
# A3-D8

# Testbeds: overview and state

| | |
|---|---|
| Project title: | Reasoning on the Web with Rules and Semantics |
| Project acronym: | REWERSE |
| Project number: | IST-2004-506779 |
| Project instrument: | EU FP6 Network of Excellence (NoE) |
| Project thematic priority: | Priority 2: Information Society Technologies (IST) |
| Document type: | D (deliverable) |
| Nature of document: | R (report) |
| Dissemination level: | PU (public) |
| Document number: | IST506779/Turin/A3-D8/D/PU/a1 |
| Responsible editors: | Cristina Baroglio |
| Reviewers: | Massimo Marchiori, Alberto Martelli |
| Contributing participants: | Hannover, Malta, Turin |
| Contributing workpackages: | A3 |
| Contractual date of deliverable: | 28 February 2007 |
| Actual submission date: | 27 February 2007 |

**Abstract**
This deliverable reports about the state of advancement of the testbeds, that have been developed within the working group A3. In particular, we will describe the most recent improvements of the Personal Reader framework (joint effort of Hannover and Turin) and of the PreDiCtS framework (Malta). A connection between the two frameworks is their relation to the emerging technology of web services.

**Keyword List**
semantic web, reasoning, web services, automatic retrieval and composition, temporal logics, recommendation, planning and verification

Project co-funded by the European Commission and the Swiss Federal Office for Education and Science within the Sixth Framework Programme.

# Testbeds: overview and state

**Charlie Abela[3] and Matteo Baldoni[1] and Cristina Baroglio[1] and Nicola Henze[2] and Ingo Brunkhorst[2] and Daniel Krause[2] and Elisa Marengo[1] and Viviana Patti[1]**

[1] Dipartimento di Informatica, Università degli Studi di Torino
Email: {baldoni,baroglio}@di.unito.it

[2] L3S Research Lab, University of Hannover
Email: {brunkhorst,krause}@l3s.de

[3] Department of Computer Science and Artificial Intelligence, University of Malta
Email: charlie.abela@um.edu.mt

27 February 2007

**Abstract**
This deliverable reports about the state of advancement of the testbeds, that have been developed within the working group A3. In particular, we will describe the most recent improvements of the Personal Reader framework (joint effort of Hannover and Turin) and of the PreDiCtS framework (Malta). A connection between the two frameworks is their relation to the emerging technology of web services.

**Keyword List**
semantic web, reasoning, web services, automatic retrieval and composition, temporal logics, recommendation, planning and verification

# Contents

## Executive summary

This deliverable reports about the most recent achievements on the issue of testbeds, carried on within the working group A3, with a particular attention to application systems, which have been or are being developed.

We will describe systems that are part of two frameworks:

1. *Personal Reader*: the Personal Reader framework itself is described in deliverable A3-D9, so after a short introduction, here we will focus on two adaptive functionalities, implemented as web services in the Personal Reader framework, namely a *curriculum planning* service and a *curriculum validation* service;

2. *PreDiCtS*: PreDiCtS is a framework developed to allow for the the retrieval of services based on composition patterns, which exploits past experience for better fitting the user's desires. After an introduction to the framework itself, we will focus on ontology-based case definition, case creation and retrieval, and planning-based composition.

The integration of the two frameworks will be part of future work.

# 1 Introduction

This deliverable reports about the most recent achievements on the issue of testbeds, carried on within the working group A3, with a particular attention to application systems, which have been or are being developed.

The systems that we will describe are part of two frameworks which will also be introduced, namely the *Personal Reader* and *PreDiCtS*. Both frameworks are related to *web services*, in fact, the personal Reader has been recently redesigned as a *web service-oriented* system, while PreDiCtsS is a framework for *retrieving* and *composing web services* in an automatic way. The relevance of the use of web services and of the capability of dealing with web services were identified as a strategic and promising direction to follow in deliverable A3-D6.

This deliverable witnesses the advancement of the works in the following way. The next sections report brief descriptions of the objectives that have been pursued and how those ideas have been or are being implemented in a collection of systems. For each system we report the objective, the main characteristics, the design issues, and the state of development.

Detailed technical descriptions are, instead, contained in scientific articles, that were added to this deliverable as an Appendix. Such articles either appeared in the proceedings of international conferences/workshops or in international journals. For those parts of the works which are under development, we will refer to technical reports, regularly registered at our institutes.

The deliverable is organized as follows. Section 2 reports the achievements concerning the Personal Reader framework, while Section 3 concerns the achievements in the PreDiCtS framework. Section 4 draws conclusions and future research lines. The deliverable is concluded by two Appendixes, the first one reports the DOAP descriptions for the described application systems, the second instead contains the collection of scientific papers, reporting the details of the described systems.

# 2 The Personal Reader

The Personal Reader (PR) framework offers an environment for designing, implementing and realizing Web content readers in a service-oriented approach (see [Henze and Krause, ]). It is characerized by a service oriented architecture to allow personalization in a *plug-and-play* way, via the use of so called Personalization Services. Each personalization service offers a personalization functionality, for instance aimed at producing recommendations which are tailored to the needs of specific users, pointers to information which is deemed as being related to the interests of the user, more detailed or more general than the one being displayed, and so forth. Personalization services are semantic in the sense that they communicate solely on the basis of *RDF* documents. Besides personalization services, the PR framework also includes other kinds of components, namely *Syndication Services*, *User Interfaces* and a *Connector*.

1. The *Connector* is a centralized and unique component, which offers mainly two functionalities:

   - *Providing a repository of available Personalization and Syndication Services.* This respository is similar to an UDDI repository, but enables a better scalability as it aids SynServices finding and selecting the appropriate PServices.
   - *Handling the communication between the Personalization and Syndication Services.* The user can define if some services should be excluded from each other. As the

Connector will not list the real URIs of the services stored in its respository it forced the services to use the Connector to perform the communication.

2. *User interfaces* visualize RDF data which are delivered by the Syndication Services. These RDF data are adopted according to a specific device. Hence, there are different user interfaces available for one application.

3. *Syndication services* (SynServices for short) are used to deliver RDF data to user interfaces. Therefore, SynServices receive events from the user interfaces, which are generated by the user interaction. In the SynService, these events are processed by a state machine. According to the current state, this machine calls SynService-specific methods, which implement the application logics. Normally, such a method behaves in the following way:

   (a) Search for appropriate PServices by asking the Connector

   (b) Create an RDF invoke request and invoke the received PServices

   (c) Syndicate and postprocess the responses from the PServices

   (d) Create the RDF response and send it to the user interface

   The implementation of a SynService is very flexible in terms of reaction of the provided PServices. If some required functionalities are not available (c.f. the corresponding PService has been deleted) the SynService tries to invoke other PServices with a lower functionality or skips this part of it's result generation process. The user interfaces are also keep flexible so that they can handle also subsets of RDF responses and visualize the remaining data. The Syndication process itself will be described in more detail in the deliverable A3-D9.

The personalization services that we describe in Sections 2.1 and 2.2 have been designed for handling *learning resources*. By the term *learning resource* we mean a particular kind of resource, used in educational frameworks.

While in early times learning resources were simply considered as "contents", strictly tied to the platform used for accessing them, recently, greater and greater attention has been posed on the issue of *re-use* and, related to this, also on the problem to allow a *cross-platform* use of educational contents, i.e. on the separation of the contents from the *means* that is necessary for taking advantage of it. The proposed solution is to adopt a *semantic annotation* of contents based on standard languages. Some proposal of languages to be used to this aim are RDF [W3C, ] and LOM [LOM, 2002].

Therefore, in this deliverable we consider a *learning resource* as formed by *educational contents* plus *semantic meta-data*. By meta-data we supply information on the learning resources at a *knowledge level*, i.e. on the basis of concepts taken from an ontology that describes the educational domain. In the personalization services that are described hereafter, we rely on the interpretation of learning resources as *actions*. for this reason, the meta-data captures the *learning objectives* of the learning resource and its *pre-requisites*. Intuitively, pre-requisites capture the set of competences that the learner should have to profitably use a reasource; by using it, the learner will acquire a new set of competences, which correspond to the learning objectives. Following the action metaphor, we can interpret the pre-requisites as the preconditions and the learning objectives as the effects of the learning resource.

Learning resources can either be web documents, book chapters, multi-media contents and the like, or they can be lectures and courses. The personalization services described below can

be applied independently from the specific type of the contents but in the application scenarios that we will consider we will refer to university courses.

As we have theoretically shown in previous work [Baldoni et al., 2004d, Baldoni et al., 2004a, Baldoni et al., 2004b], given an annotation of resources with preconditions and effects one can rely on a classical theory of actions and apply different reasoning techniques for building personalized curricula. By curriculum we hereby mean a sequence of learning resources, aimed at acquiring some competency (also known as *learning goal*).

The modeling of learning resources as actions also allows the development of verification services that can automatically detect if a user-given curriculum is compliant w.r.t a model given as a set of constraints, as we will explain below.

## 2.1 The curriculum planning service

### Motivation to building the service

Often a student knows what competency he/she would like to acquire but has no knowledge of which courses will help him/her acquiring it. An automatic system for building a curriculum is very helpful in this case, because it can suggest a pathway that is built taking into account the specific interests of the student and his/her initial knowledge (which can either be deduced from a set of previously used learning resources or can be directly declared by the user).

### Description

A *curriculum* is here interpreted as a sequence of learning resources, that are homogeneous in their representation. Given an action-based representation of the learning resources, a curriculum can be interpreted as a sequence of actions, whose execution causes transitions from a state to another, until some final state is reached. In general, curricula are supposed to allow the achievement of a given *learning goal*. A learning goal is a set of knowledge elements of interest, those that a student, following a curriculum, would like to acquire. We would like such elements to be contained in the final state reached by attending the curriculum. The *initial state*, instead, represents the initial set of competencies that we suppose as being available before the curriculum is taken. In other words, they capture the knowledge that the student already has. This set can possibly be empty.

Given a semantic annotation with preconditions and effects of the courses, classical planning techniques are exploited for *composing* personalized curricula, in the spirit of the work in [Baldoni et al., 2004b, Baldoni et al., 2004e]. Intuitively the idea is the following: given a repository of learning resources, which have been semantically annotated as described, 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.

**State of development**: The Curriculum Planning Service has been implemented and integrated as a new plug-and-play personalization service in the Personal Reader framework.

### Implementation and integration in the Personal Reader Framework

The curriculum planner is implemented in Prolog. The planning system is implemented as a Personalization Service in the Personal Reader. Figure 1 reports the overall structure of such Personalization Service, which is basically divided in two parts: a reasoner (Prolog planner)

and a wrapper (web service implementation). The web service implementation is the *interface* of the Personalization Service. This interface, which has been defined according to the standard for the Personal Reader framework (see [Henze and Krause, ]), allows for the processing of RDF documents and for inquiring (from outside) about the service capabilities. The service can, in fact, be accessed by Syndication Services, used for discovery and invocation via the central Connector component.

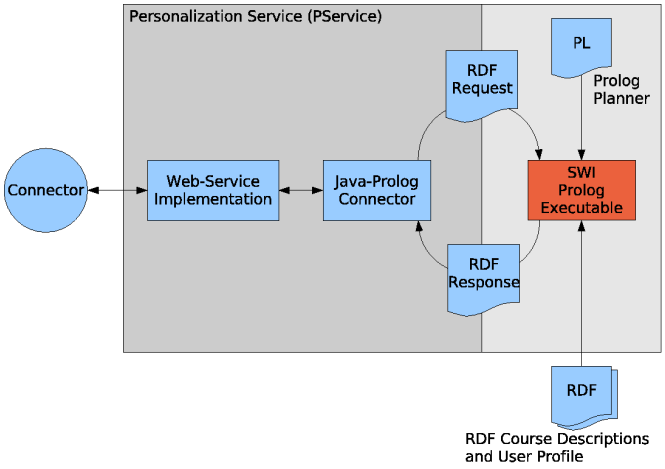


Figure 1: Curriculum Planning Web Service.

When invoked, the *Java-to-Prolog Connector* runs the SWI-Prolog executable in a subprocess; essentially it passes the RDF document containing the request *as-is* to the Prolog system, and collects the results, represented as RDF documents as well.

The curriculum planning task is accomplished by a reasoning engine, which has been implemented in SWI Prolog[1]. The interesting thing of using SWI Prolog is that it contains a semantic web library for dealing 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:

1. links to the RDF document containing the meta-data of a repository of learning resources;
2. a reference to the user's context;
3. the user's 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.

### List of attached articles

Technical details about the curricula planning web service can be found in:

[1] `http://www.swi-prolog.org/`

M. Baldoni, C. Baroglio, I. Brunkhorst, N. Henze, E. Marengo, and V. Patti. *A Personalization Service for Curriculum Planning.* In E. Herder and D. Heckmann, editors, Proc. of the 14th Workshop on Adaptivity and User Modeling in Interactive Systems, ABIS 2006, Hildesheim, Germany, October 2006.

## 2.2 The validation service

### Motivation to building the service

The automatic checking of compliance combined with curriculum sequencing techniques could be used for implementing processes like cooperation among institutes in curricula design and integration, which are actually the focus of the so called *Bologna Process*, promoted by the EU ministers responsible for higher education[2] [European Commission and Training, ],.

### Description

The validation service is the most recent advancement of research work, concerning the application of techniques for knowledge representation and for automatic reasoning to educational domains. The early stages of this work have been published by the Artificial Intelligence Review journal [Baldoni et al., 2004c]. The approach followed in that work consisted in adding a further level of representation onto the level of the learning resources (also in that case learning resources were university courses): the level of the *curriculum schema*. Curriculum schemas were defined as abstract representations, capturing the structure of a curriculum in terms of knowledge elements, and they were implemented by means of prolog-like logic clauses. Curriculums schemas were used to perform a procedure-driven form of planning, which aimed at identifying an execution of the clauses (a specific curriculum) that, on the one hand, allowed the user to reach a learning goal of interest from an initial state, capturing his/her initial knowledge. The advantage of adopting procedure-driven planning techniques is that the only curricula that are tried are the possible executions of the procedure itself, and this restricts considerably the search space.

In this context, it is also possible to apply forms of verification. In particular, to check whether a curriculum given by a user is an instance of a given schema. This approach, however, is limitating because procedure clauses, besides having a *prescriptive* nature, pose very strong constraints on the sequencing of learning resources. In particular, clauses represent what is "legal" and whatever sequence is not foreseen by the clauses is "illegal". However, in an open environment where resources are extremely various, they are added/removed dynamically, and their number is huge, this approach becomes unfeasible: the clauses would be too complex, it would be impossible to consider all the alternatives and the clauses should change along time.

For this reason it is appropriate to take another perspective and represent only those constraints which are strictly necessary, as suggested by the so called *social approach* proposed by Singh for describing communication protocols for multi-agent systems [Singh, 1998, Singh, 2000]. In this approach only the *obligations* are represented. In our application context, obligations capture relations among the times at which different pieces of knowledge are to be acquired. The advantage of this representation is that we do not have to represent all that is legal but only those *necessary conditions* that characterize a legal solution. To make example,

[2] "Curriculum design means drawing up of a common study path aimed at reaching the educational goals that have been jointly defined. In these schemes the partners offer specific segments which complement the overall curriculum designed" [European Commission and Training, ].

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. If we used the clause-based approach, instead, we should have described also what can legally be contained between the two times at which the two pieces of knowledge are acquired. Generally, the constraints-based approach is more flexible and more suitable to an open environment.

Summarizing, we have adopted a constraint-based representation of curricula models. Constraints are expressed as formulas in a temporal logic (LTL, linear-time logic [Emerson, 1990]). This kind of logic allows the verification that a property of interest is true for all the possible executions of a model, which in our case corresponds to the specific curriculum. This is often done by means of model checking techniques [Clarke and Peled, 2001]. The curriculum that we mean to check is, indeed, a Kripke structure; as thus, it is easy to verify properties expressed as temporal logic formulas. Briefly, a Kripke structure identifies a set of states and transition relation that allows passing from a state to another. In our case, the states correspond to the competencies that are owned at a certain moment. Since we assume the domain is monotonic in the sense pointed out in the previous subsection, states will 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. Since the sequence is linear and shows no branch, then, it is possible to reason on the states and with LTL logic it is possible to verify that a given formula holds starting from a state or that it holds for a set of states.

For example, the fact that a knowledge element $\beta$ cannot be acquired before the knowledge element $\alpha$ is acquired, can be written as the LTL temporal formula $\neg\beta \; U \; \alpha$, where $U$ is the *weak until* operator. Given a set of knowledge elements to be acquired, such constraints specify a partial ordering of the same elements. Other kinds of constraints might be taken into account. For instance, that a knowledge element will be acquired sooner or later ($\Diamond\alpha$, eventually operator).

