Reasoning-Based Curriculum Sequencing and Validation: Integration in a Service-Oriented Architecture

Matteo Baldoni[1], Cristina Baroglio[1], Ingo Brunkhorst[2], Elisa Marengo[1], and Viviana Patti[1]

[1] Dipartimento di Informatica, Università degli Studi di Torino
C.so Svizzera, 185 — I-10149 Torino, Italy
{baldoni,baroglio,patti}@di.unito.it,elisa.mrng@gmail.com
[2] L3S Research Center, University of Hannover, D-30539 Hannover, Germany
brunkhorst@l3s.de

Abstract. We present a service-oriented personalization system, set in an educational framework, based on a semantic annotation of courses including prerequisites and learning objectives. The system supports users in planning personalized curricula and in verifying the compliance of curricula against a model describing the designer goals. We have developed a prototype of the planning and validation services, by using SWI-Prolog and the SPIN model checker as reasoning engines. The services are supplied and combined in the Personal Reader framework.

1 Introduction

So far, reasoning in the Semantic Web is mostly reasoning about knowledge expressed in some ontology. However, personalization may involve also other kinds of reasoning and knowledge representation. Moreover, the next Web generation promises to deliver Semantic Web Services, that can be retrieved and *combined* in a way that satisfies the user. It opens the way to many forms of *service-oriented personalization*. Web services provide an ideal infrastructure for enabling *interoperability* among personalization applications and for constructing Plug&Play-like environments, where the user can select and combine the preferred kinds of services. Based on our previous work [2,3], our current goal is to implement such results in an organic system, where different reasoning-based personalization services can be combined for supporting the user in building a curriculum from a set of *learning resources* (courses). We achieve this by developing a Planner and a Validation Service within the Personal Reader(PR) Framework, where a service oriented approach to personalization is taken [12].

While in early times learning resources were simply considered as "contents", strictly tied to the platform used for accessing them, attention has been posed on the issue of *re-use* and of a *cross-platform* use of educational contents. The proposed solution is to adopt a *semantic annotation* of contents based on standard languages, e.g. RDF and LOM, and then to create learning resources formed by *educational contents* plus *semantic meta-data*, which supply information on the resources at a *knowledge level* (e.g. concepts from an ontology of the educational domain). Learning Resources are interpreted as *actions* [3,4], capturing the *learning objectives* and *pre-requisites*. Thus, we

E. Duval, R. Klamma, and M. Wolpers (Eds.): EC-TEL 2007, LNCS 4753, pp. 426–431, 2007.

can rely on a classical theory of actions and apply different reasoning methods, like planning, for building personalized curricula. Our interpretation of learning resources also enables the use of model checking techniques for developing a validation service that detects if a curriculum is compliant w.r.t an abstract model, which encodes the *curricula-design goals*.

Motivating Scenario. Curriculum planning and validation offer useful support in many practical contexts for helping both students and teaching institutions. Since taking courses at different Universities is becoming more common in Europe, creating a personalized curriculum becomes a difficult task for students. Even if the students know what competency they would like to acquire, it's harder to find the courses that help to acquire it: an automatic system that can suggest a pathway through the course repository can be very helpful. The need for support in building personalized paths through learning resources has to be *combined* with the ability to ensure the compliance of the resulting curriculum with *curricula-design goals*, expressed by the teachers or by the *institution* offering the courses. Curricula models specify general rules for building learning paths, e.g. constraints designed by the University for guaranteeing the achievement of certain learning goals. These constraints are to be expressed in terms of knowledge elements, and maybe also on features that characterize the resources. For a provider or university, which needs to certify that a specific offered curricula for achieving a certain educational goal respects some European guidelines, we could define the guidelines as a set of constraints at an abstract level, i.e. as relations among a set of competencies which should be offered in a way that meets some given scheme. The automatic checking of compliance combined with curriculum planning could be used for implementing processes like cooperation among institutes in curricula design and integration, which are the focus of the *Bologna Process* [11], promoted by the EU.

2 Curricula Representation and Reasoning

All the different kinds of objects that we need to tackle (learning resources, curricula, and curricula models) are described on the basis of a set of *competencies*, i.e. terms identifying specific *knowledge elements*. Competencies can be thought of, and implemented, as concepts in a shared ontology. In our implementation, competencies have been semi-automatically extracted, and then stored in a RDF file (see the next section).

