this case knowledge about *software_and_services_for_biology_on_the_web* is considered to be helpful in order to better understand *visualizing_information*.

"(k_1, l_1) *not immediate implies* (k_2, l_2)" imposes that when (k_1, l_1) holds in a state, $k_2 \geq l_2$ must be false in the immediately subsequent state. Afterwards, the proficiency level of k_2 does not matter. The corresponding LTL formula is $\Diamond(k_1, l_1) \supset (\Box\neg(k_2, l_2) \vee \Diamond((k_1, l_1) \wedge \bigcirc\neg(k_2, l_2)))$, that is weaker than the "classical negation" of the immediate implication. More generally, in presence of DNF formulas, "dnf_1 *not immediate implies* dnf_2" is:

$$\bigvee_{cf_i \in dnf_1, cf_j \in dnf_2} \mathsf{existence}(cf_i) \supset \mathsf{absence}(cf_j)$$

Example 7.3 Figure 2 (center) also reports an example of *not immediately implies* constraint: (*query_optimization*, 3) *not immediately implies* (*distributed_query_optimization*, 2).

This kind of constraint is useful to express the need of leaving some time to the learner, so that he/she can better assimilate some knowledge (about *query_optimization* in the example) before starting to face a complex topic (*distributed_query_optimization*).

Immediate succession

In the same way, the *immediate succession* imposes that the consequent either is acquired in the same state as the antecedent or in the state immediately after (not before nor later). The immediate succession LTL formula is "(k_2, l_2) *succeeds* (k_1, l_1)" $\wedge\Box((k_1, l_1) \supset \bigcirc(k_2, l_2))$. More generally, in presence of DNF formulas as antecedent and consequence of a "immediate succession" relation, we have the following definition for "dnf_2 *immediate succeeds* dnf_1":

$$\bigvee_{cf_i \in dnf_1, cf_j \in dnf_2} \text{“}cf_j \textit{ succeeds } cf_i\text{”} \wedge \Box(cf_i \supset \mathsf{next}(cf_j))$$

Example 7.4 Figure 2 (top-center) shows the following example: (*bioinformatics_web_based_resources*, 1) *immediately succeeds* (*software_and_services_for_biology_on_the_web*, 3). As for succession, this constraint expresses a temporal ordering between the times at which competences are to be acquired (*software_and_services_for_biology_on_the_web* first). Moreover, it specifies that *bioinformatics_web_based_resources* is to be acquired either in the same state or in the state that immediately succeeds the one at which *software_and_services_for_biology_on_the_web* is acquired. Once again, the reason is that the instructional designer considers to be important that the learner retains in his/her mind all the knowledge acquired about the first topic, in order to understand the latter.

Its negation imposes that if a competence is acquired in a certain state, in the state that follows, another competence cannot be acquired, that is $\Diamond(k_1, l_1) \supset (\Box\neg(k_2, l_2) \vee$ "(k_1, l_1) *not before* (k_2, l_2)" $\vee\Diamond((k_1, l_1) \wedge \bigcirc\neg(k_2, l_2)))$. Thus, when (k_1, l_1) is acquired (k_2, l_2) can already hold, it could never hold or it could be acquired in a state that follows the next one. More generally, in presence of DNF formulas, "dnf_2 *not immediate succeeds* dnf_1" is:

$$\bigvee_{cf_i \in dnf_1, cf_j \in dnf_2} \begin{array}{l} \mathsf{existence}(cf_i) \supset (\mathsf{absence}(cf_j) \vee \text{“}cf_i \textit{ not before } cf_j\text{”} \vee \\ \Diamond(cf_i \wedge \mathsf{next}(\mathsf{negation}(cf_j)))) \end{array}$$

8 Related works

In the literature it is possible to find different works that are related to the different aspects of the approach we present here. A recent proposal for automatizing the *competence gap verification* is done by Melia & Pahl (2006), where an analysis of pre- and post-requisite annotations of the Learning Objects, representing the learning resources, is proposed. A logic based validation engine can use these annotations in order to validate the curriculum/learning object composition. Melia and Pahl's proposal is inspired by the *CoCoA* system, by Brusilovsky & Vassileva (2003), that allows for the analysis and the consistency check of static web-based courses. Competence gaps are checked by a prerequisite checker for *linear courses*, simulating the process of teaching with an overlay student model. Pre- and post-requisites are represented as "concepts". In Melia & Pahl (2007a,b), the same authors propose a layered model for pedagogical courseware validation (CAVaM or CAVIAr), that inspired our proposal. Two kinds of validation are identified: one against pedagogical strategies, the other against pedagogical rules. For what concerns the former, one pedagogical strategy is supported: checking for the presence of competency gaps. Instead, for what concerns the latter, the authors propose the use of a rule engine, JESS, in order to verify that learning objects are shaped in a way that satisfies the instructional designer. For instance, for all goal concepts there must be a learning object of type *lecture* which teaches it. With respect to Melia and Pahl's work, the goal and constraint model that we defined is richer, because it allows the representation of any proficiency levels, and the expression of a wider range of constraint. Namely, we can express before, implication, succession and their negations, as well as time proximity, while they only foresee one kind of constraint, the before (interpreted as a prerequisite). Moreover, our constraints have a declarative semantics given by LTL. For what concerns the architecture, we have added one more layer, explicitly representing resources based on the domain model. This layer allows the representation of constraints that are local to a resource. Moreover, although they also use UML activity diagrams, they do not use the parallel operator, do not distinguish between mandatory and optional learning paths, and do not introduce the notion of duration of the composed activities. The last difference lays in the implementation of the validation tasks, that relies, in our case, in a model checker.