**State of development**: The Curriculum Validation Service has been designed and is currently being integrated as a new plug-and-play personalization service in the Personal Reader framework.

#### Implementation and integration in the Personal Reader Framework

The service is being implemented. It will soon be integrated in the framework following a schema that is analogous to the one descibed for the Curriculum Planning service.

#### List of attached articles

Technical details about curricula validation can be found in [Baldoni et al., 2006] and in:

> Matteo Baldoni, Cristina Baroglio, Ingo Brunkhorst, Elisa Marengo, Viviana Patti, A Personalization Web Service for Curricula Planning and Validation. Poster paper at the European Semantic Web Conference (ESWC), 2007.

## 3 PreDiCtS

PreDiCtS (stands for Personalised Discovery and Composition of Services) is a framework, see Figure 2 which provides for the retrieval of services based on service-composition templates. The philosophy behind PreDiCtS is based on three fundamental assumptions:

1. the service-retrieval process is considered as the process that encapsulates both service discovery and/or service composition.

2. it is important to make use of past experience to ease the service-retrieval process

3. it is important that the system identifies as clearly as possible the user's service requirements.

4. having identified both the user-required functionality and a set of suitable templates the system can start searching for services that fit into these templates.

The first assumption is motivated by the fact that composition of services comes into play whenever the service discovery process is unable to find a service that fulfils the user's requirements. At which point, the retrieval system has to identify which services, atomic or complex (a service made up of other services), can be combined together to provide a complete solution.

The second assumption allows us to consider past experiences of service-retrieval and to represent them through templates that capture generic aspects of specific service composition instances. This approach is similar to that adopted in Rajasekaran [Rajasekaran et al., 2004], Sirin [Sirin et al., 2005], Deelman [Deelman, 2003] and Weber [Weber et al., 2005], which use pre-stored abstract workflow definitions (or templates) in their composition framework. This presents the advantage that a new service-retrieval query aimed at discovering new functionalities, does not necessarily lead to a service-composition process that is created from scratch. Furthermore, the use of generalised templates provides flexibility whenever adaptations to these templates are required, since composing Web services by means of *concrete*[3] service interfaces leads to tightly-coupled compositions in which each service involved in the chain is tied to a Web service instance.

To equip PreDiCtS with the capability to identify more clearly which solution the user is after, we considered an extension to Case-Based Reasoning (CBR), which is called Conversational Case-Based Reasoning (CCBR), Aha [Aha et al., 2001]. The motivation being, that CBR restricts the user to define a complete problem definition (service-retrieval query) at the start of the case-retrieval process while CCBR uses a mixed-initiative approach that allows for a partial definition of the problem by the user, and makes use of a refinning, dialogue process to identify more clearly the user' s problem state. This phase is referred to as the *Similarity Phase* and is triggered when a new problem (i.e. a request for particular service functionality) is presented to the system. The CCBR approach provides a set of questions (also referred to as Question-Answer pairs) which the user can choose to answer. Depending on these answers, a ranked list of past, similar cases are retrieved and recommended to the requester, together with other unanswered questions. Through this dialogue process the requester can decide when to stop this iterative-filtering phase, and when to reuse or adapt a chosen case.

The PreDiCtS service-retrieval process continues with a search for suitable services, in a service-repository. We refer to this phase in PreDiCtS, as the *Integration phase*. During this phase, a mapping is attempted, from the features in the solution of the chosen, and possibly, adapted case (this we term as the Most-Similar Case or *MSC*), to actual services that are found in a service registry. Whenever the *Similarity phase* does not return a suitable MSC the user can create a new case from scratch. Building new cases is not a trivial task, since it requires detailed knowledge about the domain. Therefore such task is best left for knowledge engineers.

[3]Concrete here is referring to service descriptions that are bound to specific, existing services

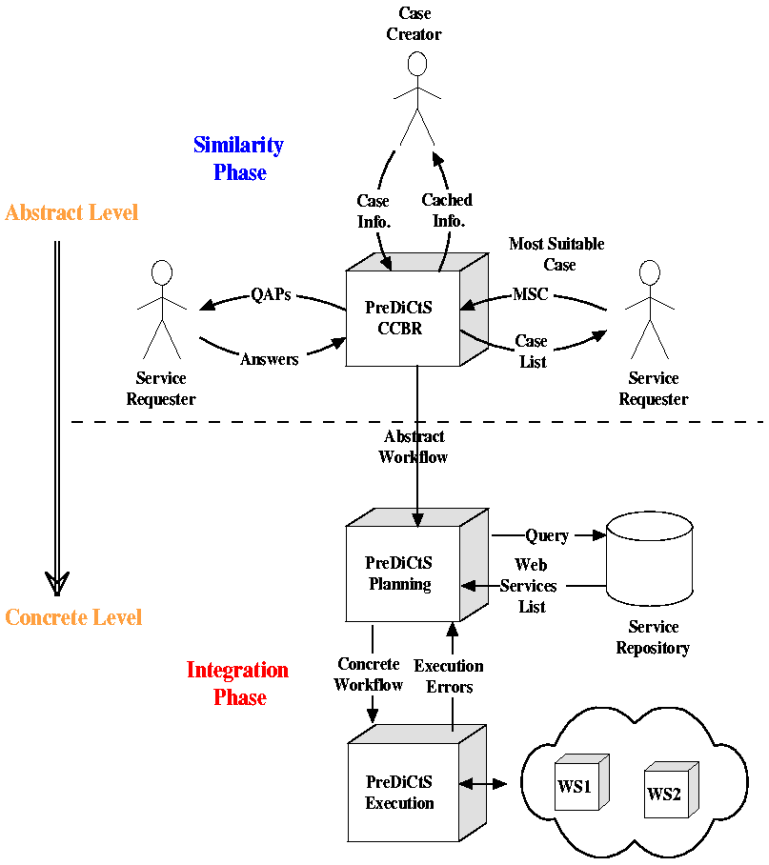


Figure 2: High-Level Architecture of PreDiCtS

We considered this situation and it is envisioned that in future versions of PreDiCtS users will be allowed to add new cases by communicating with other peers across a network.

Queries are sent to a service registry and an AI planning component is used to combine retrieved services according to this information. The planning algorithm is based on the partial order planning algorithm (POP) (defined in Weld [Weld, 1994]) and extended in Peer [Peer, 2005].

## 3.1 Integration of CCBROnto

Case-definition is based on an OWL-based ontology, which we call CCBROnto (no relation to other similar ontology presented in Gómez-Gauchía [Gómez-Gauchía et al., 2006]) and described in Abela [Abela and Montebello, 2006]. Each case is divided into three main components, namely the *CaseContext*, *Problem* and *Solution*.

The *CaseContext* represents the knowledge which is related to the case creator, case history, ranking value and case provenance. In this regard, we have considered context as defined by Dey and also ideas presented in Bry [Bry et al., 2005] and Maamar [Maamar et al., 2005] which

discuss the importance of context in relation to Web services. In PreDiCtS, context knowledge is used to identify why a case was created and by whom, certain aspects of case usage and the case relevance to problem-solving.

The *Problem* description in a PreDiCtS case is a list of question-answer pairs (QAPs) rather than a bag since these have to be ranked when they are presented to the user.

In CCBR a *Solution* represents an action that can be used to solve the requester's problem. Such actions in PreDiCtS encapsulates a process model description (or a service composition template). Each template represents a workflow of generic, unbound service components. We use the OWL-S [owls, 2004] process model as the basis for these templates in this work, though, we envision that other workflow–model description languages could be utilised as well.

**State of development**: Version 1.0 of the CCBROnto ontology has been implemented and used to create a number of test cases that were used in the initial evaluation of PreDiCtS.

**List of attached articles**

Technical details about the CCBR Ontology can be found in:

> C. Abela and M. Montebello. CCBR Ontology for Reusable Service Templates. In Proceedings of the 3rd European Semantic Web Conference (ESWC 2006), Budva, Montenegro (11th - 14th June 2006) 4011, 59-60, June 2006

## 3.2 Case Creation and Retrieval

The Case-creation process enables the case-designer to easily create a new *case* which includes the definitions of the *Context*, *Problem* and *Solution*. These definitions are based on the CCBROnto ontology mentioned earlier. By providing an easy-to-use interface and hiding the complexity of these definitions, the designer can concentrate on what he wants to achieve with the new case rather then how the various definitions will be combined together.

The context knowledge that the designer is required to add reflects his credibility and expertise. This will become very useful whenever cases are shared between PreDiCtS users. Such knowledge provides for an element of trust in the form of a *foaf:RDF* [Miller and Brickley, ] description about the designer. The system, though, is flexible enough to deal with other trust definition languages. Apart from trust in the designer himself, this context knowledge is important when building a history of case– usage. Together with user feedback, this will help in the generation of a reputation value associated with each case. It is envisioned that future case users will be provided with a means to measure the utility of a particular case to solve a problem.

Problems are represented through a set of QAPs. PreDiCtS provides for the creation, reuse and maintenance of these QAPs. Creating QAPs involves the use of a mapping tool that associates the newly defined question and answer with concepts from a chosen problem definition ontology. Answers are either assigned binary-values, that is Yes or No, or else are associated with a concept from the ontology. Reuse of QAPs is provided to allow case designers to minimise the time for the case creation process. During case-creation a set of QAPs from a chosen problem-domain are displayed and the user can decide whether a QAP is suitable for a particular case or not.

The solution in a case-description is represented through a template which includes details about a workflow of services. To generate the solution, PreDiCtS provides a workflow composer tool based on the approach adopted by Scicluna [James Scicluna, 2004] and

Buckle [Buckle et al., 2005]. This allows for a workflow to be represented through a UML Activity Diagram and then transformed into an OWL-S process definition. The present prototype handles only OWL-S-based templates, though through the use of an existing mapping tool such as, BPEL4WS2OWLS [Aslam, ], a WS-BPEL [committee, 2006] description can be successfully transformed into an OWL-S definition and used by PreDiCtS.

Retrieval in PreDiCtS provides the user with the possibility to find cases that include composition templates that fit as much as possible to his needs. This functionality is provided through a mixed-initiative process that incrementally refines the user's query and provides cases whose problem definition is similar to the query. The retrieval mechanism uses a set of similarity metrics to find the most suitable cases and also a set of respective conversation generation algorithm to present new questions for the user during each retrieval cycle.

The similarity metrics that are used are based on those described by Weber [Weber, 2003] and Gupta [Gupta, 2001]. The former provides for a simple calculation based on the direct similarities between the QAPs. On the other hand the latter is based on a more complex calculation that considers the taxonomic relations between the QAPs. We have adapted this taxonomic similarity measure so that we are now able to use the concepts and relations defined through ontologies.Nevertheless it is envisioned that other metrics such as those presented by Hefke [Hefke et al., 2006] will be used in the future. These similarity measures are ideal when comparing ontologies or parts of ontologies and are not based solely on the subsumption relation but also on other relations that consider distance and other semantical aspects.

**State of development**: Case creation and retrieval have been implemented in a prototype and initial evaluations have been carried out. However the work is still progressing and is now focused on the adaptation component and feedback mechanism. These will soon be integrated into PreDiCtS.

#### List of attached articles

Technical details about the PreDiCtS can be found in:

> C. Abela and M. Montebello. PreDiCtS: A Personalised Service Discovery and Composition Framework. In Proceedings of Semantic Web Personalization Workshop (SWP06), Budva, Montenegro (12th June 2006), 1-10, June 2006
>
> C. Abela and M. Montebello. Conversational Case-Based Reasoning Approach to Service Discovery and Composition. Technical Report, Computer Science Annual Research Workshop (CSAW), University of Malta, November 2006.

### 3.3 PreDiCtS Planning and Execution

In the *Similarity phase* we refer to planning as being the next logical step towards the actual discovery and composition of services. The idea is to take the *MSC* or its adapted copy and present it to an AI planner that will use the relevant knowledge to query a matchmaker.

The PreDiCtS' process has some advantages over other composition techniques such as those defined in Peer [Peer, 2005], Sirin [Sirin et al., 2004], Mithun [Sheshagiri, 2004], since the user' s involvement, through the CCBR process, identifies more clearly what the user is requesting. The majority of the research initiatives adopting the use of AI planning describe a process which leaves the requester out of the loop during the service-retrieval process, since it is assumed that complete information is available to the planner. On the other hand we want the requester

to be in control and to specify as clearly as possible what his needs are. Thus while certain initiatives, that are based on OWL-S, present a query to the matchmaker, which is based solely on the profile, we allow the query to be based over a generalised definition by providing a template which can be used by the planner to effectively create a successful stream of services that satisfy the requester's needs.

The implementation of this functionality requires the use of a suitable AI planning algorithm. Both the HTN (Hierarchical Task Network) and POP (Partial Order Planning) algorithms could be used in this situation as described by Sirin [Sirin et al., 2004] and Peer [Peer, 2005] respectively. We envision that a mapping component that takes as input an OWL-S description and transforms this into the planner's plan definition language is required. For this purpose we will make use of a third-party component, called OWLS2PDDL [16], that takes as input an OWL-S description and transforms this into an XML-like representation of PDDL 2.1 [McDermott, ] called PDDXML, described in Klusch [Klusch and Gerber, 2006].

PDDL is a FOL language with lisp-like syntax, and PDDXML is a dialect that uses XML syntax for PDDL expressions, which makes it easier to use. Essentially, the owls2pddl converter creates the PDDL problem description (in PDDXML syntax) by using the following conversions: (this is taken from the owls2pddl Readme file)

- *OWL-S properties -> PDDL predicates;*
- *OWL-S types -> PDDL types;*
- *OWL-S objects -> PDDL objects;*
- *OWL-S resources and datatypes -> instantiated PDDL predicates for the initial and goal state;*

Given that services have been associated to the actions, in the planning domain, it is straight forward to take the generated PDDXML file and the domain and problem definitions and present these as input to an AI planner.

Since the initial OWL-S service definition is generic and therefore incomplete, the planner has to be able to work with partial knowledge. Thus POP will be ideal for this situation and can be used to find services that will bind with the chosen template to produce a concrete OWL-S service that can eventually be executed.

**State of development**:This component is still in the design phase and development will start in due time.

# 4 Conclusion

This deliverable reports about the most recent advancements of frameworks developed within the working group A3 and, in particular, by th groups of Hannover, Malta and Turin. We have presented an executive summary that describes at a high level the novelties and the state of developments of such frameworks, and we attached in the Appendix articles that have been accepted for presentation at international conferences or for publication in international journals, in order to show the relevance of the proposals.

## Acknowledgement

This research has been co-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`).

## References

[Abela and Montebello, 2006] Abela, C. and Montebello, M. (2006). Ccbr ontology for reusable service templates. In *The Semantic Web: Research and Applications, ESWC06*, Budva, Montenegro.

[Aha et al., 2001] Aha, D. W., Breslow, L., and noz Avila, H. M. (2001). Conversational case-based reasoning. *Applied Intelligence*, 14(1):9–32.

[Aslam, ] Aslam, M. A. Bpel4ws2owls mapping tool. Available at http://bpel4ws2owls.sourceforge.net/.

[Baldoni et al., 2006] Baldoni, M., Baroglio, C., Martelli, A., Patti, V., and Torasso, L. (2006). Verifying the compliance of personalized curricula to curricula models in the semantic web. In Bouzid, M. and Henze, N., editors, *Proc. of the Semantic Web Personalization Workshop, held in conjuction with the 3rd European Semantic Web Conference*, pages 53–62, Budva, Montenegro.

[Baldoni et al., 2004a] Baldoni, M., Baroglio, C., and Patti, V. (2004a). Web-based adaptive tutoring: an approach based on logic agents and reasoning about actions. *Artificial Intelligence Review*.

[Baldoni et al., 2004b] Baldoni, M., Baroglio, C., and Patti, V. (2004b). Web-based adaptive tutoring: An approach based on logic agents and reasoning about actions. *Artificial Intelligence Review*, 1(22):3–39.

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

[Baldoni et al., 2004d] Baldoni, M., Baroglio, C., Patti, V., and Torasso, L. (2004d). Reasoning about learning object metadata for adapting scorm courseware. In Aroyo, I. L. and C. Tasso, e., editors, *Proc. of EAW'04: Methods and Technologies for personalization and Adaptation in the Semantic Web*, pages 4–13.

[Baldoni et al., 2004e] Baldoni, M., Baroglio, C., Patti, V., and Torasso, L. (2004e). Reasoning about learning object metadata for adapting SCORM courseware. In Aroyo, L. and Tasso, C., editors, *Int. Workshop on Engineering the Adaptive Web, EAW'04: Methods and Technologies for Personalization and Adaptation in the Semantic Web, Part I*, pages 4–13, Eindhoven, The Netherlands.