Learning Resources and Curricula. A *curriculum* is a sequence of *learning resources* that are homogeneous in their representation. Based on work in [3,4], we rely on an *action theory*, and take the abstraction of resources as *atomic actions*. A learning resource is modelled as an action for acquiring some competencies, called *effects*. For understanding the contents supplied by a learning resource, the user can be required to own other competencies, called *preconditions*. Both preconditions and effects can be expressed by means of a *semantic annotation* of the learning resource [4]. Since we will focus on university curricula, we will refer to learning resources as "courses". Given the above interpretation of learning resources, a *curriculum* is modelled as a *plan*, i.e. as a sequence of actions, whose execution causes transitions from a state to another, until some final state is reached. The *initial state* (possibly empty) contains all the

competences that we suppose available before the curriculum is taken, e.g. the knowledge that the student already has. This set typically *grows* as the student studies and learns. Curricula are usually designed so to allow the achievement of a *learning goal*; in such cases the *final state* should contain specific knowledge elements, e.g. all those that compose the user's learning goal. A transition between two states is due to the application of the action corresponding to a learning resource. For an action to be applicable, its preconditions must hold in the state to which it should be applied. The application of the action consists in an *update* of the state. We assume that competences can only be added to states. The intuition behind this assumption is that the act of using a new resource will never erase from the students' memory the concepts acquired insofar.

Curricula Models. We would like to restrict the set of possible sequences of resources composing a curriculum, by imposing constraints on the *order* by which knowledge elements are added to the states, e.g. "a knowledge element α is to be acquired before a knowledge element β", or by specifying some *educational objective* to be achieved, in terms of knowledge that must be contained in the final state, e.g. "a knowledge element α must be acquired sooner or later". Therefore, we represent a *curricula model* as a set of temporal constraints building upon knowledge elements. A model is independent from the available resources and it can be reused in different contexts. A natural choice for representing temporal constraints on action paths is linear-time temporal logic (LTL) [10]. This kind of logic allows to verify if a property of interest is true for all the possible executions of a model (in our case the specific curriculum). This is often done by means of model checking techniques [8]. A curriculum as we represent it is, actually, a Kripke structure, that identifies a set of states with a transition relation for passing from a state to another. Since in our domain we assume that knowledge only grows, states will always contain *all* the competencies acquired up to that moment. The transition relation is given by the actions that are contained in the curriculum that is being checked. The LTL logic can be used to verify if a given formula holds starting from a state or if it holds for a set of states. For instance, in order to specify in the curricula model constraints on *what* to achieve, we can use the formula $\Diamond\alpha$ ($\Diamond$ is the *eventually* operator), meaning that a set of knowledge elements will be acquired sooner or later. Instead, constraints on *how* to achieve the educational objectives, such as "a knowledge element β cannot be acquired before the knowledge element α is acquired", can be expressed by the LTL formula $\neg\beta\ U\ \alpha$, where U is the *weak until* operator. Writing curricula models directly in LTL is not an easy task for the user. We are developing a graphical language, called DCML (Declarative Curricula Model Language) [5], inspired by DecSerFlow [15]. By means of DCML the user can easily write curricula models, maintaining a rigorous meaning due to the logic grounding of the language.

Curriculum Planning and Validation. Given a semantic annotation with preconditions and effects of the courses, classical planning techniques are exploited for creating *personalized curricula*, in the spirit of the work in [3,4]. Intuitively the idea is that, given a repository of annotated learning resources the user expresses a *learning goal* as a set of *knowledge elements* he/she would like to acquire, and possibly also a set of already owned competencies. Then, the system applies planning to build a sequence of learning resources that will allow him/her to achieve the goal. The planning methodology that

we implemented (see Section 3) is a simple *depth-first forward planning* where actions cannot be applied more than once. An early prototype was presented in [1].