Along this line, also Garro et al. (2003) propose an approach to the automatic construction of *learning paths* (analogous to curricula), that has been implemented as part of the MASEL system (Garro et al., 2006). A learning path is a sequence of learning objects (analogous to learning resources), which allows the acquisition of a *skill* (a competence, in our terminology) that is desired by the user. Learning objects have a rich description; in particular, each learning object has some effects (skills that are supplied). Learning objects do not have preconditions, which are substituted by a "pre-requisite" relation between skills (this skill is to be acquired before this other skill). The approach also allows for the representation of the duration of learning objects and of the complementarity of certain sets of learning objects, moreover, skills include a proficiency level. In this work, however, the only reasoning task that is handled is planning, they do not have an explicit representation of curricula models and, therefore, they do not face the verification issues.

Brusilovsky and Vassileva (Brusilovsky, 2000; Brusilovsky & Vassileva, 2003) adopt a method that is close, in principle, to curriculum sequencing: *course sequencing*. The aim of course sequencing, a technique originally proposed in the field of Intelligent Tutoring Systems, is to supply users with personalized courses, which select, at every step of the learning process, the best teaching method, i.e. the teaching method that will help the user to get the closest to his/her learning goal. The term "teaching method" refers, for instance, to the possibility of facing an exercise rather than reading more detailed documentation about a topic. In particular, two models are proposed: DCG and CoCoA. Both systems help the construction of personalized courses on the basis of a semantic network, which composes a set of Domain Knowledge Elements, roughly corresponding to our competences. DCG organizes such elements in an AND-OR graph, while CoCoA adopts an heterarchic structure relying on the relations *part-of* and *attribute-of*. Both organizations represent the domain model. DCG applies "dynamic planning" techniques: at every step the student's advancements are verified by a test. If the test is passed, a new topic is presented, otherwise some replanning is performed in order to allow the student to fill the gaps. Each node in the domain model can be presented in different ways (e.g. test, exercise, motivation, example). A presentation plan component has the task of identifying the best presentation of the concept for that particular student in that particular context. This selection is performed based on a set of "teaching rules" encoded in the system. Dynamic planning fits very well the task of building student-oriented personalizations but it does not scale equally well to class-based personalization unless all students in class have similar learning skills. CoCoA supports the construction of courses by performing a set of operations among which consistency checks and quality checks. Also in this case different kinds of presentation are identified. CoCoA can perform *prerequisite checks* by simulating the execution of a course and verifying at every step that the preconditions to the current step are satisfied, i.e. that at that point the student will have the necessary knowledge. Each kind of presentation of each concept has its own prerequisites (so there are question prerequisites, presentation prerequisites, etc.). A limit of this approach is that prerequisites allow the expression of a single kind of constraint: what is to be learnt before a certain learning step can be accomplished. We have shown, by presenting DCML, that there are many other kinds of relation that it is interesting to capture, leading to the introduction of a further level of abstraction: the one given by our curricula model.

Brusilovsky & Vassileva (2003) also define various verification tasks, besides competence gaps, among which two tasks that we accomplish in the present proposal: (a) verifying that the curriculum allows the achievement of the user's *learning goals*, and (b) verifying that the curriculum is compliant against the *course design goals*. Manually or automatically supplied curricula, developed to reach a learning goal, should match the "design document", a sort of *curricula model*, specified by the institution that offers the possibility of personalizing curricula. In Brusilovsky & Vassileva (2003) design documents are said to specify general rules for designing sequences of learning resources (courses). In this proposal, the design document has a procedural nature and the verification checks that the curriculum respects the defined structure. The authors, however, do not supply guidelines or specifications that help writing design documents. We interpreted such general rules not is a procedural manner but rather as *constraints*, that are expressed in terms of competences and, in general, are not directly associated to learning resources, as instead is done for pre-requisites. They constrain the process of acquisition of concepts, independently from the resources.

Another proposal is the one by Farrell et al. (2004), who propose an approach called *Dynamic Assembly*. This approach allows the automatic generation of a course by composing pre-existing learning objects on the base of a set of parameters (e.g. level of detail, time, keywords) which are specified by the user. Parameters are expressed in a *course assembly page*. The system provides two methods, namely *in-depth* and *overview*, which respectively narrow the search to the topics listed by the user and return also learning objects concerning related topics. The result of the search is a numbered sequence of links to learning objects. The approach does not supply methods for verifying that the produced sequences, besides reaching the goal, also “make sense” from an educational point of view, for instance by exploiting some abstract model.