[Bry et al., 2005] Bry, F., Hattori, T., Hiramatsu, K., Okadome, T., Wieser, C., and Yamada, T. (2005). Context modeling in owl for smart building services. In *Proceedings of 17. Workshop über Grundlagen von Datenbanken, Wörlitz, Germany (17th–20th May 2005)*. GI.

[Buckle et al., 2005] Buckle, M., Abela, C., and Montebello, M. (2005). A bpel engine and editor for the .net framework. In *3rd IEEE European Conference on Web Services (IEEE ECOWS 2005)*, Vaxjo, Sweden.

[Clarke and Peled, 2001] Clarke, O. E. M. and Peled, D. (2001). *Model checking*. MIT Press, Cambridge, MA, USA.

[committee, 2006] committee, W.-B. (2006). Ws-bpel specification. Available at http://www.oasis-open.org/committees/download.php/18714/wsbpel-specification-draft, May-17.htm.

[Deelman, 2003] Deelman, E. (2003). Mapping abstract complex workflows onto grid environments.

[Emerson, 1990] Emerson, E. A. (1990). Temporal and model logic. In *Handbook of Theoretical Computer Science*, volume B, pages 997–1072. Elsevier.

[European Commission and Training, ] European Commission, E. and Training. The bologna process. `http://europa.eu.int/comm/education/policies/educ/bologna/bologna_en.html`.

[Gómez-Gauchía et al., 2006] Gómez-Gauchía, H., Díaz-Agudo, B., and González-Calero, P. A. (2006). Ontology-driven development of conversational cbr systems. In *ECCBR*, pages 309–324.

[Gupta, 2001] Gupta, K. M. (2001). Taxonomic conversational case-based reasoning. In *ICCBR*, pages 219–233.

[Hefke et al., 2006] Hefke, M., Zacharias, V., Abecker, A., Wang, Q., Biesalski, E., and Breiter, M. (2006). An extendable java framework for instance similarities in ontologies. In *ICEIS (2)*, pages 263–269.

[Henze and Krause, ] Henze, N. and Krause, D. Personalized access to web services in the semantic web. In *The 3rd International Semantic Web User Interaction Workshop (SWUI, collocated with ISWC 2006*.

[James Scicluna, 2004] James Scicluna, Charlie Abela, M. M. (2004). Visual modelling of owl-s services. In *IADIS International Conference WWW/Internet*, Madrid, Spain. IADIS.

[Klusch and Gerber, 2006] Klusch, M. and Gerber, A. (2006). Evaluation of service composition planning with owls-xplan. In *WI-IATW '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2006 Workshops)(WI-IATW'06)*, pages 117–120, Washington, DC, USA. IEEE Computer Society.

[LOM, 2002] LOM (2002). LOM: Draft Standard for Learning Object Metadata. http://ltsc.ieee.org/wg12/index.html.

[Maamar et al., 2005] Maamar, Z., Mostéfaoui, S. K., and Mahmoud, Q. H. (2005). Context for personalized web services. In *HICSS*.

[McDermott, ] McDermott, D. Pddl, planning domain definition language. Available at http://cs-www.cs.yale.edu/homes/dvm/.

[Miller and Brickley, ] Miller, L. and Brickley, D. Foaf specification. Available at http://www.foaf-project.org/.

[owls, 2004] owls (2004). OWL-S: Web Ontology Language for Services, W3C Submission. http://www.org/Submission/2004/07/.

[Peer, 2005] Peer, J. (2005). A pop-based replanning agent for automatic web service composition. In *ESWC*, pages 47–61.

[Rajasekaran et al., 2004] Rajasekaran, P., Miller, J. A., Verma, K., and Sheth, A. P. (2004). Enhancing web services description and discovery to facilitate composition. In *SWSWPC*, pages 55–68.

[Sheshagiri, 2004] Sheshagiri, M. (2004). Automatic composition and invocation of semantic web services. Master's thesis.

[Singh, 1998] Singh, M. P. (1998). Agent communication languages: Rethinking the principles. *IEEE Computer*, 31(12):40–47.

[Singh, 2000] Singh, M. P. (2000). A social semantics for agent communication languages. In *In Issues in Agent Communication*, number 1916 in LNCS, pages 31–45. Springer.

[Sirin et al., 2005] Sirin, E., Parsia, B., and Hendler, J. (2005). Template-based composition of semantic web services. In *Agents and the Semantic Web, AAAI Fall Symposium*.

[Sirin et al., 2004] Sirin, E., Parsia, B., Wu, D., Hendler, J., and Nau, D. (2004). Htn planning for web service composition using shop2. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 1(4):377–396.

[W3C, ] W3C. RDF Primer. Available at http://www.w3.org/TR/rdf-primer/.

[Weber, 2003] Weber, B. (2003). *Integration of Workflow Management and Case-Based Reasoning, Supporting Business Processes through an Adaptive Workflow Management System*. PhD thesis, University of Innsbruck.

[Weber et al., 2005] Weber, B., Rinderle, S., Wild, W., and Reichert, M. (2005). Ccbr-driven business process evolution. In *ICCBR*, pages 610–624.

[Weld, 1994] Weld, D. (1994). An introduction to least commitment planning. *AI Magazine*, 15:27–61.

# Appendix 1

We collect in Appendix 1 the DOAP descriptions of the described systems. DOAP is a short notation standing for "Description Of A Project"[4]. It is an XML/RDF vocabulary developed to describe open source projects. More specifically:

[4]http://usefulinc.com/doap/.

- Figure 3 reports the DOAP description for the curriculum planning service;
- Figure 4 reports the DOAP description for the validation service;
- Figure 5 reports the DOAP description for PreDiCtsS.

```
<Project xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns="http://usefulinc.com/ns/doap#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/"
         xmlns:admin="http://webns.net/mvcb/">
 <name>Personal Reader: Curriculum Planner Service</name>
 <shortname>Curriculum Planner</shortname>
 <shortdesc>A PService for creating personalized Curriculum Plans</shortdesc>
 <description>The Curriculum Planner is a Personalized Semantic Web
              Service for creating a personalized study plan based on the users
              preferences and given learning goals. It uses a simple prolog
              reasoner to create a plan from a university course database
 </description>
 <homepage rdf:resource="http://www.personal-reader.de/" />
 <wiki rdf:resource="http://wiki.kbs.uni-hannover.de/" />
 <download-page rdf:resource="http://www.personal-reader.de/download" />
 <os>Linux</os>
 <os>Unix</os>
 <programming-language>Java</programming-language>
 <programming-language>Prolog</programming-language>
 <license rdf:resource="http://usefulinc.com/doap/licenses/gpl" />
 <license rdf:resource="http://usefulinc.com/doap/licenses/mit" />
 <maintainer>
  <foaf:Person>
     <foaf:name>Ingo Brunkhorst</foaf:name>
     <foaf:homepage rdf:resource="http://www.l3s.de/~brunkhor/"/>
     <foaf:mbox_sha1sum>d99f3114b5202c4a2a8d3c24dd8d48b8f667e8c0</foaf:mbox_sha1sum>
  </foaf:Person>
 </maintainer>
 <developer>
  <foaf:Person>
     <foaf:name>Elisa Marengo</foaf:name>
     <foaf:homepage rdf:resource="http://www.di.unito.it/"/>
     <foaf:mbox_sha1sum>93c5ab22037963f3c9734675af8ba157f4d0de5e</foaf:mbox_sha1sum>
  </foaf:Person>
 </developer>
 <helper>
  <foaf:Person>
     <foaf:name>Matteo Baldoni</foaf:name>
     <foaf:homepage rdf:resource="http://www.di.unito.it/~baldoni/"/>
  </foaf:Person>
 </helper>
 <repository>
   <CVSRepository>
     <module>CurriculumPlannerService</module>
   </CVSRepository>
 </repository>
</Project>
```

Figure 3: DOAP description of the planning service.

```
<Project xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns="http://usefulinc.com/ns/doap#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/"
         xmlns:admin="http://webns.net/mvcb/">
 <name>Personal Reader: Curriculum Validation Service</name>
 <shortname>Curriculum Validator</shortname>
 <shortdesc>A PService for validation Curriculum Plans</shortdesc>
 <description>The Curriculum Validator is a Personalized Semantic Web
       Service for validation of a study plan against the formal model of a
       curriculum. It uses SPIN model checking to validate a user given plan
       against a specified formal curriculum model.
 </description>
 <homepage rdf:resource="http://www.personal-reader.de/" />
 <wiki rdf:resource="http://wiki.kbs.uni-hannover.de/" />
 <download-page rdf:resource="http://www.personal-reader.de/download" />
 <os>Linux</os>
 <os>Unix</os>
 <programming-language>Java</programming-language>
 <programming-language>Prolog</programming-language>
 <programming-language>Promela</programming-language>
 <license rdf:resource="http://usefulinc.com/doap/licenses/gpl" />
 <license rdf:resource="http://usefulinc.com/doap/licenses/mit" />
 <maintainer>
  <foaf:Person>
     <foaf:name>Ingo Brunkhorst</foaf:name>
     <foaf:homepage rdf:resource="http://www.l3s.de/~brunkhor/"/>
     <foaf:mbox_sha1sum>d99f3114b5202c4a2a8d3c24dd8d48b8f667e8c0
     </foaf:mbox_sha1sum>
  </foaf:Person>
 </maintainer>
 <developer>
  <foaf:Person>
     <foaf:name>Elisa Marengo</foaf:name>
     <foaf:homepage rdf:resource="http://www.di.unito.it/"/>
     <foaf:mbox_sha1sum>93c5ab22037963f3c9734675af8ba157f4d0de5e
     </foaf:mbox_sha1sum>
  </foaf:Person>
 </developer>
 <helper>
  <foaf:Person>
     <foaf:name>Matteo Baldoni</foaf:name>
     <foaf:homepage rdf:resource="http://www.di.unito.it/~baldoni/"/>
  </foaf:Person>
 </helper>
 <repository>
   <CVSRepository>
     <module>CurriculumValidatorService</module>
   </CVSRepository>
 </repository>
</Project>
```

Figure 4: DOAP description of the validation service.

```
<Project xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns="http://usefulinc.com/ns/doap#"
 xmlns:foaf="http://xmlns.com/foaf/0.1/"
 xmlns:admin="http://webns.net/mvcb/">
 <name>Personalised Discovery and Composition of Services</name>
 <shortname>PreDiCtS</shortname>
 <description>PreDiCtS, is a framework for the personalised retrieval
      of service templates. The retrieval process in PreDiCtS uses a
      mixed- initiative technique based on Conversational Case-Based
      Reasoning (CCBR), that provides i) for a clearer identification
      of the user's service requirements and ii) based on these
      requirements, finds suitable service templates that satisfy the
      user's goal</description>
 <homepage
    rdf:resource="http://staff.um.edu.mt/cabe2/research/projects/
                  predicts/predicts.html" />
 <download-page
    rdf:resource="http://staff.um.edu.mt/cabe2/research/projects/
                  predicts/download.html" />
 <os>Linux</os>
 <os>Windows</os>
 <programming-language>Java</programming-language>
 <license rdf:resource="http://usefulinc.com/doap/licenses/gpl" />
 <license rdf:resource="http://usefulinc.com/doap/licenses/mit" />
 <developer>
    <foaf:Person>
       <foaf:name>Charlie Abela</foaf:name>
       <foaf:mbox_sha1sum>6888726cc05d7e512a412b06052992ce85ed4788
       </foaf:mbox_sha1sum>
       <foaf:homepage rdf:resource="http://staff.um.edu.mt/cabe2"/>
    </foaf:Person>
 </developer>
 <helper>
   <foaf:Person>
      <foaf:name>Matthew Montebello</foaf:name>
      <foaf:mbox_sha1sum>7e1a50d3779f57b75b0049ab48275ed327de969d
      </foaf:mbox_sha1sum>
      <foaf:homepage
         rdf:resource="http://www.cs.um.edu.mt/staff/mmont.html"/>
  </foaf:Person>
 </helper>
</Project>
```

Figure 5: DOAP description of PreDiCtS.

# Appendix 2

This appendix contains the following articles:

- C. Abela and M. Montebello. PreDiCtS: A Personalised Service Discovery and Composition Framework. In Proceedings of Semantic Web Personalization Workshop (SWP06), Budva, Montenegro (12th June 2006), 1-10, June 2006.
- C. Abela and M. Montebello. Conversational Case-Based Reasoning Approach to Service Discovery and Composition. Technical Report, Computer Science Annual Research Workshop (CSAW), University of Malta, November 2006.
- M. Baldoni, C. Baroglio, I. Brunkhorst, N. Henze, E. Marengo, and V. Patti. *A Personalization Service for Curriculum Planning.* In E. Herder and D. Heckmann, editors, Proc. of the 14th Workshop on Adaptivity and User Modeling in Interactive Systems, ABIS 2006, Hildesheim, Germany, October 2006.
- M. Baldoni, C. Baroglio, I. Brunkhorst, E. Marengo, V. Patti, A Personalization Web Service for Curricula Planning and Validation. Poster at European Semantic Web Conference, 2007.

# PreDiCtS: A Personalised Service Discovery and Composition Framework

Charlie Abela, Matthew Montebello

Department of Computer Science and AI
University of Malta
{charlie.abela, matthew.montebello}@um.edu.mt

**Abstract.** The proliferation of Web Services is fostering the need for applications to provide more personalisation during the service discovery and composition phases. An application has to cater for different types of users and seamlessly provide suitably understandable and refined replies. In this paper, we describe the motivating details behind PreDiCtS[1], a framework for personalised service discovery and composition. The underlying concept behind PreDiCtS is that, similar service composition problems could be tackled in a similar manner by reusing past composition best practices. These have to be useful and at the same time flexible enough to allow for adaptations to new problems. For this reason we are opting to use template-based composition information. PreDiCtS's retrieval and refinement technique is based on conversational case-based reasoning (CCBR) and makes use of a core OWL ontology called CCBROnto for case representations.

**Keywords:** CCBR, Ontologies, Semantic Web, Web services

## 1. Introduction

Reusability and interoperability are at the core of the Web Services paradigm. This technology promises seamlessly interoperable and reusable Web components that facilitate rapid application development and integration. When referring to composition, this is usually interpreted as the integration of a number of services into a new workflow or process. A number of compositional techniques have been researched ranging from both, manual and semi-automatic solutions through the use of graphical authoring tools [18], [19], to automated solutions based on techniques such as AI planning [17] [20] and others.

The problem with most of the composition techniques mentioned above is three fold (i) such approaches attempt to address service composition by composing web services from scratch, ignoring reuse or adaptation of existing compositions or parts of compositions, (ii) it is assumed that the requester knows exactly what he wants and

1 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

how to obtain it and (iii) composing web services by means of *concrete* service interfaces leads to tightly-coupled compositions in which each service involved in the chain is tied to a web service instance. Using this approach for service reuse, may lead to changes in the underlying workflow which range from slight modifications of the bindings to whole re-designing of parts of the workflow description. Therefore in our opinion, services should be interpreted at an abstract level to facilitate their independent composition. [10] adds, "*abstract workflows capture a layer of process description that abstracts away from the task and behaviour of concrete workflows"*, and this allows for more generalisation and a higher level of reusability. A system can start by considering such abstractly defined workflow knowledge and work towards a concrete binding with actual services that satisfy the workflow.

To make effective reuse of such abstract workflow definitions one could consider CBR, that is amenable for storing, reusing and adapting past experience for current problems. Nevertheless CBR restricts the user to define a complete problem definition at the start of the case-retrieval process. Therefore a mixed-initiative technique such as CCBR [3] is more appropriate since it allows for a partial definition of the problem by the user, and makes use of a refinement process to identify more clearly the user's problem state.

In summary we have identified the following motivating points:

1. Reusability of compositions has the advantage of not starting from scratch whenever a new functionality is required.
2. For effective reusability a higher level of abstraction has to be considered, which generalises service concepts and is not bound to specific service instances.
3. Personalisation of compositions can be achieved by first identifying more clearly the user's needs and then allowing for reuse and adaptation of past compositions based on these needs prior to binding with actual services.

The goal of this work is to present, the motivation behind, and prototype of PreDiCtS, a framework which allows for personalisation of service discovery and composition through the reuse of past composition knowledge. One could say that we are trying to encode and store common practices of compositions which could then be retrieved, reused and adapted through a personalisation technique. The solution we propose in PreDiCtS has two phases.