There are two main validation tasks that can be performed on curricula and curricula models. The simplest one consists in *checking the soundness* w.r.t. the learning dependencies and the learning goal of curricula which are built *by hand* by users themselves. Usually, soundness verification is performed manually by the learning designer, with hardly any guidelines or support [9]. Not all sequences which can be built starting from a set of learning resources are lawful. It is important to verify that all the *competencies*, that are necessary to fully understand the contents, offered by a learning resource, are introduced or available before that resource is accessed. In other words, a course can appear at a certain point in a sequence only there are no *competency gaps*. These implicit "applicability constraints" capture dependencies that are innate to the nature of the taught concepts. Given the interpretation of resources as actions, the verification of the *soundness of a curriculum*, w.r.t. the learning dependencies and the learning goal, can be interpreted as an *executability check* of the curriculum.

The other interesting verification task consists in checking if a curriculum (possibly automatically generated by a personalization service) is *compliant against the course design goals* [7]. A curriculum personalized w.r.t. the user desires, that is proved to be sound, cannot automatically be considered as being *valid* w.r.t. a particular *curricula model* describing some designer goal. The curricula model imposes further constraints on *what* to achieve and *how* achieving it. In our validation service (Section 3) the verification tasks are performed by using the SPIN model checker [13]. SPIN is used for verifying systems that can be represented by *finite state structures*, where the specification is given in an LTL logic. The verification algorithm is based on the exploration of the state space. This is exactly what we need for performing all the verification tests that we mentioned, provided that we can translate the curriculum in the internal representation used by the model checker (in SPIN such representation is given in Promela).

3 Implementation in the Personal Reader Framework

The Personal Reader Platform provides a framework for implementing personalization in the Semantic Web in a service-oriented approach, allowing to investigate how (semantic) web service technologies can provide a suitable infrastructure for building personalization applications. So called *Personalization Services* (PServices) [12] are the basic building blocks for implementing plug-and-play like personalization services in this architecture, they are semantic in the sense that they communicate solely on the basis of *RDF* documents. Besides PServices, the PR framework also includes other kinds of components, namely *Syndication Services*, *User Interfaces* and a *Connector*.

Corpus of Courses and Metadata Description. Despite some manual post-processing for fixing inconsistencies, we extracted a corpus of courses and the related meta-data by extracting real data from the Hannover University database via an automatic extraction with the Lixto [6] tool. We focussed only on a subset of the courses and manually post-processed the data, resulting in corpus with 65 courses, with 390 effects and 146 preconditions. Metadata contains also course names, semester, credit points, the type of course (e.g. laboratory, etc.), schedule and location.

Reasoners as PServices. We implemented two independent PServices for our system, the "Curriculum Planning PService", and the "Curriculum Validation PService" (Fig. 1), whic can be used by other applications as well.

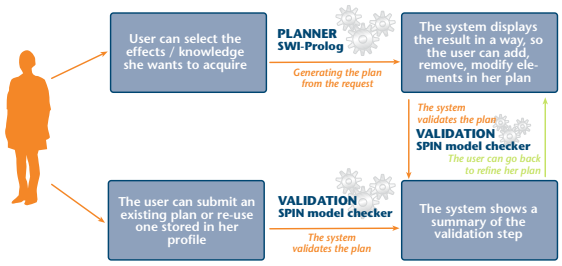

Fig. 1. The interaction with the system

The Curriculum Planning PService is basically divided in two parts: the core reasoner (the planner) and the wrapper (the web service implementation) interfacing with the PR framework. The reasoning engine that actually accomplish the curriculum planning task has been implemented in SWI Prolog by using a classical depth-first search algorithm. The initial state is set by using information about the user's context provided by the User Modelling module of the PR. SWI Prolog 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 PR. The request document contains: a) links to the RDF document containing the database of courses, annotated with metadata, b) a reference to the user's context c) the user's actual learning goal, i.e. a set of knowledge concepts that the user would like to acquire, and that are part of the *domain ontology* used for the semantic annotation of the actual courses. 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. The reasoner returns as output a *RDF response document*, which contains a list of plans that fulfill the user's learning goals and profile. Information stored in the user profile is used for *ranking higher* those plans that include the user's preferred topics.