The works by Castro & Manzano (2001); Wu & Havens (2005); Lambert et al. (2006) focus on the curriculum planning problem, intended as the problem of planning an *academic schedule* of activities. So the goal is to organize the different courses of an academic curriculum on a given set of time periods (e.g. semesters), satisfying some academic (e.g. course availability, prerequisites, eligibility rules) or student constraints. The problem of planning an academic schedule can be naturally tackled by adapting to the application context a *Constraint Satisfaction Problem (CSP) modeling framework*, as proposed in Castro & Manzano (2001); Wu & Havens (2005); Lambert et al. (2006). In general, CSP models can be considered a good starting point for dealing with bindings to the fruition of the learning resources that may depend on the kind of infrastructure supplied by the academic institution (e.g. number and location of rooms, support services for computing, number of teachers and so on). In particular, Wu & Havens (2005) have proposed an extended model that exploits *mixed-initiative* constraint reasoning algorithms and provides flexibility in satisfying not only the constraints given by the institution offering the courses, but also the student’s preferences and needs. In a first phase (*curriculum planning*), a set of courses available in a certain period of time is identified, by interpreting the task as a constraint satisfaction problem with preferences. Courses are then presented to the user by means of an interface that organizes them in a table, whose columns correspond to teaching periods. The user can adjust the presented solution by modifying part of it. The first plan is obtained by applying the constraint given by the institution, while the modification actions taken by the user are interpreted as user constraints. A modified plan is validated in order to see if it still respects the overall set of constraints and then the interaction with the user can be repeated. The kinds of constraint that the system can handle are of different types. Among them: a course can appear only once per plan, courses can require other courses (not competences) as prerequisites, it is not possible to attend “equivalent” courses, some courses are mandatory. As one can observe, this kind of constraints is not defined on the competences required/thought by courses nor they express the constraints on the fruition of the teaching materials supplied by the teacher.

In Castro & Manzano (2001); Lambert et al. (2006) the authors tackle the problem of building *balanced academic curricula*. This term identifies curricula in which courses are well-distributed, according to a set of *load constraints*, along the trimesters/years. Besides load constraints also constraints concerning course prerequisites are considered during the planning phase. As in the previous approach, a course prerequisites correspond to other courses which are to be attended before the one at issue. The problem is interpreted as an optimization problem, which can be tackled in different ways (e.g. branch and bound, constraint programming, genetic algorithms).

Last but not the least, recently some attention has been posed also on ontologies for representing in a uniform way information like academic programs and examination regulations. For example, Hackelbusch & Appelrath (2008) proposes an ontology for representing this information as well as the universities supply of courses and the individual results of the students. The idea is to use this kind of ontologies in the process of producing or validating personalized curricula.

9 Conclusions

This article integrates and extends the results of a collaboration between the Department of Computer Science from the University of Torino and the University of Hannover, carried on within the *Personalized Information Systems* working group of the REWERSE European Network of Excellence (Baldoni et al., 2005, 2007a,b). In this work, we faced the problem of curricula authoring and developed a layered architecture, inspired by Melia & Pahl (2007a,b), that harmonizes different kinds of knowledge and of activity, expressed at various abstraction levels. As for all Semantic Web applications, each layer represents knowledge according to a well-defined semantics, thus enabling the application of automatic reasoning techniques as well as the integration with the representations belonging to other abstraction layers. The presented proposal is an evolution of earlier works (Baldoni et al., 2004a,b, 2005), where we applied semantic annotations to learning objects, with the aim of building compositions of new learning objects, based on the user's learning goals and exploiting planning techniques. That proposal was based on a different approach that relied on the experience of the authors in the use of techniques for reasoning about actions and changes which, however, did not allow to tackle with curricula models and to the related verification tasks.

In the design of the layered architecture, great attention has been posed on the representation of courses and curricula, and on how to define curricula models. The chosen approach relies on the notion of competence, thus introducing an abstract perspective, in which courses do not directly depend on one another but rather have knowledge prerequisites and effects on the learner model (Baldoni et al., 2002; Eisinger et al., 2005). Expressing learning goals and curricula in a way that is independent from specific resources and from their nature makes our proposal suitable to open environments, where the set of learning resources changes along time. For instance, a teacher may offer also a multimedia version of the material of his/her course for those students who cannot attend lectures. The availability of multiple versions of a same course does not affect in any way the descriptions of curricula model of the school at which he/she teaches, and the learning goals of his/her students.

Based on competences it is also possible to define pedagogical guidelines and rules that a curriculum must respect. To this aim, we have identified a set of *inter-conceptual* constraints and defined DCML, a graphical language for designing curricula models. As described in the previous sections, DCML allows the representation of temporal constraints posed on the acquisition of competences (supplied by courses), taking into account both the concepts supplied/required and the proficiency level. The language has a grounding in Linear Temporal Logic (Emerson, 1990) and, hence, allows the application of various forms of reasoning and, in particular, to execute validation tasks like: the verification that a curriculum does not have competence gaps, the verification that a curriculum allows the acquisition of a learning goal, the verification that a curriculum satisfies the constraints contained in a curricula model. Such tasks can be accomplished independently from the source of the involved learning resources as well as from their media.