For the first phase, which we call the *Similarity Phase*, we have adopted a mixed-initiative technique based on CCBR. This provides for the personalisation process. Given a new problem or service composition request, this approach allows first to retrieve a ranked list of past, similar situations which are then ranked and suggested to the requester. Through a dialogue process the requester can decide when to stop this iterative-filtering phase, and whether to reuse or adapt a chosen case. Case definition is through an OWL-based ontology which we call CCBROnto [2] and which provides for the description of context, problem and solution knowledge At present PreDiCtS allows for case creation and retrieval (adaptation is in the pipeline) and once a case (or set of cases) is retrieved, it can be presented to the next phase, which we call the *Integration Phase* where a mapping is attempted, from the features found in the chosen solution, to actual services found in a service registry. Due to space restrictions this is dealt with in a future paper.

The rest of this paper is organized as follows. In Section 2 we will give some brief background information on CCBR. Then in Section 3 we will give an overview of the OWL case ontology, CCBROnto. In Section 4 we will present the architecture of PreDiCtS and some implementation details mainly focusing on the case-creator and case-retriever components. After which we present the last section with future work and concluding remarks.

## 2. Conversational Case-Based Reasoning

Case-Based Reasoning is an artificial intelligence technique that allows for the reuse of past experience to solve new problems. The CBR process requires the user to provide a well-defined problem description from the onset of the process. But users usually cannot define their problem clearly and accurately at this stage. On the other hand, CCBR allows for the problem state to be only partially defined at the start of the retrieval process. Eventually the process allows more detail about the user's needs to be captured by presenting a set of discriminative and ranked questions automatically. Depending on the user's supplied answers, cases are filtered out and incrementally the problem state is refined. With each stage of this problem refinement process, the system presents the most relevant solutions associated to the problem. In this way the user is kept in control of the direction that this problem analysis process is taking while at the same time she is presented with solutions that could solve the initial problem. If no exact solution exists, the most suitable one is presented and the user is allowed to adapt this to fit her new requirements. Nevertheless, this adaptation process necessitates considerable domain knowledge as explained in [4], and is best left for experts.

One issue with CCBR is the number of questions that the system presents to the user at every stage of the case retrieval process. This issue was tackled by [11] which defined question-answer pairs in a taxonomy and by [1] through the use of knowledge-intensive similarity metrics. In PreDiCtS we have adapted the former method[2] since a QA pairs taxonomy is defined to be an acyclic directed graph in which nodes are related to other nodes through parent-child relations and it is assumed that a node subsumes all its descendent nodes. This is very similar to how classes in OWL are related via the *subClassOf* relation and this fits well with the underlying case structure that we use in PreDiCtS.

## 3. CCBROnto

CCBROnto is an important component of PreDiCtS since it provides for (i) case and question-answer pair definitions, and (ii) the association of domain and case-specific knowledge. In CCBROnto the topmost concept is a *Case*. Its basic components are defined by the *CaseContext*, *Problem* and *Solution* classes. In [8] context is defined as "*any information that can be used to characterize the situation of an entity. An entity*

[2] Whenever we refer to this taxonomic theory we will be referring the work done by Gupta

*is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"*. We fully agree with this definition and in the *CaseContext*, we have included knowledge related to the case creator, case history, ranking and case provenance. We have considered ideas presented in [6], [7] and [15] which discuss the importance of context in relation to Web Services and stresses on the importance of the use of context in CBR, especially when cases require adaptation. Such context knowledge makes it possible to differentiate between users and thus the system could adapt cases accordingly. For example in the travelling domain, both going to a conference and going for a holiday may require similar services, such as hotel booking and flight reservation, though the use of a conference booking service is only required in the former. Thus, based on the contexts or roles of the users (a researcher the former and a tourist the latter) the CBR system can adapt the case knowledge to present cases that satisfy the requirements of both. A researcher can adapt the case for the tourist by including a suitable conference booking service.

In PreDiCtS we consider highly important such context knowledge since it helps to identify, why a case was created and by whom, together with certain aspects of case usage and its relevance to solving a particular problem. The *CaseCreator* provides a *Role* description that the creator associates himself with, together with a *foaf:Person* instance definition that describes who this person is. The motivation behind using foaf is to eventually be able to embed some level of reputation relevant to the person who created the case. The importance of this feature will become more visible and important when cases are shared.

The *CaseContext* also provides a place holder for *CaseHistory*. The knowledge associated with this feature is important when it comes to case ranking and usage, since it allows users to identify the relevance and usefulness of a case in solving a particular problem. It is also important for the case administrator when case maintenance is performed. Cases whose history indicates negative feedback may be removed from the case base. Case *Provenance* is also used in conjunction with reputation since it indicates a URL from where the case originated. Encapsulating such information in each case will help in maintaining a reliable case base.

The *Problem* state description in a PreDiCtS case is based on the taxonomic theory. Every problem is described by a list of QA pairs rather then a bag. This is required since QA pairs have to be ranked when they are presented to the user. Each *QAPair* is associated with a *CategoryName*, a *Question* and an *Answer* (see Fig.1). Each question has a textual description and is associated with a concept from the domain ontology through the *isRelatedTo* relation. We further assume that Answers could be either binary or nominal-valued. For this reason we have created two types of answer classes, *YesNoAnswer* and *ConceptAnswer*. The former is associated with a literal represented by either a *Yes* or a *No*. While the latter, requires an association with a concept in some domain ontology, through the previously mentioned *isRelatedTo* property. The motivation behind the use of this property is related to the taxonomic theory, which requires that QA pairs are defined in a taxonomy so that during case retrieval, the number of redundant questions presented to the requester is reduced. Thus during the case creation stage, each question and answer description is associated with an ontological concept defined in the domain of discourse. This is similar to how [1] associates ontology concepts with pre-defined questions. In

PreDiCtS we want to make use of such <concept-question> association so that questions and answers are implicitly defined in a taxonomy. This association is also important when similarities between QAPairs and between cases are calculated.

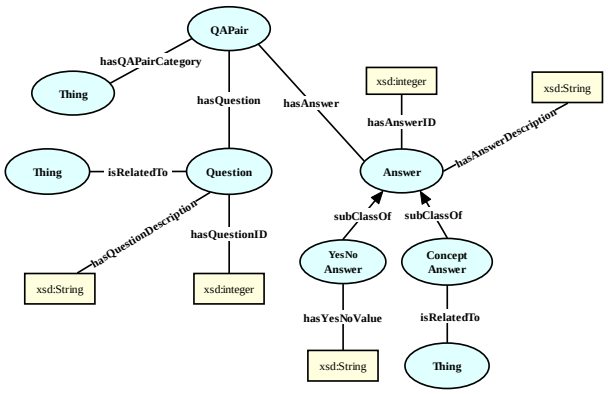


**Fig.1:** CCBROnto Problem structure

The Solution in PreDiCtS provides a hook where composition templates can be inserted. The main goal behind such a structure is to be able to present abstract composition knowledge as solutions to the user's request and at the same time allow for more flexibility when searching for actual services. In fact each *Solution* is defined to have an *Action* which has a description and is*DefinedBy* an *AbstractTemplate*. A template can be sub-classed by any service composition description, such as that defined by OWL-S. An OWL-S template in this case is an intersection between a service, profile and process definitions.

## 4. PreDiCtS: implementation issues

As explained in other sections, the PreDiCtS framework allows for the creation and retrieval of cases in its *Similarity phase* (see Fig. 2). The respective components that perform these two tasks are the *CaseCreator* and the *CaseRetrieval*. PreDiCtS is written in Java and is developed in Eclipse. It uses a MySQL database to store the cases and makes use of both Jena and the OWL-S APIs.

The *Similarity phase* is triggered by the user whenever she requires knowledge related to past compositions. In PreDiCtS the user is not expected to know exactly which type of services or service composition are required but she is required to answer a set of questions such that the system identifies more clearly what is required. Given information related to the domain, the retrieval process is initiated whereby all questions in a taxonomy relevant to that particular domain are presented to the user. Given the set of questions to choose from, the user can then decide to answer some of these questions. Depending on the answers provided, the system will try to find cases

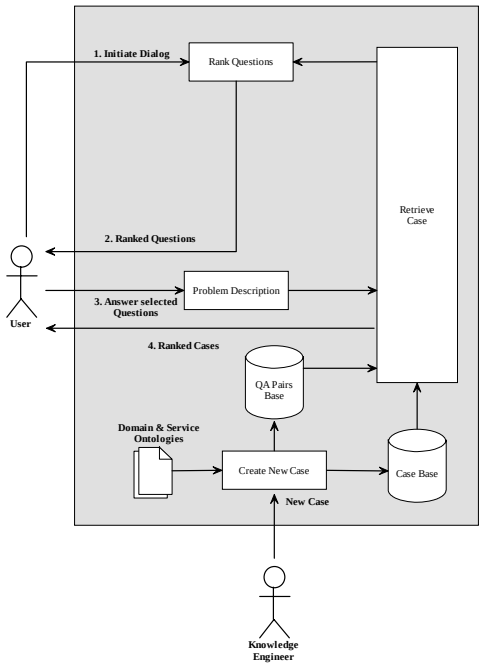


**Fig.2:** Taxonomic CCBR in PreDiCtS (adapted from Weber03)

in which questions where answered in a similar manner. A similarity measure is used to rank cases. The questions which are present in the retrieved cases but which are still unanswered, yet are related to the problem, are then presented in a ranked order to the user. The process continues until the user either chooses a case which includes a suitable solution or else, in absence of such a case, decides to adapt one of the most similar cases, thus further personalising the solution to her needs. The user can also opt to create a case from scratch to meet her requirements.

In the next sections we will describe the above mentioned PreDiCtS components by referring to an example from the health domain which deals with the combination of services that are used when a patient is admitted to hospital.

### 3.2 Case Creation

The CaseCreator component allows the expert user to add a new case to the case base. A case *c* can be defined as $c = (dsc, cxt, \{q_1a_3….q_ia_j\}, act, frq)$ where;
*dsc* is a textual description of the case.
*cxt* represents a set of context related features, such as *Role* and *CaseCreator* information based on foaf.
$\{q_1a_3….q_ia_j\}$ is a representation of the problem state by a set of question-answer pairs

*act* denotes the solution which is represented by service composition knowledge stored in an abstract template.
*frq*, is the frequency with which a case is reused.

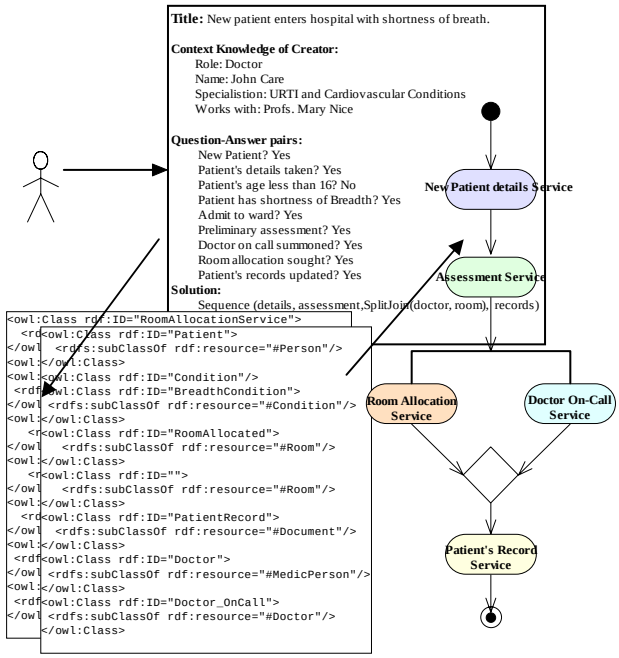


**Fig. 3:** Adding a new case

The example presented in Fig. 3 represents the combination of knowledge that is required to build a new case. PreDiCtS takes into consideration both domain and composition knowledge and combines them, based on the knowledge of the creator. In the example, the case creator is a Doctor (John) who specialises in URTI (Upper Respiratory Tract Infections) and cardiovascular conditions. The case in question represents the situation where a new patient, who is more then 16 years old, has entered hospital with shortness of breath. The creator enters context information about himself and any relations that he has with other persons. In this scenario, John has work relations with Professor Mary Nice. This information provides for a level of reputation in the expertise of the creator. The composition knowledge in this case represents a number of services that the hospital system wants to use to efficiently cater for patients entering hospital. This particular functionality is required to monitor the patient from the moment that he enters the hospital until he is comfortably stationed in a room.

To add service information to a case, the creator can use a visual component which is based on UML activity diagrams, though other representations, which are more user-friendly, are being considered. Each visual representation is mapped into a

process model representation. In this work we use OWL-S as the underlying language for this representation.

A *service* definition in OWL-S is just a place holder for information relating the profile, process and grounding. We are not considering any grounding knowledge at this stage, since this will be tackled later on in the *Integration phase* when actual service bindings are sought. As regards the profile, we only consider that knowledge which is relevant and which is not tide to specific providers. The profile part of the template includes the definitions of *inputs* and *outputs*, *profilehierarchy* and references to the process and service components. The profile hierarchy is considered to be of particular importance since it represents a reference to the service domain knowledge, that is, it identifies the taxonomic location of a particular set of service profiles. We think that such ontologies will become increasingly more important in relation to best practice knowledge. The template also provides information related to how a number of service components are combined together. What is most important here, are the control constructs such as *Sequence*, *If-Then-Else*, and *Split* that determine the order of execution of the service components. These service components are defined through the OWL-S *Perform* construct which associates a particular service component with another by binding its outputs to another service component's inputs.

An important aspect of case-creation in CCBR is the addition of question-answer pairs since they are fundamental for the case retrieval process. Through PreDiCtS we allow the creator to either reuse existing QA pairs or create new ones. Textual questions are associated with concepts defined in ontologies and this provides an implicit taxonomic structure for QA pairs. Such association provides the possibility to reason about these concepts, and also to limit the number of questions to present to the user during the retrieval process. The taxonomic theory requires that each case includes the most specific QA pair from a particular taxonomy. Given the open-world assumed by ontologies on the Web, we assume that the knowledge (triples) associated with a set of QA pairs is closed by adapting the idea of a local-closed world defined by [12].

Adding a new case to the case base is mainly the job of the knowledge expert, nevertheless we envision that even the not so expert user may be able to add cases when required. For this reason we have used the same technique as that used by recommending systems and also adopted by [21], which allows case-users to give feedback on the utility of a particular case to solve a specific problem.

### 3.3 Case Retrieval

Similarity is based on an adaptation of the taxonomic theory, and is divided into two steps, similarity between question-answer pairs and an aggregate similarity to retrieve the most suitable cases. The prior, involves the similarity between the QA pairs chosen by the user and those found in a case. In the taxonomic theory two pairs are defined to be more similar if the one found in the case is a descendant (therefore more specific) of the other, rather then its parent (therefore more generic). Though we have adopted this similarity assessment metric, we take into consideration that each QA

pair is a set of triples or rather an acyclic directed graph. Thus similarity between QA pairs is based on the similarity between two such graphs. The taxonomic similarity is calculated as follows:

$$sim\ (C_{Q1}, C_{Q2}) = \begin{cases} 1 & if\ \ C_{Q2} \subseteq C_{Q1} \\ (n+1-m)/(n+1+m) & if\ \ C_{Q1} \subseteq C_{Q2} \\ 0 & otherwise \end{cases}$$

where, $C_{Q1}$ and $C_{Q2}$ are concepts

$n$= number of edges between $C_{Q1}$ and the root i.e. the concept *Thing*

$m$= number of edges between $C_{Q1}$ and $C_{Q2}$

Having calculated such similarity between QA pairs then an aggregate similarity metric is used to calculate the overall similarity between the user query $Q_U$ and a case problem description, $P_C$. This aggregate similarity is calculated as follows:

$$sim\ (Q_U, P_C) = \frac{\sum_{i \in Q_U, j \in P_C} sim(C_{Q\,i}, C_{Q\,j})}{T}$$

where, $T$ in the original taxonomic theory represents the number of taxonomies, here it represents the number of different ontologies that are used to define the concepts found in the QA pairs.

We are also looking at other research work which provides for similar measures, in particular work related to ontology-based similarity measures [13], [16] and semantic distance [5], [14]. Such work is important since it does not only consider the taxonomic similarity between concepts but also similarity based on the number of relations and attributes associated with the concepts.

## 4. Conclusion

In this paper we presented the main concepts behind PreDiCtS. The use of CCBR as a pre-process to the service discovery and composition is promising since it provides for inherent personalisation of the service request and thus as a consequence also more personalised compositions. We also presented CCBROnto as a case definition language which allows for seamless integration between CCBR and the Semantic Web, by providing reasoning capabilities about concepts within the case definitions. Nevertheless, there is still a lot to be done, especially where it comes to case generation and evaluation. A case base can only be evaluated effectively if the number of cases is large. We are infact considering the possibility of generating cases, for experimental purposes, by extracting the required template knowledge from already available service descriptions and then adding context information and QA pairs. Other issues for future consideration include the design of the questions and the

way in which they are associated with ontology concepts, the effective evaluation of the similarity metrics used with an eye on work being done on semantic similarity and also the inclusion of an adaptation component. The latter will provide for more personalisation of the solutions presented by PreDiCtS and thus also of the services that will be presented to the user.

