Learner Modeling on the Semantic Web*

Peter Dolog and Michael Schäfer

L3S Research Center, University of Hannover,
Expo Plaza 1, 30539 Hannover, Germany,
dolog@l3s.de

Abstract. Learners are assessed by several systems during their life-long learning. Those systems can maintain fragments of information about a learner derived from his learning performance and/or assessment in that particular system. Customization services would perform better if they would be able to exchange as many relevant fragments of information about the learner as possible. This paper presents the conceptualization and implementation of a framework which provides a common base for the exchange of learner profiles between several sources. The exchange representation of learner profiles is based on standards. An API is designed and implemented to create/export and manipulate such learner profiles. The API is implemented for two cases, as a Java API and as web services with synchronized model exchange between multiple sources. Application cases of the API are discussed shortly as well. Furthermore, the process of importing learner models from foreign systems is analyzed. Possible conflicts are discussed and conflict handling in different types of learner systems is described.

1 Introduction

Each user adapted service or application needs a user profile to perform the adaptation accordingly. In the area of education, several approaches have been proposed to collect information about users such as preferences, following clicking behavior to collect likes and dislikes, and questionnaires asking for specific information to assess learner features (e.g. tests, learner assessment dialogs, and preference forms). In addition, several tools have been designed to improve learner models by open active learner modeling. The variety of use cases are supported by such tools like maintaining and comparing the student's own and the system's believes about his knowledge [3], multiple choice questionnaires [2], collaborative peer assessment in discussions [1], and dialogues with interactive topic maps [4].

These systems can be seen as services to improve user or learner models in open environments. Different users may prefer a different style of evaluation and thus may want to choose one or more of them which are the most suitable for them to evaluate their profiles. To benefit from such heterogeneous services, an interoperable learner profile and an infrastructure to support its exchange should be provided. The following questions arise: *how to represent the learner profile*,

* This work is partially supported by EU/IST ELENA project IST-2001-37264.

how to access the learner profile, and *how to provide an extensible API to process heterogeneous profiles.*

The remainder of the paper is structured as follows: Section 2 discusses standard based representations of learner profiles, their instantiation, and mappings from internal data models. Section 3 discusses how the models can be accessed by means of a Java API, web services, querying infrastructure for RDF, and application cases which have been implemented. The import process of learner models is explained in Section 4. Finally, section 5 provides a summary and an outline of possible further work.

2 Learner Profile Exchange Model

In order to be able to exchange a learner profile between e-Learning and learner assessment systems, we need to provide explicit information about what is going to be exchanged, which values of the specific subject are considered and how the information is bound to a learner. Learner profile standards and open specifications provide us with a representation for subjects of exchange, e.g. learner performance, portfolio, preferences, learning style, certificates, evaluations, and assessment. Domain ontologies provide us with exchangeable/sharable models of domains. Such ontologies can model either the domain which will be overlaid in the learner profile, learner competencies/skills, or can model stereotype structures.

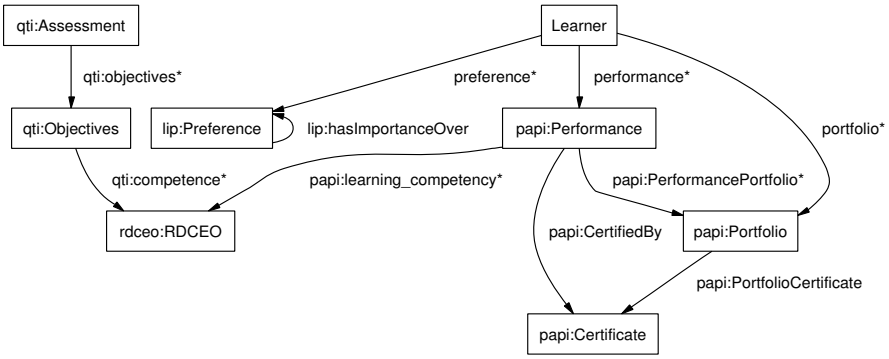

Fig. 1. An excerpt of a conceptual model for learner profile based on standards

Learner Ontology. Figure 1 depicts an excerpt of a learner profile ontology configured from fragments based on three specifications[1]. The conceptual model

[1] Refer to http://www.l3s.de/~dolog/learnerrdfbindings/ for an extended model of the learner profile.

describes a situation where a learning performance[2] of a student is exchanged as his achieved competency[3] records[4]. The competencies have been evaluated by learner assessment (e.g. tests) and were derived from learning objectives of tests[5]. Furthermore, all other educational activities, further materials, and projects created within the activities are reported within the portfolio of the performance. Additional information which is reported under preferences[6] comprises language, device, resource and learning style preferences. The standards and open specifications guarantee wider acceptance between eLearning systems and as such can be seen as good candidates for the learner exchange models.