We have shown how model checking techniques (Clarke & Peled, 2001) can be used to execute all these validation tasks. This use of model checking is inspired by van der Aalst & Pesic (2006), where LTL formulas are used to describe and verify the properties of a composition of Web Services. Another recent work that inspired this proposal is Terenziani et al. (2006), where medical guidelines, represented by means of the GLARE graphical language, are translated in a Promela program, whose properties are verified by using SPIN. Similarly to Terenziani et al. (2006), the use of SPIN gives an *automaton-based semantics* to a curriculum – the automaton generated by SPIN from the Promela program –. The declarative, formal, representation of curricula models is instead given as a set of LTL constraints. This representation enables other forms of reasoning. In fact, as for all logical theories, we can use an inference engine to derive other theorems or to discover inconsistencies in the theory itself.

The underlieing assumption that we have done is that compentencies are given in terms of a shared vocabulary, which is known by all the involved actors, i.e learners, course creators, instructional designers. This is a common assumption in the Semantic Web field, and it is a necessary assumption for allowing the interoperability of the actors, mediated by the application (Hackelbusch & Appelrath, 2008). This assumption, however, is very strong if we think to a realistic application. In this case, it is, in fact, quite likely that different actors and different schools use different vocabularies. This is a problem to face if we mean the proposed techniques to be used in the real world. We have already generally mentioned ontology alignment, merging and integration (Euzenat & Shvaiko, 2007). More practically, one possible, emerging solution for coping with this problem could be to rely on *upper ontologies*. Upper ontologies have, in fact, been proposed for being used as a *lingua franca*, to which mapping different vocabularies (Mascardi et al., 2007) in an automatic way. Specifically for what concerns the curriculum sequencing task, the implementation that we have described uses SWI-Prolog, which is thought for the integration with Semantic Web technology and provides a library for dealing with RDF. In this work we have proposed a representation of learning resources that are supplied by different sources and through different media (e.g., textual documents as well as videos or interactive applications or even actual lectures and courses). In the general case, the basic representation of resources, given in terms of competences, can also easily be enriched by adding *media-related* information and *accessibility* information, expressing constraints on how a resource is to be played/used. This information differs, in quality, from competence-based preconditions; in fact, while the latter only concerns the learner's knowledge, the former concerns some either environmental or physical aspect concerning the ability and the possibility of *using* the resource. So, for instance, a visually-impaired learner cannot watch an explanatory animation. For what concerns resources, this kind of information could be added to the precondition by enriching the annotation. On the other hand, the accessibility constraints of a learner could be added to the learner model. It is easy to extend the validation and the planning functionalities so to take into account also this kind of information. Further along this line also tasks that explicitly require a manipulation of the learning resources, which depends on their nature, and that for this reason are orthogonal to our proposal, can be integrated. Among them, media processing workflows and synchronization among media objects. The first task consists in processing, filtering, and fusing multimedia (and sometimes also real-time) data, providing various Quality of Service (QoS) guarantees (Candan et al., 2006; Brunkhorst et al., 2008). The latter, instead, deals with spatial and temporal compositions of media items (Celentano & Gaggi, 2003; W3C, 2001). Some approaches, e.g. Bertolotti (2005), express correct synchronizations by means of spatial and temporal constraints, that explicitly concern resources. For what concerns synchronization, Allen's Interval Algebra (Allen, 1983) has been used in order to formalize connections between temporal intervals by means of a sets of base relations. The intervals coincide with the length of a media object (the elapsed time from its beginning to its end). Extensions of this algebra, e.g. Laborie (2006), have been proposed to tackle also spatial relations beetwen different objects. The modularity of our architecture allows one to enrich, when necessary, the representation models and to extend the framework so to deal with tasks that are source-specific or media-related.

In this article, we have also reported about the integration of the above approach into the Personal Reader Framework (Henze & Krause, 2006). Despite some manual post-processing for fixing inconsistencies, we used real information from the Hannover University database of courses for extracting the meta-data. Currently the courses are annotated also by meta-data concerning the *schedule*, given in terms of *semesters*, and location of courses, like for instance room-numbers, addresses and teaching hours. This information is not used by the implemented services yet but it would be interesting to develop, as future work, also other services, which, along the lines of Wu & Havens (2005), complete the curricula planning process and produce an actual schedule.

The Personal Reader Platform provides a framework for implementing a service-oriented approach to personalization in the Semantic Web, and supplies a suitable infrastructure for building personalization applications, that consist of re-usable and interoperable personalization functionalities. The idea of taking a service-oriented approach to personalization is quite new and was born within the personalization working group of the Network of Excellence REWERSE. The adoption of a service-oriented architecture for the Personal Reader makes the introduction of new personalization functionalities very easy because each service focuses on dealing with a specific kind of knowledge and on performing a well-identified task, and is developed independently from the other services. Depending on the available resources and on the user's desires, it is possible to combine different services so to compose the functionalities that they implement, and, thus, perform more complex personalization tasks.

As future work, it will also be interesting to complement the architecture by integrating Web 2.0 features and develop personalization functionalities for handling information extracted from the activity of the community of users. This seems a very promising direction for recommendation purposes. For instance, when a user is uncertain on which curriculum to choose between two offers that are equivalent from the point of view of the supplied knowledge, he/she could be recommended to select one of the two on the basis of the behavior of other members in the community that we know as being "friends" of the current user.