## References


1. A.Aamodt, M. Gu, A Knowledge-Intensive Method for Conversational CBR, Proc. ICCBR'05, Chicago, August 2005
2. C. Abela, CCBROnto, http://www.semantech.org/ontologies/CCBROnto.owl
3. D.W Aha, L.A. Breslow, H. Muñoz-Avila, Conversational case-based reasoning. Applied Intelligence, 14, 9-32. (2001).
4. M.S. Aktas, D.B. Leake. et al, A Web based CCBR Recommender System for Ontology aided Metadata Discovery, GRID'04
5. A. Bernstein, et al, Simpack: a Generic Java library for Similarity Measures in Ontologies, University of Zurich, August 2005.
6. F. Bry et al, Context Modeling in OWL for Smart Buildings, Proc. of GvB2005.
7. M. d'Aquin, et al, Decentralized Case-Based Reasoning for the Semantic Web, Proc. ISWC 2005
8. A. Dey, Understanding and Using Context, in proceeding of Personal and Ubiquitous Computing, issue on Situated Interaction and Ubiquitous Computing, Feb 2001.
9. B. Diaz-Agudo et al, On Developing a Distributed CBR Framework through Semantic Web Services, Workshop on OWL: Experiences and Directions, Galway'05
10. A. Goderis et al. Seven bottlenecks to workflow reuse and repurposing. 4th Int. Semantic Web Conference, Galway, Ireland, 6-10 Nov. 2005
11. K. Gupta, Taxonomic Conversational Case-Based Reasoning, Proceedings of the 4th International Conference on Case-Based Reasoning, 2001
12. J. Heflin, H. Muñoz-Avila, LCW-Based Agent Planning for the Semantic Web, AAAI Workshop WS-02-112002
13. M. Hefke, A Framework for the successful Introduction of KM using CBR and the Semantic Web Technologies, I-Know 2004
14. N.Henze, M. Herrlich, The Personal Reader: A Framework for Enabling Personalization Services on the Semantic Web, Proc. of ABIS 04, Berlin, Germany.
15. Z. Maamar et al, Context for Personalised Web Services, 38th Hawaii International Conference on system Science, 2005
16. A. Maedche, V. Zacharias, Clustering Ontology-based Metadata in the Semantic Web, Joint Conferences (ECML'02) and (PKDD'02), Finland, Helsinki, 2002
17. J. Peer, A POP-based Replanning Agent for Automatic Web Service Composition ESWC'05
18. J. Scicluna et al, Visual Modelling of OWL-S Services, IADIS International Conference WWW/Internet, Madrid Spain, October 2004
19. E. Sirin et al, Semi-automatic composition of web services using semantic descriptions, in ICEIS 2003, Angers, France, April 2003
20. E. Sirin et al. HTN planning for web service composition using SHOP2. Journal of Web Semantics, 1(4):377-396, 2004
21. B.Weber, S. Rinderle, W. Wild, M. Reichert, CCBR–Driven Business Process Evolution, Proc. ICCBR'05, Chicago, August 2005

# CCBR Ontology for Reusable Service Templates[1]

Charlie Abela, Matthew Montebello
Department of Computer Science and AI
University of Malta
+356 2590 7295

{charlie.abela, matthew.montebello}@um.edu.mt

## ABSTRACT
We present the motivation and design of CCBROnto, an OWL Ontology for Conversational Case-Base Reasoning (CCBR). We use this ontology to define cases that can eventually be stored, retrieved and reused by a mixed-initiative approach based on CCBR. We apply this technique for retrieving Web Service Composition templates.

## Categories and Subject Descriptors
I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods

## General Terms
Design, Implementation

## Keywords
CCBR, Ontologies, OWL, Web Services

## 1. INTRODUCTION
Web Services composition is usually interpreted as the integration of a number of services into a new workflow or process. A number of compositional techniques have been researched [9,10] that attempt to address service composition by composing web services from scratch while ignoring reuse or adaptation of existing compositions or parts of compositions. Furthermore composing web services by means of *concrete* service interfaces leads to tightly-coupled compositions in which each service involved in the chain is tied to a web service instance. This approach may lead to changes in the underlying workflow which range from slight modifications of bindings to whole redesigning of parts of the workflow description. Therefore we interpret services at an abstract level to facilitate their independent composition. Infact our approach is more similar to [8,11,12], which use pre-stored abstract workflow definitions or templates in their composition framework. Abstract workflows allow for more generalisations and a higher level of reusability [5]. The use of such templates can be thought of as a pre-processing stage towards service discovery and composition, whereby abstractly defined workflow knowledge can be concretely bound to actual services that satisfy a template. To make effective reuse of such templates we have considered CCBR [6]. This extends from CBR and allows for partial definition of the problem by using a mixed-initiative refinement process to identify more clearly the user's problem state.

## 2. RELATED WORK
In recent work relating CBR to the Semantic Web [2, 4], we find the definition of two ontologies, CaseML and CBROnto. These are both defined for CBR rather then CCBR and thus do not define concepts related to question-answer (QA) pairs, which are at the core of the CCBR process. Nonetheless we considered these when we designed and implemented our OWL-based ontology, which we call CCBROnto (this has no relation to CBROnto). We make use of this ontology within our personalised service discovery and composition framework (PreDiCtS) to define cases of best practice composition knowledge. In what follows we make explanatory references to this ongoing work.

## 3. CCBRONTO
In CCBROnto the basic components of a *Case* are defined by the *CaseContext*, *Problem* and *Solution* classes. This structure is motivated by the underlying methodology used in PreDiCtS. In this framework we adapt the CCBR approach to help the user refine his query for a particular service request. The problem description is defined by a set of discriminating QA pairs, which characterize a particular solution. On the other hand, the solution is a place holder for a reusable service composition template which is a container of best practice knowledge about composition of generic service components. In the following sections we will explain in more detail the basic *Case* components and illustrate by means of an example how such a case is defined.

### 3.1 Context
In [3], the term context is defined as "*any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"*.

We fully agree with this definition and in the *CaseContext*, see Figure 1, we have included knowledge related to the case creator, case history, and case provenance. We have also considered ideas presented in [7] and [1] which discuss the importance of context in relation to Web Services. In PreDiCtS context knowledge helps to identify, *(i)* why a case was created and by whom, *(ii)* certain aspects of case usage and *(iii)* the case relevance to problem solving. The *CaseCreator* includes a reference to the *Role* description, that the creator associates himself with, together with a *foaf:Person* instance-definition that describes who this person is. The motivation behind using

Demos and Posters of the 3rd European Semantic Web Conference (ESWC 2006), Budva, Montenegro, 11th - 14th June, 2006

1This research has partially been funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Program project REWERSE number 506779

foaf is to keep track of reputation knowledge which could be used to reliably share cases between PreDiCtS users.

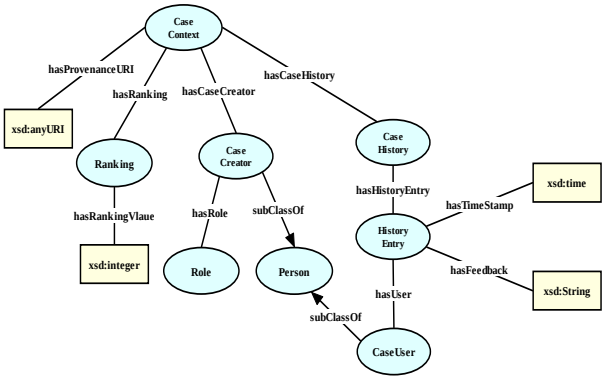


**Figure 1: CCBROnto Context structure**

The *CaseContext* also provides a place holder for *CaseHistory*, which becomes important when it comes to case ranking and usage, since it allows users to identify the relevance and usefulness of a case in solving a particular problem. It is also important for the case administrator when case maintenance is performed. Cases whose history indicates negative feedback may be removed from the case base. Case *Provenance* is also used in conjunction with reputation issues, since it associates a case with a URL indicating the case-origin.

```
<ccbr:Case rdf:ID="case1">
  <ccbr:CaseContext rdf:ID="cntxt1">
    <ccbr:hasProvenanceURI rdf:resource="http://www.....org"/>
    <ccbr:hasCaseCreator>
      <ccbr:CaseCreator rdf:ID="ccr1">
        <ccbr:hasRole rdf:resource="&role;#KnowledgeEng"/>
        <foaf:Person>
          <foaf:name>Joe Black</foaf:name>
          <foaf:mbox rdf:resource="mailto:joe@test.org"/>
          <foaf:homepage rdf:resource="http://www...../joe"/>
        </foaf:Person>
      </ccbr:CaseCreator>
    </ccbr:hasCaseCreator>
  </ccbr:CaseContext>
  <ccbr:Problem rdf:ID="prob1">
    <ccbr:QAPairList>
      <list:first>
        <ccbr:QAPair rdf:nodeID="quest1"/>
      </list:first>
      <list:rest rdf:resource="&list;#nil"/>
    </ccbr:QAPairList>
  </ccbr:Problem>
  <ccbr:Solution rdf:ID="sol2">
    <ccbr:hasAction>
      <ccbr:OWLSTemplate rdf:ID="tmpl3">
        <ccbr:hasServiceTemplate rdf:resource="#Trav_Serv"/>
        <ccbr:hasProcessTemplate rdf:resource="#Trav_Proc"/>
        <ccbr:hasProfileTemplate rdf:resource="#Trav_Prof"/>
      </ccbr:OWLSTemplate>
    </ccbr:hasAction>
  </ccbr:Solution>
</ccbr:Case>
```

**Figure 2: CCBROnto Case instance definition**

## 3.2 Problem

The *Problem* state description, see Figure 3, in a PreDiCtS case is based on the taxonomic theory of [6]. Every problem is described by a list of QA pairs rather than a bag. This is required since QA pairs have to be ranked when they are presented to the user. Each QA pair consists of a *CategoryName*, a *Question* and an *Answer*. Since the taxonomic theory requires that QA pairs are defined in a taxonomy during the case creation stage, each question description is associated, through the property *isRelatedTo*, with an ontological concept defined in the domain of discourse. This relation is not intended to fully capture the natural semantics of the QAs, rather it is important when calculating similarities.

A typical QA pair example from the traveling domain might include the question, "*What type of transportation?* This is related, by means of the *isRelatedTo* property, to the concept *Transportation*, which is defined in the Traveling domain. On the other hand, we assume that *Answers* could have either a binary or nominal value and are respectively defined in the ontology by the *YesNoAnswer* and *ConceptAnswer* classes.

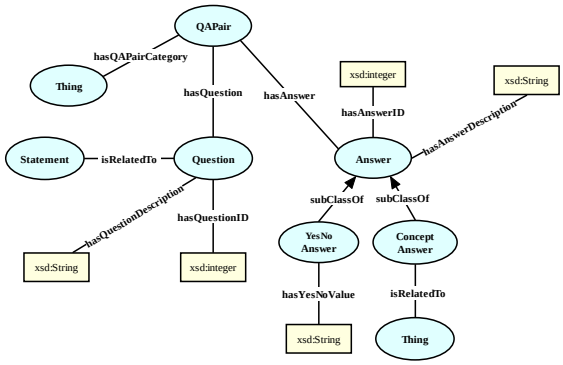


**Figure 3: Case Problem Definition**

The former points to the binary literals, while the latter is used to represent answers that are associated to a concept in a domain ontology through the previously mentioned *isRelatedTo* property.

## 3.3 Solution

The solution in PreDiCtS provides a hook where composition templates can be inserted. Each *Solution*, see Figure 4, is defined to be an *Action* which has a description and a composition template. A template can be sub-classed by a description such as that defined by OWL-S, as shown in Figure 2, though in practice it can be specialized also by other service descriptions.

# 4. CONCLUSION

Through the use of CCBROnto we are able to define cases whose solutions are composition templates. This allows our PreDiCtS framework to retrieve such templates by consulting the user in every stage and presenting her with the most suitable composition knowledge available to choose from. The user can then decide whether to reuse as is, or possibly adapt this to fit her personal needs.

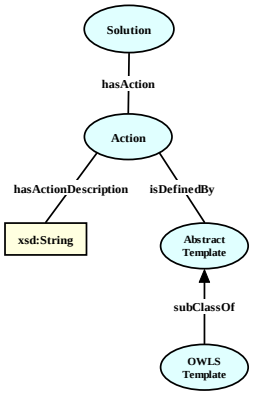


**Figure 4: Case Solution Definition**

## 5. REFERENCES


[1] Bry, F., et al, *Context Modeling in OWL for Smart Buildings*, in proceedings of GvB2005.

[2] Chen, H., et al, *CaseML: RDF based markup language for CBR on the Semantic Web*, ICCBR 2003

[3] Dey, A.K., *Understanding and Using Context*, in proceedings of Personal and Ubiquitous Computing, issue on Situated Interaction and Ubiquitous Computing, February 2001.

[4] Diaz-Agudo, B., et al, *On Developing a Distributed CBR Framework through Semantic Web Services*, OWL Experiences and Directions Workshop, Galway, 2005

[5] Goderis, A., et al, *Seven bottlenecks to workflow reuse and repurposing*, 4th Int. Semantic Web Conf., Galway, Ireland, 6-10 Nov. 2005

[6] Gupta, K., *Taxonomic Conversational Case-Based Reasoning*, 4th International Conference on CBR, 2001

[7] Maamar, Z., et al, *Context for Personalized Web Services*, 38th Hawaii International Conference on system Science, 2005

[8] Rajasekaran, P., et al, *Enhancing Web services description and discovery to facilitate composition*, First International Workshop, SWSWPC, July 2004

[9] Sirin, E., Parsia, B., et al, *Filtering and selecting semantic web services with interactive composition techniques*, IEEE Intelligent Systems, 19(4): 42-49,2004

[10] Sirin, E., et al, *Planning for web service composition using SHOP2*, Journal of Web Semantics, 1(4):377-396, 2004

[11] Sirin, E., et al, *Template-based composition of semantic web services*, AAAI Fall Symposium on Agents and the Semantic Web, Virginia, November 2005.

[12] Weber, B., et al, *CCBR-Driven Business Process Evolution*, Proc. 6th Int. Conf. on Case-Based Reasoning (ICCBR'05), Chicago, August 2005

# Verifying the compliance of personalized curricula to curricula models in the semantic web*

Matteo Baldoni, Cristina Baroglio, Alberto Martelli
Viviana Patti, and Laura Torasso

Dipartimento di Informatica — Università degli Studi di Torino
C.so Svizzera, 185 — I-10149 Torino (Italy)
{baldoni,baroglio,mrt,patti,torassol}@di.unito.it

**Abstract.** In this work we propose the introduction of a decoupling between personalized curricula and curricula models. A curricula model is formalized as a set of time constraints, while personalized curricula are formalized by means of an action theory. Given this framework, it is possible to make various interesting verification tasks automatic. In particular, we will discuss the possibility of verifying the compliance of personalized curricula to models, by using temporal reasoning. Compliance verification allows to check the soundness of a curriculum customized w.r.t. available resources and user goals against a model that expresses temporal learning dependencies at the knowledge level.

## 1 Introduction

The Semantic Web is concerned with adding a semantic layer to resources that are accessible over the internet in order to enable sophisticated forms of re-use and reasoning. In the last years standard models, languages, and tools for dealing with machine-interpretable semantic descriptions of Web resources have been developed. In this context a strong new impulse to research on personalization can be given: the introduction of machine-processable semantics makes the use of a variety of reasoning techniques for implementing personalization functionalities possible, widening the range of the forms that personalization can assume.

*Learning resources* are particular kind of resources specifically useful in an educational framework. Especially with the development of peer-2-peer and service oriented e-learning architectures, it become fundamental to explore solutions for personalizing w.r.t. the user's needs the retrieval and the composition of learning web resources. In our opinion sophisticated personalization functionalities

* 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), and it has also been supported by MIUR PRIN 2005 "Specification and verification of agent interaction protocols" national project.

should combine lesson learnt in the community of traditional educational systems (especially for what concerns the re-use of learning resources), and the new possibility of running reasoning techniques developed in the AI community over the semantically annotated learning resources.

In recent years, the *educational systems* community has focussed greater and greater attention to the problem of separating the *contents* of learning resources, from the *means* that is necessary for taking advantage of the contents. The chief goal is to enable a reuse of the learning resources, where re-use is more and more often intended as a process by which the contents of a new complex learning resource, e.g. a course, are assembled, at least partly, starting from already encoded contents, the optimal situation being a complete decoupling of the resources from the platforms used for playing them. A first significant step in this direction is represented by the birth of SCORM [1] and of Learning Design [13, 14]. The former allows to build a new course (formally, a new SCO) on top of existing SCOs or assets. The latter, is focussed on the design of processes and workflows among a group of actors that take part to the learning activities. These tools, however, suffer the lack of a machine-interpretable information about the learning resources, for enabling forms of automatic composition and of verification, possibly based on reasoning.