An early prototype[1] of the Curriculum Validation PService based on the model checker SPIN has been designed and is currently being embedded in the PR. Model checking is the algorithmic verification of the fact that a finite state system complies to its specification. In our case the specification is given by the curricula model and consists of a set of temporal constraints, while the finite state system is the curriculum to be verified. The advantage of using a model checker like SPIN, rather than an ad hoc implementation is that it can handle any kind of LTL temporal formula. Moreover, we can also deal with the validation of *non-linear* curricula, i.e. curricula that contain branching points. This kind of curricula allow to account for *uncertainties* of the user. In fact a branching point corresponds to a possible choice among alternative resources.

[1] `http://www.l3s.de/~brunkhor/semweb/curriculum/`

4 Conclusion

In this work we have sketched the current state of the integration of semantic personalization web services for Curriculum Planning and Validation within the Personal Reader Framework. We are actually investigating how to extend the application with a module of geo-spatial reasoning working on meta-data like floor-plans and locations.

In [14] an analysis of pre- and post-requisite annotations of learning object is proposed with the aim of dealing with competency gap verification. In this approach, whenever an error will be detected by the validation phase, a correction engine will be activated, that produce suggestions on how to correct the wrong curriculum, by using reasoning-by-cases. The suggestions are presented to the course developer, who can decide which ones to adopt. Once a curriculum have been corrected, it must be validated again: the corrections might introduce errors. The proposal is inspired by the CocoA system [7], that allows to perform the consistency check of web-based courses.

References

1. Baldoni, M., Baroglio, C., Brunkhorst, I., Henze, N., Marengo, E., Patti, V.: A Personalization Service for Curriculum Planning. In: Proc. of the 14th Workshop on Adaptivity and User Modeling in Interactive Systems, ABIS 2006, pp. 17–20. Hildesheim, Germany (2006)
2. Baldoni, M., Baroglio, C., Martelli, A., Patti, V., Torasso, L.: Verifying the compliance of personalized curricula to curricula models in the semantic web. In: Proc. of the Semantic Web Personalization Workshop, pp. 53–62, Budva, Montenegro (2006)
3. Baldoni, M., Baroglio, C., Patti, V.: Web-based adaptive tutoring: An approach based on logic agents and reasoning about actions. Artificial Intelligence Review 1(22), 3–39 (2004)
4. Baldoni, M., Baroglio, C., Patti, V., Torasso, L.: Reasoning about learning object metadata for adapting SCORM courseware. In: Proc. of EAW'04, pp. 4–13 (2004)
5. Baldoni, M., Marengo, E.: Curricula model checking: declarative representation and verification of properties. In: Proc. of EC-TEL'07. LNCS, Springer, Heidelberg (2007)
6. Baumgartner, R., Flesca, S., Gottlob, G.: Visual web information extraction with Lixto. In: VLDB, pp. 119–128. Morgan Kaufmann, San Francisco (2001)
7. Brusilovsky, P., Vassileva, J.: Course sequencing techniques for large-scale web-based education. Int. J. Cont. Engineering Education and Lifelong learning 13(1/2), 75–94 (2003)
8. Clarke, O.E.M., Peled, D.: Model checking. MIT Press, Cambridge, MA, USA (2001)
9. De Coi, J.L., Herder, E., Koesling, A., Lofi, C., Olmedilla, D., Papapetrou, O., Sibershi, W.: A model for competence gap analysis. In: Proc. of WEBIST 2007 (2007)
10. Emerson, E.A.: Temporal and modal logic. Handbook of Theoretical Computer Science B, 997–1072 (1990)
11. European Commission, Education and Training. The Bologna process, `http://europa.eu.int/comm/education/policies/educ/bologna/bologna_en.html`
12. Henze, N., Krause, D.: Personalized access to web services in the semantic web. In: The 3rd Int. Semantic Web User Interaction Workshop, SWUI (November 2006)
13. Holzmann, G.J.: The SPIN Model Checker. Addison-Wesley, Reading (2003)
14. Melia, M., Pahl, C.: Automatic Validation of Learning Object Compositions. In: Information Technology and Telecommunications Conf. IT&T'2005: Doctoral Symposium (2006)
15. van der Aalst, W.M.P., Pesic, M.: DecSerFlow: Towards a Truly Declarative Service Flow Language. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) WS-FM 2006. LNCS, vol. 4184, Springer, Heidelberg (2006)