Acknowledgements

The authors would like to thank prof. Giuseppe Berio for the helpful discussions. This research has partially been funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (cf. `http://rewerse.net`), by DAAD and Ateneo Italo-Tedesco through the Vigoni German and Italian researchers exchange Program 2007-2008 and by Regione Piemonte through the project ICT4LAW.

Short biographical notes on all contributors

Matteo Baldoni is associate professor of Computer Science at the University of Torino since 2006. He received a Ph.D. in Computer Science from the same university. He has a background in computational logic, multi-modal and non-monotonic extensions of logic programming and reasoning about actions. Current research interests: communication protocol design and implementation, conformance and interoperability of services, personalization by reasoning in the semantic web. He has been co-chair of the workshop AWESOME@MALLOW'007 and '009, and has been co-chair of the last five editions of the DALT@AAMAS international workshop. He chairs the working group "Sistemi ad Agenti e Multiagente" of the Italian Association for Artificial Intelligence.

Cristina Baroglio is associate professor of Computer Science at the University of Torino since 2005. She has a Ph.D. in Cognitive Sciences from the same university, and has a background in Machine Learning and Automated Reasoning. She is co-chair of the workshop "Agents, Web Services, and Ontologies: Integrated Methodologies" (AWESOME@MALLOW'007 and '009), and chair of the Reasoning web Summer School board since 2008. Her research interests include: semantic web and semantic web services, adaptation based on reasoning, conformance and interoperability of services to choreographies/protocols, automatic teaching to artificial agents, formal approaches to e-learning.

Ingo Brunkhorst Ingo Brunkhorst is a Ph.D. student and researcher at the L3S research center in Hannover, Germany. Until 2008 he has been assistant coordinator of the working group on Personalization of the European network of Excellence REWERSE. He is an expert of Semantic Web technologies and one of the main developers of the Personal Reader.

Nicola Henze is a junior professor at the Leibniz Universität of Hannover, Germany, and she is an associate member of the L3S research center. In 2000 she got a Ph.D. by discussing a thesis with the title "Adaptive Hyperbooks: Adaptation for Project-Based Learning Resources". She is the winner of the Semantic Web Challenge Award 2005 for the Personal Publication Reader. Unitl 2008, she has been the coordinator of the working group on Personalization of the Network of Excellence REWERSE (Reasoning on the Web with Rules and Semantics).

Elisa Marengo is Ph.D. student of Computer Science at the University of Torino. She received her degree summa cum laude in Sistemi per il Trattamento dell'Informazione in 2008 from the University of Turin. Her main research interests are: e-learning, knowledge representation and reasoning, web services and semantic web.

Viviana Patti is a researcher associate in Computer Science at the University of Torino since 2005. She received her Ph.D. in Computer Science in 2002. Her research interests include: computational logics for agent programming, reasoning enabling personalization in the semantic web, web service interoperability and conformance verification, web-based education courseware and curricula. She has been involved in the organization of several international events mainly in the fields "Personalization in the Semantic Web", "Web Information systems and Technologies" and "Agents". She has been member of the European Network of Excellence REWERSE.

References

Abel, F., Brunkhorst, I., Henze, N., Krause, D., Mushtaq, K., Nasirifar, P., et al., Personal Reader Agent: Personalized Access to Configurable Web Services. (2006). Technical report, Distributed Systems Institute, Semantic Web Group, University of Hannover. `http://www.personal-reader.de/wp/publications`.

Allen, J.F. (1983). Maintaining knowledge about temporal intervals. Vol. 26 (11), (pp. 832–843). November 1983. ACM.

Antoniou, G., Franconi, E., & van Harmelen, F. (2005). Introduction to Semantic Web Ontology Languages. In N. Eisinger & J. Małuszyński (Eds.), Reasoning Web, Lecture Notes in Computer Science (pp. 1–21). Springer-Verlag.

Antoniou, G., Uwe Aßmann, U., Baroglio, C., Decker, S., Henze, N., Patranjan, P.L., et al. (Eds.). Chapter title. *Reasoning Web, Third International Summer School 2007, Dresden, Germany, September 3-7, 2007, Tutorial Lectures*, Vol. 4636 of *Lecture Notes in Computer Science.* Springer (2007).

Antoniou, G., & van Harmelen, F. (2004). *A Semantic Web Primer*. the MIT Press.

Baldoni, M., Baroglio, C., Brunkhorst, I., Henze, N., Marengo, E., & Patti, V. (2006a). A Personalization Service for Curriculum Planning. In E. Herder & D. Heckmann (Eds.), Proc. of the 14th Workshop on Adaptivity and User Modeling in Interactive Systems, ABIS 2006, Vol. 1/2006, (pp. 17–20). October 2006. Hildesheim, Germany: University of Hildesheim, Institute of Computer Science.

Baldoni, M., Baroglio, C., Brunkhorst, I., Marengo, E., & Patti, V. (2007a). Reasoning-based Curriculum Sequencing and Validation: Integration in a Service-Oriented Architecture. In E. Duval & R. Klamma (Eds.), Proc. of EC-TEL 2007 - 2nd Europ. Conf. on Technology Enhanced Learning, Vol. 4756 of *LNCS* (pp. 426–431). Springer.