Standard languages for semantic annotation like RDF [16] and LOM [12] can be used for filling this lack and adding some meta-data to the resources. In particular by meta-data we can supply information on the learning resources at the *knowledge level*, e.g. knowledge about the *learning objectives* of the resource and its *prerequisites*. Given such kind of annotation, we can interpret a learning resource as an action, that can profitably be used if the learner has a given set of competences (preconditions); by using it, the learner will acquire a new set of competences (effects). As we have shown in previous work [3, 2], given an annotation of resources with preconditions and effects one can rely on a classical theory of actions and applying different reasoning techniques for offering different kind of personalization functionalities. For instance, one could use *classical planning* for performing curriculum sequencing, i.e. for selecting and sequencing a set of resources which will allow a user to achieve her/his learning goal [3]. Moreover it possible to exploit *temporal projection* for validate a student-given curriculum verifying whether all the preconditions are respected [2]. Last but not least it is possible to exploit *procedural planning* for performing curriculum sequencing at the level of university courses, in order to help a student to customize a curriculum offered by the University w.r.t his/her interest [2]. In our previous work all these tasks were accomplished by exploiting the metaphor of learning resources as actions and the representation at the knowledge level of the student learning goal and knowledge profile. We exploited a reasoning engine based of the logic language DyLOG [4], that provided a unified framework for performing both classical and procedural planning and temporal projection.

In this work we aim at taking a step further on this line of research and we focus on a kind of verification that can be profitably combined with the curriculum sequencing personalization functionalities investigated in previous

work, by leading to implement sophisticated personalization applications in a unified framework. Given a semantic annotation of the resources based on the metaphor of resources as actions, we will focus on a new kind of reasoning, which can be accounted as a *compliance* verification of personalized curricula w.r.t. a curricula model. Personalized curricula are intended as learning paths through learning resources personalized w.r.t. specific user need, e.g. they could be the result of a curriculum sequencing method that exploits the planning techniques mentioned above. Curricula models specify general rules for building such paths and can be interpreted as constraints. These constraints are to be expressed in terms of knowledge elements, and maybe also on features that characterize the resources. If such resources are courses, they should not be based, generally speaking, on specific course names. So a constraint might impose a lab course to be attended after a theory course on the same topics but not that the course C123 should follow C122.

Verifying the compliance a curriculum to a model means checking: first of all, that the resources are sequenced in such a way that their preconditions are respected, that the learning goal is achieved in the end, and that along the sequence the constraints imposed by the model are satisfied. In the following we present a preliminary proposal for a knowledge representation that suits the outlined problem domain and sketch the techniques by which the comparison of courses to constraint-based schemas can be performed.

Compliance verification can be useful in many practical cases where the need of personalizing learning resource sequencing w.r.t. to the student desire has to be *combined* with the ability to check that the result of personalization fit some abstract constraints, possibly imposed by a third party. A given University could, for instance, certify that the specific curricula that it offers for achieving a certain educational goal -that built upon the local university courses- respect some European schemes defined at the abstract level of competence. Such automatic checking of compliance combined with curriculum sequencing techniques could be used for implementing processes like cooperation in curriculum design and curricula integration which are actually the focus of the so called *Bologna Process* [8], promoted by the EU ministers responsible for higher education: "Curriculum design means drawing up of a common study path aimed at reaching the educational goals that have been jointly defined. In these schemes the partners offer specific segments which complement the overall curriculum designed". Further use cases are sketched in the conclusions.

## 2 Knowledge representation and verification

In this section we discuss about the possible formal representations of *specific curricula*, intended as sequences of learning resources (e.g. documents or entire courses) and *curricula models*, intended as specifications of general schemata for achieving a certain *educational goal*, where relationships among competencies are described.

### 2.1 Description of resources based on an action theory

Let us consider a specific curriculum as a sequence of resources, that are homogeneous in their representation. Based on work in [2, 3], we represent such resources in an action theory, taking the abstraction of resources as simple actions. We interpret a learning resource as the action of acquiring some knowledge elements (effects); it can be used only if the user owns given knowledge elements or competencies (preconditions). Thus, a resource can be described in terms of knowledge elements. For instance let us use a classical STRIPS-like notation for describing the resource called *db_for_biotech* with prerequisites *relational databases* and effects *scientific databases* as:

ACTION: db_for_biothec(),
PREREQ: relational_db, EFFECTS: scientific_db

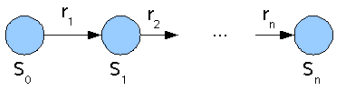


**Fig. 1.** The labels on the edges, $r_1$, $r_2$, ... $r_n$, represent learning resources. The states $S_i$ represent sets of competences that are available at a given time.

As mentioned in the introduction, the idea is to introduce a semantic annotation of learning resources that describe both their pre-requisites and effects, as done in the curriculum sequencing application in [3].

A curriculum is a sequence of resources/actions. Actions in the sequence cause transitions from a state to another, starting from an initial state up to a final state. The *initial state* represents the initial set of competences that we suppose available before the curriculum is taken (e.g. the basic knowledge that the student already has). This set can also be empty. The subsequent states are obtained by applying the actions (resources) that tag the transitions. Each of such actions has a set of preconditions that must hold in the state to which the action is applied and cause some modifications that consist in an update of the state. The prerequisites of action $r_i$ must hold in the state $S_{i-1}$. The state $S_i$ is obtained by adding to $S_{i-1}$ the effects of $r_i$. See Figure 1. We assume that competences can only be added to a state after executing the action of attending a course (or more in general reading a learning material). The intuition behind this assumption is that no new course will ever erase from the students memory the concepts acquired in previous courses, thus knowledge grows incrementally. Formally, it correspond to assume that the domain is monotonic.

### 2.2 Curricula models

Curricula models are to be defined on the basis of knowledge elements as well. In particular, we would like to restrict the set of possible sequences of resources, by imposing constraints on the order by which knowledge elements are added to the states, e.g. "a knowledge element $\alpha$ is to be acquired before a knowledge element $\beta$", "a knowledge element $\alpha$ is guaranteed to be acquired", "the acquisition of $\alpha$ implies that also $\beta$ will be acquired subsequently". Therefore we will represent a curriculum model as a set of *temporal constraints*. Being defined on knowledge elements, a curriculum model is independent from the specific resources that are taken into account, then, it can be used in an open and dynamic world like the web. A set of similar constraints defines a schema that can be used for checking user specific curricula intended as sequences of actual resources.

A natural choice for representing temporal constraints on action paths is linear-time temporal logic (LTL) [7]. This kind of logic allows the verification that a property of interest is true for all the possible executions of a model, which in our case corresponds to the specific curriculum. This is often done by means of model checking techniques [6]. The curriculum that we mean to check is, indeed, a Kripke structure; as thus, it is easy to verify properties expressed as temporal logic formulas. Briefly, a Kripke structure identifies a set of states and transition relation that allows passing from a state to another (see Figure 1). In our case, the states correspond to the competencies that are owned at a certain moment. Since we assume the domain is monotonic in the sense pointed out in the previous subsection, states will 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. Since the sequence is linear and shows no branch, then, it is possible to reason on the states and with LTL logic it is possible to verify that a given formula holds starting from a state or that it holds for a set of states.

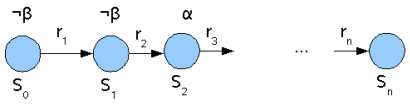


**Fig. 2.** $\beta$ can hold only after $\alpha$ becomes true, therefore, in the states that follow $S_3$ $\beta$ can either hold or not hold.

For example, the fact that a knowledge element $\beta$ cannot be acquired before the knowledge element $\alpha$ is acquired, can be written as the LTL temporal formula $\neg\beta\ U\ \alpha$, where $U$ is the *weak until* operator (see Figure 2). Given a set of knowledge elements to be acquired, such constraints specify a partial ordering of the same elements. Other kinds of constraints might be taken into account. For instance, that a knowledge element will be acquired sooner or later ($\Diamond\alpha$, eventually operator). A curriculum model is meant to allow the achievement of

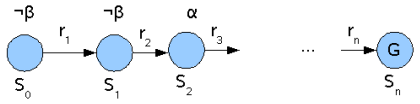


**Fig. 3.** A curriculum that allows the acquisition of the learning goal $\mathcal{G}$.

a given *learning goal*, that consists in a set of knowledge elements. We expect that the learning goal will hold in the final state of every curriculum that matches with the model (see Figure 3).

### 2.3 Compliance Verification

Given a representation of a user specific curriculum as sequence of actions/resources $(r_1, r_2, ..., r_n)$ with preconditions and effects, based on knowledge elements in an action theory $(\mathcal{A})$, and a representation of a curricula model, based on temporal constraints $(\mathcal{T})$ and a learning goal $(\mathcal{G})$, it is possible to apply different reasoning techniques for performing various interesting tasks. Besides planning, that we have already explored in previous works [2, 3], in this formal framework we can verify the *compliance* of s user specific curriculum to a model. The verification could be based on temporal reasoning techniques, like temporal projection, and on model checking techniques. Verifying the compliance means, in simple words, to check whether the curriculum respects the model, i.e that the sequence $r_1, ..., r_n$ is sound w.r.t. the precondition and effect relations specified in $\mathcal{A}$, that the sequence allows reaching the goal $\mathcal{G}$, and that the sequence respects the temporal constraints in $\mathcal{T}$. Intuitively, we can think of combining temporal projection and model checking by verifying

$$\mathcal{A} \models_{AL} \mathcal{G} \textbf{ after } r_1, ..., r_n \quad (*)$$

where $AL$ is any action logic that supports temporal projection, and

$$r_1, ..., r_n \models_{LTL} \mathcal{T} \quad (**)$$

where $LTL$ is a linear-time temporal logic.

## 3 Possible implementation

In the following we discuss the possibility of exploiting existing technology and languages for developing a system that can perform the forms of verifications described above. In particular we will deal both with the selection of languages for the representation of models and of curricula, and with the exploitation of existing tools for performing the verifications. A semantic representation of an action is quite simple and mostly consists of two lists of knowledge elements: those required for using the resource and those supplied by the resource. In

order for the knowledge elements themselves to have a semantic value, they can be implemented as terms in shared vocabulary (the simplest form of ontology). RDF can be used as an implementation language. Resources are used to define curricula. In this work we focus on curricula obtained by sequencing resources, therefore we represent a curriculum as an action sequence. This sequence can be considered as a simple kind of program that contains no branch or loop or recursive call. For what concerns action representation, there is a wide choice of action languages that are valuable candidates: we could use a logic programming action language, like $\mathcal{A}$ by Gelfond and Lifschitz [9], DyLOG [4], or GOLOG [15], all of which provide proof procedures that support temporal projection (*).

Given that a curriculum has passed the temporal projection test, we can use a model checker to verify the temporal constraints (**). 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 curriculum model and consists of a set of temporal constraints, while the finite state system is the curriculum to be verified. Among the various model checkers that have been developed, it is worthwhile to mention SPIN [11] and NuSMV [5]. SPIN, in particular, 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 the second step of our compliance test, provided that we can translate the curriculum in the internal representation used by the model checker. In the case of SPIN, the internal representation is given in the Promela language. For example, we can represent the knowledge elements as boolean variables, therefore actions as transitions that modify the values of some of these variables. The constraints will be temporal formulas that use such variables. The verification that the constraint should along the whole curriculum is performed automatically by the model checker.

In the case of linear curricula it would be easy to integrate in the temporal projection algorithm the direct verification of the constraints. The opposite solution of integrating the temporal projection into a model checker, which is the one that we mean to pursue, has the advantage of allowing the extension of the compliance test to curricula that have a more complex structure. In fact, curricula might contain tests, branching points, and repetitions. For example, if the curriculum corresponds to a learning resource that has been assembled on the basis of other learning resources (for instance a SCORM object), it might contain, as well as a program, also loops. As well as in the two-steps solution described above, it would be necessary to have a translation mechanism that allows turning the representation of the action theory into the internal formalism, used by the model checker [10].

For the sake of completeness, hereafter, we report a part of the Promela code for an example. The code allows the execution of both temporal projection and model checking. Temporal projection is handled as a deadlock verification: if the sequence is correct w.r.t. the action theory, no deadlock arises, otherwise a deadlock will be detected. The complete example and an explanation of it are available at `http://www.di.unito.it/~alice/ccompliance/`. This cur-

riculum passes the compliance test under the temporal constraints $\neg f7\ U\ f5$ and $\neg f8\ U\ f5$. In the web site it is possible to retrieve also examples of curricula which fail the test.

```
mtype = { course1, course2, course3, course4, course5 };
mtype = { done, stop, success, fail }
chan attend = [0] of { mtype };
chan feedback = [0] of { mtype };
bool f1, f2, f3, f4, f5, f6, f7, f8;

init {    f1 = true; f2 = false; f3 = false; f4 = false;
          f5 = false; f6 = false; f7 = false; f8 = false;
          run TestCompliance();
          run UpdateState();  }

inline Curriculum4() {
    attend!course1; feedback?done;
    attend!course2; feedback?done;
    attend!course5; feedback?done;
}

proctype TestCompliance() {
    Curriculum4()
    feedback!stop; feedback?success;
}

proctype UpdateState() {
    do
    :: attend?course1 -> if
       :: (f1) -> f2 = true; f3 = true; f4 = true; feedback!done;
       fi;
    :: attend?course2 -> if
       :: (f3) -> f4 = true; f5 = true; feedback!done;
       fi;
    :: attend?course3 -> if
       :: (f2 && f6) -> f7 = true; f8 = true; feedback!done;
       fi;
    :: attend?course4 -> if
       :: (f2 && f5) -> f7 = true; feedback!done;
       fi;
    :: attend?course5 -> if
       :: (f2 && f4) -> f7 = true; f8 = true; feedback!done;
       fi;
    :: feedback?stop -> if
       :: (f4 && f5 && f8) -> feedback!success;
       :: else -> feedback!fail;
       fi;
       break;
    od  }
```

The above program is hand-coded but, as the modularity of the example witnesses, it would be easy to produce an automatic translator able to turn the description of sets of courses and the description of sequences of resources into Promela code. Such code could, then, be validated according to curricula models encoded as sets of temporal constraints.

## 4 Conclusions

In this work we have presented a two-level representation of curricula, aimed at capturing the distinction between curricula and models of curricula that define general rules or constraints to be satisfied. We have shown that by implementing curricula models as temporal constraints, and curricula as sequences of actions, it is possible to verify the compliance of a curriculum to a model by exploiting reasoning techniques that combine temporal projection and model checking.

The possibility of verifying the compliance of curricula to models is extremely important in many applicative contexts where the need of personalizing learning resource sequencing w.r.t. to the student desire has to be combined with the ability to check that the result of personalization fit some abstract models. In this sense we can say that the compliance verification we propose is complementary w.r.t the capability of applying planning techniques for building from a set of available resources, personalized curricula aimed at reaching a given learning goal. Representing models as sets of constraints gives great freedom in the definition of specific curricula because it cuts away the undesired curricula without imposing unnecessary constraints. The same freedom is not supplied if we represent, as in [2], models as procedures. Procedures have a prescriptive nature that over-rules the possible solutions; the greater flexibility introduced by the use of temporal constraints has a positive effect on the possible personalization of the solutions, by allowing a greater autonomy in selecting among alternatives.

Concerning use cases, we have already mentioned the Bologna process. Another practical application could be helping a teacher that must teach a same topic to different classes, with background and purposes that vary. For instance, to teach Java to a University class as well as to professionals that work in an information technology enterprise. The teacher might be interested in the fact that all students of both classes acquire a same set of competences, with known time constraints, however, since the target students are so different it is useful to prepare two different courses exploiting different learning resources. The University students must, in fact, be taught also the theoretical background concerning object-oriented programming. On the other hand, the professionals will surely be more interested in more practical lessons, containing many real-world examples of application. The teacher might select public-domain (semantically annotated) learning resources from on-line repositories and use them to compose two different curricula personalized w.r.t. the different student targets. Nevertheless, by applying the approach that we have proposed he would have the possibility of verifying that the built curricula respect an abstract curriculum schema, derived from the expertise and the experience of the teacher himself.

We are working at the actual development of a system on the line of the sketch described in the previous section. Moreover, we are thinking to an extension (both from a formal and an implementation perspective), in which hierarchies of knowledge elements are used instead of plain vocabularies. Hierarchies allow a representation of knowledge elements at different levels of abstraction, thus they would allow other forms of verification. In order to include them, it might be necessary to integrate forms of ontological reasoning in the framework.