Baldoni, M., Baroglio, C., & Henze, N. (2005). Personalization for the Semantic Web. In N. Eisinger & J. Maluszynski (Eds.), Reasoning Web, First International REWERSE Summer School 2005, Vol. 3564 of *LNCS* (pp. 173–212). July 2005. Malta: Springer-Verlag.

Baldoni, M., Baroglio, C., Henze, N., & Patti, V. (2002). Setting up a Framework for Comparing Adaptive Educational Hypermedia: First Steps and Application on Curriculum Sequencing. In N. Henze (Ed.), Proc. of ABIS-Workshop 2002: Personalization for the mobile World, Workshop on Adaptivity and User Modeling in Interative Software Systems, (pp. 43–50). October 2002. Hannover, Germany: Learning Lab Lower Saxony (L3S).

Baldoni, M., Baroglio, C., & Marengo, E. (2007b). Curricula Modeling and Checking. In R. Basili & M.T. Pazienza (Eds.), Proc. of the 10th Congress of the Italian Association for Artificial Intelligence, AI*IA 2007, Vol. 4733 of *LNCS* (pp. 471–482). Springer.

Baldoni, M., Baroglio, C., Martelli, A., Patti, V., & Torasso, L. (2006b). Verifying the compliance of personalized curricula to curricula models in the semantic web. In M. Bouzid & N. Henze (Eds.), Proc. of the Semantic Web Personalization Workshop, held in conjunction with the 3rd European Semantic Web Conference, (pp. 53–62). Budva, Montenegro: Springer.

Baldoni, M., Baroglio, C., & Patti, V. (2004a). Web-based adaptive tutoring: an approach based on logic agents and reasoning about actions. *Artificial Intelligence Review*, *22*(1), 3–39.

Baldoni, M., Baroglio, C., Patti, V., & Torasso, L. (2004b). Reasoning about learning object metadata for adapting SCORM courseware. In L. Aroyo & C. Tasso (Eds.), AH 2004: Workshop Proceedings, Part I, International Workshop on Engineering the Adaptive Web, EAW'04: Methods and Technologies for personalization and Adaptation in the Semantic Web, (pp. 4–13). August 2004. Eindhoven, The Netherlands: Technische Universiteit Eindhoven.

Baumgartner, R., Flesca, S., & Gottlob, G. (2001). Visual Web Information Extraction with Lixto. In P.M.G. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao & R.T. Snodgrass (Eds.), VLDB (pp. 119–128). Morgan Kaufmann.

Baumgartner, R., Henze, N., & Herzog, M. (2005). The Personal Publication Reader: Illustrating Web Data Extraction, Personalization and Reasoning for the Semantic Web. In A. Gómez-Pérez & J. Euzenat (Eds.), European Semantic Web Conference ESWC 2005, (pp. 515–530). May 29. Heraklion, Greece: Springer.

Berners-Lee, T. (2000). Semantic Web - Keynote at XML 2000 Conference. http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html.

Berners-Lee, T. (2002). The Semantic Web - MIT/LCS seminar. http://www.w3c.org/2002/Talks/09-lcs-sweb-tbl/.

Bertolotti, P. (2005). *Behavioral characterization for multimedia documents*. PhD Thesis, Università degli Studi di Torino.

Brady, A., Conlan, O., Wade, V., & Dagger, D. (2008). Supporting Users in Creating Pedagogically Sound Personalised Learning Objects. In W. Nejdl, J. Kay, P. Pu & E. Herder (Eds.), Adaptive Hypermedia and Adaptive Web-Based Systems, Vol. 5149 of *LNCS* (pp. 52–61). July 2008. Springer.

Brunkhorst, I., Tönnies, S., & Balke, W.T. (2008). Multimedia Content Provisioning Using Service Oriented Architectures. In ICWS ’08: Proceedings of the 2008 IEEE International Conference on Web Services, (pp. 262–269). Bejing, China: IEEE Computer Society.

Brusilovsky, P. (2000). Course sequencing for static courses? Applying ITS techniques in large-scale web-based education. *Intelligent tutoring systems*, (pp. 625–634).

Brusilovsky, P., & Vassileva, J. (2003). Course sequencing techniques for large-scale web-based education. *Int. J. Cont. Engineering Education and Lifelong learning*, *13*(1/2), 75–94.

Cabral, L., Domingue, J., Galizia, S., Gugliotta, A., Norton, B., Tanasescu, V., et al. (2006). IRS-III: A Broker for Semantic Web Services based Applications. In I.F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold & L. Aroyo (Eds.), Proceeding of the 5th International Semantic Web Conference (ISWC 2006). Athens, USA: Springer.

Candan, K.S., Kwon, G., Peng, L., & Sapino, M.L. (2006). Modeling Adaptive Media Processing Workflows. IEEE International Conference on Multimedia and Expo, (pp. 573–576). IEEE.

Castro, C., & Manzano, S. (2001). Variable and value ordering when solving balanced academic curriculum problems. In Proc. of the 6th Workshop of ERCIM WG on Constraints. ACM.