## Acknowledgements

The authors would like to thank prof. Nicola Henze for the precious discussions.

## References

1. ADLnet. Scorm specification. Available at http://www.adlnet.org.
2. M. Baldoni, C. Baroglio, and V. Patti. Web-based adaptive tutoring: an approach based on logic agents and reasoning about actions. *Artificial Intelligence Review*, 2004.
3. M. Baldoni, C. Baroglio, V. Patti, and L. Torasso. Reasoning about learning object metadata for adapting scorm courseware. In In L. Aroyo and editors C. Tasso, editors, *Proc. of EAW'04: Methods and Technologies for personalization and Adaptation in the Semantic Web*, pages 4–13, 2004.
4. M. Baldoni, L. Giordano, A. Martelli, and V. Patti. Programming rational agents in a modal action logic. *Annals of Mathematics and Artificial Intelligence*, 2004.
5. Alessandro Cimatti, Edmund M. Clarke, Fausto Giunchiglia, and Marco Roveri. NUSMV: A new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.
6. O. E. M. Clarke and D. Peled. *Model checking*. MIT Press, Cambridge, MA, USA, 2001.
7. E. A. Emerson. Temporal and model logic. In *Handbook of Theoretical Computer Science*, volume B, pages 997–1072. Elsevier, 1990.
8. Education European Commission and Training. The bologna process. http://europa.eu.int/comm/education/policies/educ/bologna/bologna_en.html.
9. M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–321, 1993.
10. Fausto Giunchiglia and Paolo Traverso. Planning as model checking. In *ECP*, pages 1–20, 1999.
11. Gerard J. Holzmann. The model checker SPIN. *Software Engineering*, 23(5):279–295, 1997.
12. Learning technology standards committee. draft standard for learning object metadata, 2002.
13. IMSGlobal. Learning design specifications. Available at http://www.imsglobal.org/learningdesign/.
14. R. Koper and C. Tattersall. *Learning Design: A Handbook on Modelling and Delivering Networked Education and Training*. Springer Verlag, 2005.
15. H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains. *J. of Logic Programming*, 31:59–83, 1997.
16. W3C. RDF Primer. Available at http://www.w3.org/TR/rdf-primer/.

# A Personalization Web Service for Curricula Planning and Validation

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

[1] Department of Computer Science
University of Torino
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, given at a knowledge level (what the course teaches, what is requested to know for attending it in a profitable way). The system supports users in building personalized curricula, formalized by means of an action theory. It is also possible to verify the soundness of curricula w.r.t. a model, expressing constraints at a knowledge level. For what concerns the first task, classical planning techniques are adopted, which take into account both the student's initial knowledge and her learning goal. Instead, curricula validation is done against a model, formalized as a set of time constraints. The planning service is supplied as a new plug-and-play personalization service in the Personal Reader framework. We have developed a prototype of the validation system by using the well-known SPIN model checker.

## 1 Introduction

The Semantic Web is concerned with adding a semantic layer to resources that are accessible over the internet in order to enable sophisticated forms of re-use and reasoning. In the last years standard models, languages, and tools for dealing with machine-interpretable semantic descriptions of Web resources have been developed. In this context a strong new impulse to research on personalization can be given: the introduction of machine-processable semantics makes the use of a variety of reasoning techniques for implementing personalization functionalities possible, widening the range of the forms that personalization can assume.

*Learning resources* are a particular kind of resource, used in educational frameworks. Especially with the development of peer-to-peer and service-oriented e-learning architectures, it became fundamental to explore solutions for personalizing, w.r.t. the user's needs, the retrieval and the composition of learning web resources. In our opinion sophisticated personalization functionalities should combine lessons learnt in the community of traditional educational systems (especially for what concerns the re-use

of learning resources), and the new possibility of running reasoning techniques, developed in the AI community, over the semantically annotated learning resources. In recent years, the *educational systems* community has focussed greater and greater attention on the problem of separating the *contents* of learning resources, from the *means* that is necessary for taking advantage of the contents. Standard languages for semantic annotation like RDF[3] and LOM[4] are used to facilitate the re-use of complex learning resources and allowing the building of new courses on top of existing assets. In particular by metadata we can supply information on the learning resources at the *knowledge level*, e.g. knowledge about the *learning objectives* of the resource and its *prerequisites*. Given such kind of annotation, we can interpret a learning resource as an action, that can profitably be used if the learner has a given set of competences (preconditions); by using it, the learner will acquire a new set of competences (effects). As we have shown in previous work [1–3], given an annotation of resources with preconditions and effects one can rely on a classical theory of actions and applying different reasoning techniques for offering different kind of personalization functionalities. In the following we will use *classical planning* for performing curriculum sequencing, i.e. for selecting and sequencing a set of resources which will allow a user to achieve her/his learning goal.

Another reasoning technique which can profitably be combined with the curriculum sequencing is the verification of the *compliance* of personalized curricula w.r.t. a curricula model. Personalized curricula are intended as learning paths through learning resources personalized w.r.t. specific user need, e.g. they could either be the result of a curriculum sequencing method that exploits the planning techniques mentioned above, or they could be written by hand by a user, based on own criteria. Curricula models specify general rules for building such paths and can be interpreted as constraints. These constraints are to be expressed in terms of knowledge elements, and maybe also on features that characterize the resources. Compliance verification is useful in many practical cases where the need of personalizing learning resource sequencing w.r.t. to the student desire has to be *combined* with the ability to check that the result of personalization fit some abstract constraints, possibly imposed by a third party. A given University could, for instance, certify that the specific curricula that it offers for achieving a certain educational goal —built upon the local university courses— respect some European schemes defined as constraints defined at an abstract level, as relations among of knowledge elements and competencies. The automatic checking of compliance combined with curriculum sequencing techniques could be used for implementing processes like cooperation among institutes in curricula design and integration, which are actually the focus of the so called *Bologna Process* [4], promoted by the EU ministers responsible for higher education[5].

While SCORM [5] and Learning Design [6, 7] represent the most important steps in the direction of managing and using e-learning based courses and workflows among a group of actors participating in learning activities, most of the available tools lack

[3] W3C Resource Description Framework `http://www.w3.org/TR/rdf-primer/`

[4] Learning Object Metadata `http://ltsc.ieee.org/wg12/`

[5] "Curriculum design means drawing up of a common study path aimed at reaching the educational goals that have been jointly defined. In these schemes the partners offer specific segments which complement the overall curriculum designed" [4].

the machine-interpretable information about the learning resources, and as a result are not yet open for reasoning methods for personalization and automatic composition and verification.

In our approach, using the Personal Reader (PR) framework, we use a service oriented architecture to allow personalization in a *plug-and-play* way, via the use of so called Personalization Services, which are the basic building blocks for allowing users a personalized view on Web contents. Each service is offering a personalization functionality, e.g. recommendations tailored to the needs of specific users, pointers to *related / interesting / more detailed / more general* information, and so on. These semantic web services communicate solely based on RDF documents. The two services we developed are, first the "Curriculum Planning Service", which uses a subset of the course database of the University of Hannover as the knowledge base and a Prolog reasoner to create curriculum sequences. Second, we present the "Curriculum Validation Service", which uses the SPIN reasoner to validate a given curriculum sequence to a formalized curriculum model.

Section 2 describes our approach to reasoning about curricula, details of the implementation of the new services are given in section 3: The *Curriculum Planning Service*, for building personalized paths in a space of semantic learning resources —university courses, in our particular application domain—, and the *Validation Service* that allows checking whether a curriculum satisfies the constraints given by a curricula model. We finish with conclusions and further work in Section 4

## 2 Curricula representation and reasoning

### 2.1 Description of resources based on action theory

In this work we interpret a *curriculum* as a sequence of learning resources that are homogeneous in their representation. Based on work in [2, 1], we represent such resources in an *action theory*, taking the abstraction of resources as *simple actions*, which are described only in terms of knowledge elements. A learning resource is modelled as an action for acquiring some knowledge elements (*effects*); such an action can be applied only if the executor owns given knowledge elements or competencies (*preconditions*). As an example of "learning resource as action", let us describe the resource *db_for_biotech* having as prerequisites the fact of having knowledge about *relational databases* and as effects the fact of supplying knowledge about *scientific databases* (we adopt a classical STRIPS-like notation):

ACTION: db_for_biothec(),
PREREQ: relational_db, EFFECTS: scientific_db

Both preconditions and effects can be expressed by means of a *semantic annotation* of the learning resource [1].

Given the above interpretation of learning resources, a curriculum can be interpreted as a sequence of simple actions, whose execution causes transitions from a state to another, until some final state is reached. The *initial state* represents the initial set of competences that we suppose available before the curriculum is taken (e.g. the basic

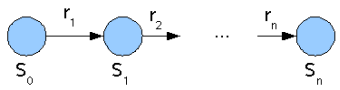


**Fig. 1.** The labels on the edges, $r_1$, $r_2$, ... $r_n$, represent learning resources. The states $S_i$ represent sets of competences that are available at a given time.

knowledge that the student already has). This set can also be empty. With reference to Figure 1, the subsequent states are obtained by applying the actions one after the other. Of course, for an action to be applicable, its preconditions must hold in the state to which it should be applied (the prerequisites of action $r_i$ must hold in the state $S_{i-1}$). The application of the action will consist in an update of the state. We assume that competences can only be added to a state after executing the action of using a learning resource. The intuition behind this assumption is that the act of using a new resource will never erase from the students' memory the concepts acquired previously, thus knowledge grows incrementally. Formally, we assume that the domain is monotonic.

In the following we will often refer to learning resources as "courses" due to the particular application domain that we have considered, i.e. University curricula.

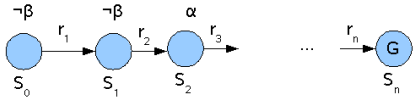


**Fig. 2.** A curriculum that allows the acquisition of the learning goal $\mathcal{G}$.

In general, curricula are supposed to allow the achievement of a given *learning goal*. A learning goal is a set of knowledge elements of interest, those that a student, following a curriculum, would like to acquire. A curriculum allows the acquisition of a set of knowledge elements if such elements are contained in the final state reached by it (see Figure 2). The learning goal is to be taken into account in a variety of tasks. The construction of a personalized curriculum is, actually, the construction of a curriculum which allows the achievement of the goal expressed by the user. In Section 3 we will describe a *curricula planning service* for accomplishing this task.

### 2.2 Curricula models

Curricula models consist in sets of constraints that specify desired properties of curricula. Curricula models are to be defined on the basis of knowledge elements as well as learning resources (courses). In particular, 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 $\alpha$ is to

be acquired before a knowledge element $\beta$", or specifying some *educational objective* to be achieved, in terms of knowledge that must be contained in the final state, e.g. "a knowledge element $\alpha$ will be surely acquired soon or later". Therefore, we will represent a curricula model as a set of *temporal constraints*. Being defined on knowledge elements, a curricula model is *independent* from the specific resources that are taken into account, for this reason, it can be reused in different contexts and it is suitable to open and dynamic environments like the web.

A natural choice for representing temporal constraints on action paths is linear-time temporal logic (LTL) [8]. 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 [9].

The curricula as we represent them are, actually, Kripke structures. Briefly, a Kripke structure identifies a set of states with a transition relation that allows passing from a state to another (see Figure 1). In our case, the states contain the knowledge items that are owned at a certain moment. Since the domain is monotonic (as explained above we can assume that knowledge can only grow), 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. It is possible to use the LTL logic to verify if a given formula holds starting from a state or if it holds for a set of states.

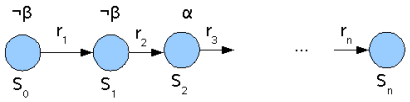


**Fig. 3.** $\beta$ can hold only after $\alpha$ becomes true, therefore, in the states preceding $S_3$ $\beta$ cannot hold; afterwards it can either hold or not hold.

For example, in order to specify in the curricula model constraints on *what* to achieve, we can use the formula $\Diamond\alpha$, where $\Diamond$ is the eventually operator. Intuitively such a formula express that a set of knowledge elements will be acquired sooner or later. Moreover, constraints concerning *how* to achieve the educational objectives, such as "a knowledge element $\beta$ cannot be acquired before the knowledge element $\alpha$ is acquired", can be expressed by the LTL temporal formula $\neg\beta\ U\ \alpha$, where $U$ is the *weak until* operator (see Figure 3). Given a set of knowledge elements to be acquired, such constraints specify a partial ordering of the same elements.

### 2.3 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, 10].

For what concerns the *composition* of personalized curricula, intuitively the idea is that, given a repository of learning resources, which have been semantically annotated

as described, 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 particular planning methodology that we implemented (see 3.2) is a simple depth-first forward planning, where actions cannot be applied more than once. Starting from the initial state, the set of applicable actions (i.e. those whose preconditions are contained in the state) is identified and one is selected and its application is simulated leading to a new state. The new state is obtained by adding to the previous one the knowledge elements that define the effects of the selected action. The procedure is repeated until either the goal is reached or a state is reached, in which no action can be applied and the goal condition does not hold. In the latter situation, backtracking is applied to look for another solution. The procedure will eventually end because the set of possible actions is finite and each is applied at most once. If the goal is achieved, the sequence of actions that label the transitions leading from the initial to the final state is returned as the resulting plan. The backtracking mechanism allows to collect a set of alternative solutions.

Personalized curricula can be composed automatically by a service as described above. However, it can be also interesting to consider the case when they are built manually by a user, reflecting his/her personal interests and needs. Of course, not all sequences which can be built starting from a set of learning resources, that are provided by a particular institute, are lawful. Constraints, imposed by courses themselves, must be respected. In other words, a course can appear at a certain point in a sequence only if it is applicable at that point. The reason is that we mean students to understand the topics taught in the various courses, and to this aim precedences and dependencies that are innate to the nature of the taught concepts are to be considered. Moreover, the student can be interested in checking if the curriculum actually allows him/her to achieve the learning goal.

In such scenario a validation service for checking the soundness of the plan w.r.t. to learning dependencies and goals is necessary. Given the interpretation of resources as actions, on the one hand the verification to accomplish is an “executability” check of the curriculum: given an an initial state, the actions are hypothetically executed one after the other until a final state is reached. A course can be executed only if, the current state contains all the concepts that are in course precondition. Intuitively, it will be applied only if the student owns the notions that are required for understanding the topics of the course. On the other hand, given that all the courses in the sequence can be applied, one after the other, the final state that is reached must be compared with the learning goal of the student: all the goals, in terms of concepts acquired, must be achieved, so the corresponding knowledge elements must be contained in the final state. Such verification task can be accomplished by the validation service described in Section 3.3.

Beside validation of user given curricula w.r.t learning dependencies and goals, the validation service allows to perform another interesting verification task: checking if a personalized curriculum is valid w.r.t. a particular *curricula model*. Indeed a personalized curriculum that is proved to be executable, cannot, however, be automatically

considered as being *valid* w.r.t. a particular *curricula model*. A curricula model, in fact, imposes constraints on *what* to achieve and *how* achieving it. The possibility of verifying the compliance of curricula to models is extremely important in many applicative contexts where the need of personalizing learning resource sequencing w.r.t. to the student desire has to be combined with the ability to check that the result of personalization fit some abstract models.

In some cases these checks could be performed during the construction process but it is important to be able to perform the verification independently from the construction of curricula. To understand why, let us consider a simple scenario concerning a University which has already verified that its curricula satisfy the guidelines given by the EU for a certain year. After some years the guidelines change: the University will need to check if its curricula, which already exist, satisfy the new guidelines.

## 3 Implementation in the Personal Reader Framework

The Personal Reader Framework have been developed with the aim of offering to the users a uniform entry point for accessing the Semantic Web, and in particular Semantic Web Services. Indeed it offers an environment for designing, implementing and realizing Web content readers in a service-oriented approach (see Figure 4). For a more detailed description, see [11].

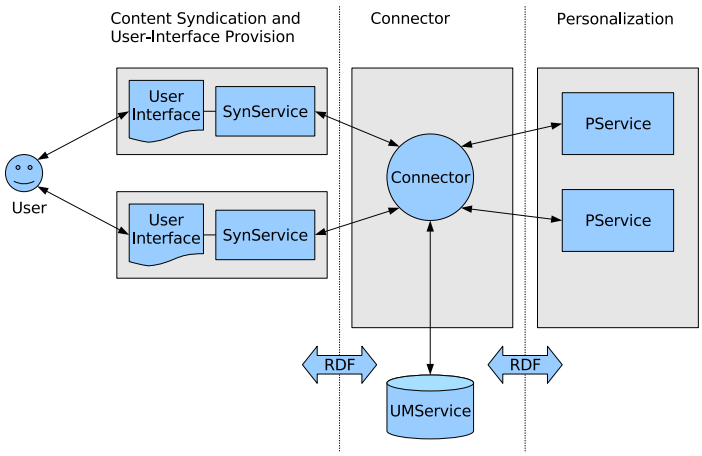


**Fig. 4.** Personal Reader Framework Overview

*Personalization Services* (PServices) deliver personalized recommendations for content, obtained or extracted from the Semantic Web. *Syndication Services* (SynServices) are used to create the appropriate user interfaces for displaying the results provided by the different PServices. The *Connector* is used as a central component required for

organizing the communication among the involved parties. It supports the discovery, selection, customization and invocation of services and, additionally, communicates with a specialized service for user modelling, *UMService*, for creating and maintaining a central user model.