Celentano, A., & Gaggi, O. (2003). Template-Based Generation of Multimedia Presentations. *International Journal of Software Engineering and Knowledge Engineering*, *13*(4), 419–445.

Clarke, O.E.M., & Peled, D. (2001). *Model checking*. Cambridge, MA, USA: MIT Press.

Conlan, O., Dagger, D., & Wade, V. (2002). Towards a Standards-based Approach to e-Learning Personalization using Reusable Learning Objects. In Proc. of World Conference on E-Learning, E-Learn 2002, (pp. 210–217). October 2002. Montreal, Canada: AACE.

De Coi, J.L., Herder, E., Koesling, A., Lofi, C., Olmedilla, D., Papapetrou, O., et al. (2007). A Model for Competence Gap Analysis. In J. Cordeiro, J. Filipe & S. Hammoudi (Eds.), WEBIST 2007, Proceedings of the Third International Conference on Web Information Systems and Technologies: Internet Technology / Web Interface and Applications. Barcelona, Spain: INSTICC Press.

Eisinger, N., & Maluszynski, J. (Eds.). Chapter title. *Reasoning Web, First International Summer School 2005, Tutorial Lectures*, Vol. 3564 of *Lecture Notes in Computer Science*. Msida, Malta: Springer (2005).

Emerson, E.A. (1990). Temporal and modal logic. In Handbook of Theoretical Computer Science, (pp. 995–1072). Elsevier.

European Commission, Education and Training (1999). The Bologna Process. http://europa.eu.int/comm/education/policies/educ/bologna/bologna_en.html.

Euzenat, J., & Shvaiko, P. (2007). *Ontology matching*. Heidelberg (DE): Springer-Verlag.

Farrell, R., Liburd, S.D., & Thomas, J.C. (2004). Dynamic assembly of learning objects. In S.I. Feldman, M. Uretsky, M. Najork & C.E. Wills (Eds.), Proc. of WWW 2004. ACM.

Gallardo, M.d.M., Merino, P., & Pimentel, E. (2002). Debugging UML Designs with Model Checking. *Journal of Object Technology*, *1*(2), 101–117.

Garro, A., Leone, N., & Ricca, F. (2003). Logic Based Agents for E-learning. In Proc. of the IJCAI’03 Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems, (pp. 36–45). August 2003. Acapulco, Mexico: University of Koblenz.

Garro, A., Palopoli, L., & Ricca, F. (2006). Exploiting agents in e-learning and skills management context. *AI Commun*, *19*(2), 137–154.

Guelfi, N., & Mammar, A. (2005). A Formal Semantics of Timed Activity Diagrams and its PROMELA

Translation. In Proc. of the 12th Asia-Pacific Software Engineering Conference (APSEC'05), (pp. 283–290). IEEE Computer Society.

Hackelbusch, R., & Appelrath, H. (2008). A Distributed Ontological Approach as a Basis for Software in the Context of Academic Programs. In P. Dillenbourg & M. Specht (Eds.), EC-TEL, Vol. 5192 (pp. 122–127). September 2008. Springer.

Henze, N. (2005). Personal Readers: Personalized Learning Object Readers for the Semantic Web. In 12th International Conference on Artificial Intelligence in Education, AIED05. Amsterdam, The Netherlands: IOS Press.

Henze, N., & Krause, D. (2006). Personalized Access to Web Services in the Semantic Web. In the 3rd International Semantic Web User Interaction Workshop, Athens, Georgia, USA.

Henze, N., & Nejdl, W. (2001). Adaptation in open corpus hypermedia. *IJAIED Special Issue on Adaptive and Intelligent Web-Based Systems*, *12*, 325–350.

Holzmann, G.J. (2003). *The SPIN Model Checker, Primer and Reference Manual*. Addison-Wesley.

Laborie, S. (2006). Spatio-temporal Proximities for Multimedia Document Adaptation. In J. Euzenat & J. Domingue (Eds.), Artificial Intelligence: Methodology, Systems, and Applications, Vol. 4183 of *LNCS*. Springer.

Lambert, T., Castro, C., Monfroy, E., & Saubion, F. (2006). Solving the Balanced Academic Curriculum Problem with an Hybridization of Genetic Algorithm and Constraint Propagation. In L. Rutkowski, R. Tadeusiewicz, L.A. Zadeh & J. Zurada (Eds.), Artificial Intelligence and Soft Computing, ICAISC 2006, Vol. 4029 of *LNCS* (pp. 410–419). Springer.

Learning Technology Standards Committee (2002). IEEE Standard for Learning Object Metadata. IEEE Standard 1484.12.1, Institute of Electrical and Electronics Engineers.

Lennon, J., & Maurer, H. (2003). Why it is Difficult to Introduce e-Learning into Schools And Some New Solutions. *In journal J.UCS*, *9*, 1244–1257.

Li, L., Karmouch, A., & Georganas, N.D. (1994). Multimedia teleorchestra with independent sources: Part 1 - temporal modeling of collaborative multimedia scenarios. *Multimedia Systems*, *1*(4), 143–153.

Little, T., & Ghafoor, A. (1990). Synchronization and Storage Models for Multimedia Objects. *IEEE Journal on Selected Areas in Communications*, *8*(3), 413427.

Marengo, E., *Curricula di Studi: Modelli e Verifica di Competenze*. Laurea Specialistica in Sistemi per il Trattamento dell'Informazione, Corso di Studi in Informatica, Università degli Studi di Torino (2008).

Mascardi, V., Rosso, P., & Cordì, V. (2007). Enhancing Communication inside Multi-Agent Systems. An Approach based on Allignment via Upper Ontologies. In M. Baldoni, C. Baroglio & V. Mascardi (Eds.), Proceedings of MALLOW-AWESOME'007. First Workshop on Agents, Web-Services, and Ontologies, (pp. 92–107). Durham University.

Melia, M., & Pahl, C. (2006). Automatic Validation of Learning Object Compositions. Information Technology and Telecommunications Conference IT&T'2005: Doctoral Symposium. Carlow, Ireland: TecNet.

Melia, M., & Pahl, C. (2007a). An Information Architecture for Validating Courseware. In Proc. of LODE. CEUR-WS.org.

Melia, M., & Pahl, C. (2007b). Pedagogical Validation of Courseware. In E. Duval, R. Klamma & M. Wolpers (Eds.), Proc. of EC-TEL, Vol. 4753 of *LNCS* (pp. 499–504). Springer.

Mohan, P., & Brooks, C. (2003). Learning Object on the Semantic Web. In D.S. V. Devedzic J. Spector & Kinshuk (Eds.), Proc. of the 3rd IEEE International Conference on Advanced Learning Technologies. Athens, Greece: IEEE.

OMG (2007). Unified Modeling Language: Superstructure, version 2.1.1. OMG, Object Management Group.

Paschke, A., & Biletskiy, Y. (Eds.). (2007). *Advances in Rule Interchange and Applications*. Vol. 601 of *LNCS*. Springer.

Pazzi, L. (1998). Three Points of View in the Characterization of Complex Entities. In N. Guarino (Ed.), *Formal Ontology in Information Systems*. IOS Press.

Persico, D. (1996). Courseware validation: a case study. *Journal of Computer Assisted Learning*, *12*, 232–244.

Pistamiglio, O., *Sviluppo di un plugin grafico per Eclipse: DCML Designer*. Laurea specialistica in informatica, Corso di Studi in Informatica, Università degli Studi di Torino (2007).

REWERSE (2008). Reasoning on the Web with Rules and Semantics. `http://www.rewerse.net`.

Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., et al. (2005). Web Service Modeling Ontology. *Applied Ontology*, *1*(1), 77–106.

Rosmalen, P.V., Vogten, H., Es, R., Passier, H., Poelmans, P., & Koper, R. (2006). Authoring a full life cycle model in standards-based, adaptive e-learning. *Journal of Educational Technology and Society*, *9*(1), 72–83.

Russell, S., & Norvig, P. (2003). *Artificial Intelligence, A Modern Approach, Second Edition*. Prentice Hall Series in Artificial Intelligence.

Samples, J.W. (2002). The pedagogy of technology - our next frontier?. *Connexions*, *14*(2), 4–5.

Schaffert, S., & Bry, F. (2002). A gentle introduction to Xcerpt, a rule-based query and transformation language for XML. In M. Schroeder & G. Wagner (Eds.), RuleML, Vol. 60. CEUR-WS.org.

Stojanovic, L., Staab, S., & Studer, R. (2001). eLearning based on the Semantic Web. In W.A. Lawrence-Fowler & J. Hasebrook (Eds.), WebNet2001 - World Conference on the WWW and Internet. Orlando, Florida, USA: AACE.

Terenziani, P., Giordano, L., Bottrighi, A., Montani, S., & Donzella, L. (2006). SPIN Model Checking for the Verification of Clinical Guidelines. In G. Brewka, S. Coradeschi, A. Perini & P. Traverso (Eds.), Proc. of ECAI 2006 Workshop on AI techniques in healthcare: evidence-based guidelines and protocols. August 2006. Riva del Garda: IOS Press.

van der Aalst, W.M.P., & Pesic, M. (2006). DecSerFlow: Towards a Truly Declarative Service Flow Language. In M. Bravetti & G. Zavattaro (Eds.), Proc. of WS-FM, LNCS, September Vienna: Springer.

Vassileva, J. (1992). Dynamic CAL-Courseware Generation Within an ITS-Shell Architecture. In I. Tomek (Ed.), Computer Assisted Learning, Vol. 601 of *LNCS* (pp. 581–591). Springer.

W3C (2001). Synchronized Multimedia Integration Language (SMIL 2.0) Specification.

W3C (2004b). OWL Web Ontology Language Semantics and Abstract Syntax. Available at http://www.w3.org/TR/owl-semantics/.

W3C (2004a). RDF Primer. Available at http://www.w3.org/TR/rdf-primer/.

Wu, K., & Havens, W.S. (2005). Modelling an Academic Curriculum Plan as a Mixed-Initiative Constraint Satisfaction Problem. In B. Kégl & G. Lapalme (Eds.), Canadian Conference on AI, Vol. 3501 of *LNCS* (pp. 79–90). Springer.