### 3.1 Metadata Description of Courses

In order to create the corpus of courses for the system, we started with information collected from an existing database of courses. We used the Lixto [12] tool to extract the needed data from the web-pages provided by the HIS-LSF[6] system of the University of Hannover. This approach was chosen based on our experience with Lixto in the *Personal Publication Reader* [13] project, where we used Lixto for creating the publications database by crawling the publication pages of the project partners. The effort to adapt our existing tool for the new data source was only small. From the extracted metadata we created an RDF document, containing course names, course catalog identifier, semester, number of credit points, effects and preconditions, and the type of course, e.g. laboratory, seminar or regular course with examinations in the end, as illustrated in Figure 5.

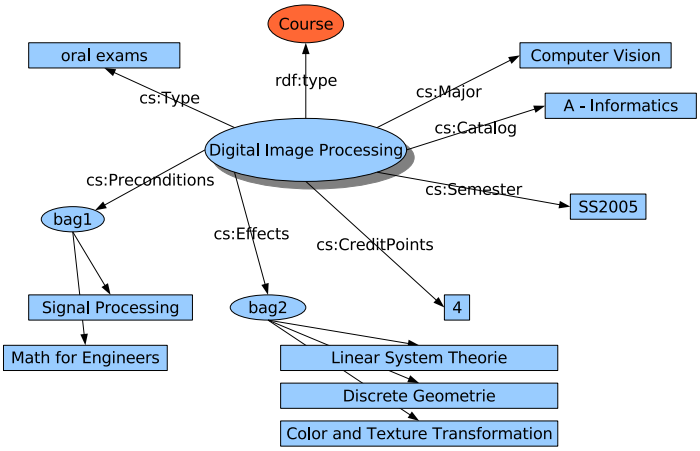


**Fig. 5.** An annotated course from the Hannover course database

The larger problem was that the quality of most of the information in the database turned out to be insufficient, mostly due to inconsistencies in the description of prerequisites and effects of the courses. Additionally the corpus was not annotated using

[6] HIS is a popular infrastructure for managing higher education in Germany, `http://www.his.de/`

a common set of terms, but authors and department secretaries used a slightly varying vocabulary for each of their course descriptions, instead of relying on a common classification system, like e.g. the ACM CCS[7] for computer science.

As a consequence, we focussed only on a subset of the courses (computer science and engineering courses), and manually post-processed the data. Courses are annotated with prerequisites and effects, that can be seen as knowledge concepts or competences, i.e. ontology terms. After automatic extraction of effects and preconditions, the collected terms were translated into proper English language, 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.

### 3.2 Embedding the Prolog Reasoner in a Personalization Web-Service

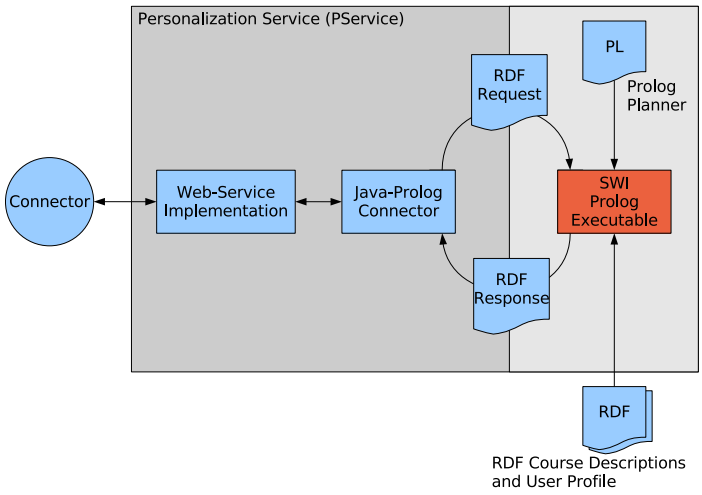


**Fig. 6.** Curriculum Planning Web Service

In order to integrate the Planning Service as a plug-and-play personalization service in the Personal Reader architecture we worked at embedding the Prolog reasoner into a web service. Figure 6 gives an overview over the components in the current implementation. The web service implements the Personalization Service (*PService* [11]) 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

[7] `http://www.acm.org/class/1998/`

RDF document containing the request *as-is* to the Prolog system, and collects the results, already represented as RDF. The service itself can be accessed by Syndication Services using discovery and invocation via the central Connector component. As a proof-of-concept, we developed a SynService that provides a simple user interface for the selection of learning goals and for displaying the generated plans.

The curriculum planning task itself is accomplished by a reasoning engine, which has been implemented in SWI Prolog[8]. 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, 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.

Given a request, the reasoner runs the Prolog planning engine on the database of courses annotated with prerequisites and effects. The initial state is set by using information about the user's context, which is maintained by the User Modelling component of the PR. In fact such user's context includes information about what is considered as already learnt by the student (attended courses, learnt concepts) and such information is included in the request document. The Prolog planning engine has been implemented by using a classical depth-first search algorithm [14]. This algorithm is extremely simple to implement in declarative languages as Prolog.

At the end of the process, a *RDF response document* is returned as an output. It contains a list of plans (sequences of courses) that fulfill the users learning goals and profile. The maximum number of possible solutions to compute can be set by the user in the request document. Notice that further information stored in the user profile maintained by the PR could be used at this stage for sorting the list of plans with the aim of adapting the presentation of the solutions.

### 3.3 The validation service

The validation service has been implemented based upon the model checker SPIN [15]. 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 validation tests that we mentioned in the previous sections, provided that we can translate the curriculum in the internal representation used by the model checker. In the case of SPIN, the internal representation is given in the Promela language.

The validation tool that we have developed can handle curricula that contain branching points. This kind of curricula do not correspond to a simple sequence of learning resources but to a more complex structure, accounting also for uncertainties of the user.

[8] `http://www.swi-prolog.org/`

A branching point corresponds to a possible choice among alternative resources. Therefore, a curriculum with branches corresponds to a set of curricula, according to the interpretation given in the first part of the paper. Notice that a curriculum with branches can contain some paths that are valid w.r.t. the curricula model and some paths which are not.

In order to verify if a curriculum with branches is valid w.r.t. a curricula model, we will use temporal logic formulas and *model checking* techniques. In particular, we have chosen SPIN, a model checker used to validate systems which can be represented as *finite state automata* and that allows for the verification of LTL formulas. Such formulas are used to describe the system requisites, and in our case they are used to define the peculiarities of a *curriculum*.

To check a curriculum with SPIN, this must be translated in the Promela language. In this implementation, the *concepts* are represented as *boolean variables*. Initially, only those variables that represent the initial knowledge of the student are true. *Courses* are implemented as actions that can modify the value of the variables. Since our application domain is monotonic (knowledge can only grow) only those variables, whose value is false in the initial state, can be modified. The program is made of two processes: one is named *Curriculum* (see Figure 7) and the other *UpdateState* (see Figure 8). While the former contains a representation of the curriculum itself, the latter contains the code for updating the state step by step along the simulation of curriculum execution. The two processes communicate by means of two channels, *attend* and *feedback*. The notation *attend!courseName* in Figure 7 simulates the fact that the course with name *courseName* is to be attended. In this case the sender process is *Curriculum* and the receiver is *UpdateState*. *UpdateState* will check the preconditions of the course in the current state and will send a feedback to *Curriculum* after updating the state. On the other hand, the notation *feedback?feedbackMsg* represents the possibility for the process *Curriculum* of receiving a feedback of kind *feedbackMsg* from the process *UpdateState*.

Given these two processes it is possible to perform the temporal projection test, aimed at verifying the correctness of the curriculum only w.r.t. the precondition/effect check (executability check). Temporal projection is handled as a deadlock verification: if the sequence is correct w.r.t. the action theory, no deadlock arises, otherwise a deadlock will be detected. The *curricula model* is to be supplied apart, as a set of temporal logic formulas.

Notice that the curriculum reported in Figure 7 contains branching points. The branching points are encoded by the non-deterministic *if*; each such *if* statement refers to a set of alternative courses (e.g. *langugesEnvironmentProg* and *programmingLanguages*). Figure 8 represents the process that updates the current state. Depending on the course communicated by the channel *attend*, it updates the state. The process continues until the message *stop* is communicated. Then the learning goal is checked.

Let us see how to use the model checker to verify the *temporal constraints* that make a curricula model. 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 curriculum model and consists of a set of temporal constraints, while the finite state system is the curriculum to be verified.

```
inline Curriculum() {
attend!architecture; feedback?done;
attend!lab_languages; feedback?done;
attend!programmingII; feedback?done;
if
:: true -> attend!algorithms; feedback?done;
:: true -> skip;
fi;
attend!db; feedback?done;
attend!economy; feedback?done;
attend!operative_systems; feedback?done;
attend!int_sys; feedback?done;
attend!lab_sweb; feedback?done;
if
:: true -> attend!langugesEnvironmentProg; feedback?done;
:: true -> attend!programmingLanguages; feedback?done;
:: true -> skip;
fi;
attend!net_prog; feedback?done;
}
```

**Fig. 7.** The Promela code representing a curriculum.

```
proctype UpdateState() {
do
:: attend?architecture ->
   if
   :: (programming_basis_java) -> architecture_calculators = true;
      feedback!done;
   fi;
:: attend?lab_languages ->
   if
   :: (programming_basis) -> imperative_programming_bases_C = true;
      data_structure_basis = true; feedback!done;
   fi;
:: attend?programmingII ->
   if
...
:: feedback?stop ->
   if
   :: (os) -> feedback!success;
   :: else -> feedback!fail;
   fi; break;
od
}
```

**Fig. 8.** The Promela code representing the process *UpdateState*.

In general, a model is made of a set of temporal constraints, using different temporal operators, which must hold at the same time. SPIN allows to specify and verify every kind of LTL formulas. It is not necessary to implement specific procedures for each operator or for each kind of formula. The following is an example of curriculum model: *not(jdbc) until(sql and relational_algebra), not(op_systems) until(basis_of_prog), not(basis_of_oo) until(basis_of_prog), eventually(basis_of_prog) implies eventually(basis_of_java_prog), eventually(database), eventually(web_services)* . For instance the first constraint means that before learning *jdbc* the student must own knowledge about *sql* and about *relational algebra*, while the last one means that soon or later the knowledge about web service must be acquired.

By using SPIN it is possible to verify that the constraints in the model hold in all the possible executions of a curriculum represented by the processes *Curriculum* and *UpdateState* described above.

The advantage of using a model checker rather than an ad hoc implementation it that it can handle any kind of LTL temporal formula and moreover it can deal with the validation of *not linear* curricula. This makes the system suitable to more realistic application frameworks. In fact, for what concerns curricula written by hand, users often do not have a clear mind that allows them to write a single sequence. In the case of curricula built by an automatic system, there are planners that are able to produce sets of alternative solutions gathered in a tree structure.

## 4 Conclusion and Further Work

In this work we have shown the integration of semantic personalization web services for Curriculum Planning and Validation within the Personal Reader Framework. The goal of personalization is to create sequences of courses that fit the specific context and the learning goal of individual students. Despite some manual post-processing for fixing inconsistencies, we used real data from the Hannover University database of courses.

Curriculum Planning and Validation offer a useful support in many practical contexts. Exchanging Courses, and taking courses at different Universities becomes more and more common in Europe. As a consequence, building a curriculum might become a complicated task for students, who must build a reasonable path through an enormous set of courses across the European countries, each described in different languages and on the basis of different keywords. As a further development, in this same scenario, it would be interesting to let our Curriculum Planning Service take into account further information that is often already associated to the course descriptions, concerning the schedule and location of courses, like for instance room-numbers, addresses and teaching hours. Such metadata could be used by the reasoner, besides the learning prerequisites and effects, in order to find a solution that fits the desires and the needs of the user in a more complete way.

Another application scenario in which the Validation service can offer an interesting support is drawn by the effort, that European universities are carrying on to reduce costs by cooperating in designing and integrating curricula. Through the Bologna Process initiative, the European Community aims at harmonizing the academic careers across

Europe and curricula integration[9]. In this context, in fact, there is a need to check if the curricula proposed by a University satisfy the requirements of the European community. A system that helps performing these checks in an automatic way would be of great help.

The Curriculum Planning Service has been integrated as a new plug-and-play personalization service in the Personal Reader framework. More information about the Personal Reader can be found at the Homepage at `http://www.personal-reader.de/`. The current and future implementations of the Curriculum Planning Demonstrator are available at `http://semweb2.kbs.uni-hannover.de:8080/plannersvc`. The Personal Reader Platform provides a natural framework for implementing a service-oriented approach to personalization in the Semantic Web, allowing to investigate how (semantic) web service technologies can provide 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[10] (Reasoning on the Web with Rules and Semantics). To the best of our knowledge the work that is the closest to our approach is the one at the School of computing from the Dublin City University and, in particular, by M. Melia et al., who are developing a course validation and correction framework (personal communication). In this approach, whenever an error will be detected by the validation phase, a correction engine will be activated. This engine will use a "Correction Model" to produce suggestions for correcting the wrong curriculum, by means of a reasoning-by-cases approach. The suggestions will, then, be presented to the course developer, who is in charge to decide which ones to adopt (if any). Once a curriculum will have been corrected, it will have to be validated again, because the corrections might introduce new errors.

We have developed a prototype of the validation system and we are currently integrating it in the Personal Reader Framework.

## Acknowledgement

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`).

## References

1. Baldoni, M., Baroglio, C., Patti, V., Torasso, L.: Reasoning about learning object metadata for adapting scorm courseware. In Aroyo, I.L., C. Tasso, e., eds.: Proc. of EAW'04: Methods and Technologies for personalization and Adaptation in the Semantic Web. (2004) 4–13
2. Baldoni, M., Baroglio, C., Patti, V.: Web-based adaptive tutoring: an approach based on logic agents and reasoning about actions. Artificial Intelligence Review (2004)

[9] `http://europa.eu.int/comm/education/policies/educ/bologna/bologna_en.html`

[10] `http://rewerse.net`

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) (September 2004) 3–39
4. European Commission, E., Training: The bologna process `http://europa.eu.int/comm/education/policies/educ/bologna/bologna_en.html`.
5. : SCORM: The sharable content object reference model (2001) http://www.adlnet.org/Scorm/scorm.cfm.
6. IMSGlobal: Learning design specifications Available at http://www.imsglobal.org/learningdesign/.
7. Koper, R., Tattersall, C.: Learning Design: A Handbook on Modelling and Delivering Networked Education and Training. Springer Verlag (2005)
8. Emerson, E.A.: Temporal and model logic. In: Handbook of Theoretical Computer Science. Volume B. Elsevier (1990) 997–1072
9. Clarke, O.E.M., Peled, D.: Model checking. MIT Press, Cambridge, MA, USA (2001)
10. Baldoni, M., Baroglio, C., Patti, V., Torasso, L.: Reasoning about learning object metadata for adapting SCORM courseware. In Aroyo, L., Tasso, C., eds.: Int. Workshop on Engineering the Adaptive Web, EAW'04: Methods and Technologies for Personalization and Adaptation in the Semantic Web, Part I, Eindhoven, The Netherlands (2004) 4–13
11. Henze, N., Krause, D.: Personalized access to web services in the semantic web. In: The 3rd International Semantic Web User Interaction Workshop (SWUI, collocated with ISWC 2006. (November 2006)
12. Baumgartner, R., Flesca, S., Gottlob, G.: Visual web information extraction with lixto. In Apers, P.M.G., Atzeni, P., Ceri, S., Paraboschi, S., Ramamohanarao, K., Snodgrass, R.T., eds.: VLDB, Morgan Kaufmann (2001) 119–128
13. Baumgartner, R., Henze, N., Herzog, M.: The personal publication reader: Illustrating web data extraction, personalization and reasoning for the semantic web. In Gómez-Pérez, A., Euzenat, J., eds.: ESWC. Volume 3532 of Lecture Notes in Computer Science., Springer (2005) 515–530
14. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall (1995)
15. Holzmann, G.J.: The model checker SPIN. Software Engineering **23**(5) (1997) 279–295