Instantiation and Mappings from Internal Models. The tools, which use a different internal data model and would like to participate in an exchange of learner profiles, have to provide mappings between their internal data model and the exchange model. Besides that, an evidence about how a learner model was derived should be provided to allow other systems to interpret the model correctly. If we take for example an overlay model of a domain, the sub domain concepts are bound to the learner performance together with time stamps, certificates and resources which contributed to the performance. The sub concepts, referenced as competency hierarchies, are further bound to assessment resources like dialogs used, questionnaires filled in with their results, activities with concept maps performed, and so on. This information allows to trace back the computation of particular learner model fragments and to determine how they contribute to the overall integrated model.

3 Accessing the Learner Profile

Figure 2 depicts several scenarios of how to access and exchange learner profile fragments. The fragments can be accessed programmatically by the use of a Java API, the web service which exports the learner model through the API and acts as a learner model server, and through a query infrastructure for RDF repositories like Edutella [10].

Access through Java API. We build a Java API which is structured according to the learner profile fragments mentioned above. The API is meant to be used to retrieve, insert, and update the learner profiles stored in the structures described above. The API defines a class and properties for each class from the RDFS for the learner model. The interface provides access functions for getting, deleting and updating a model of the fragment. It provides further functions to derive additional information or to process more complex manipulations over referenced

[2] IEEE PAPI is being used to model performance and portfolio: http://ltsc.ieee.org/archive/harvested-2003-10/working_groups/wg2.zip.
[3] IMS reusable definition of competency and educational objectives (IMS RDCEO).
[4] Refer to http://www.imsglobal.org/ for all IMS specifications.
[5] IMS questions and test interoperability (IMS QTI).
[6] IMS learner information package (IMS LIP).

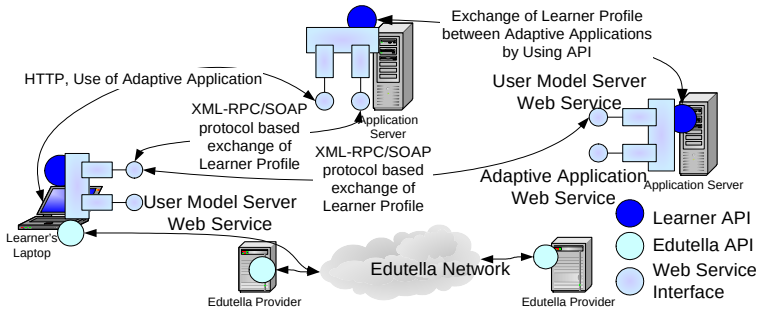

Fig. 2. The use of the API in several scenarios

information types as well. The API is implemented for the RDF representation (instances of the RDFS described above). The API is easily extensible by providing further specializations if additional extensions and interface implementations for local repositories and data models are needed.

Access through API as Web Services. The second implementation is provided through web services where several clients can access one model which is persistent on one server. The server holds the main model, i.e. the data of a learner profile gathered from several sources, and handles all requests from the clients. Each client is uniquely identified at the server and can be used by a browsing or assessment system. Furthermore, a client can be used by other learning systems which want to make use of the learner profiles or which want to contribute to them. The model can be accessed directly by invoking functions of a web service or in a synchronized replicated way; i.e. each client has its own repository which is synchronized with the main server every time a change occurs. The web services framework can be used in a distributed way as well (several servers exchanging learner models between each other).

Retrieval through RDF querying infrastructure. The learner profiles are created in RDF. Therefore, a query infrastructure for RDF data is another access option. Edutella provides a datalog-based language to query RDF data provided in a distributed P2P environment. This option enables to collect various fragments by utilizing for example the algorithm from [5]. Another advantage of the P2P sharing infrastructure used with the learner profiles is that it can facilitate an expert finding based on the provided profile which can be queried by people who need a help in learning.

Recent Application Cases of the Framework. The API has been tested at a simple browsing and dialog system (Learner Browser) and with the UML-guide system [7]. In the UML-Guide the API is used to record clicking behavior of the learner in a knowledge map by means of events triggered when a particular knowledge map item is clicked. In the Learner Browser, the profile can be

browsed through several categories of a learner data with possibility to use it for self- reflection; i.e. to update simple categories like preferences, add a competence based on an evaluation by a test, and so on. Further implementations towards other assessment services are envisaged.

4 Importing Learner Models

The learner profile framework which was described above offers different and independent learner systems the possibility to collaboratively build and use learner profiles. The individual learner systems might access central learner models from other systems and perhaps contribute to them, or they might create and maintain their own learner models. As a consequence, it is quite possible that the information about one or more learners is scattered across different systems and learner models.

In this context, where multiple heterogeneous systems want to make use of and contribute to learner models, it is important to provide the means to import external (fragments of) learner models. Special attention has to be paid to maintaining data integrity and resolving conflicts during an import.

4.1 Basic Import Scenario

The basic import scenario is shown in figure 3: System A wants to import the learner data from a separated and possibly heterogeneous system B.

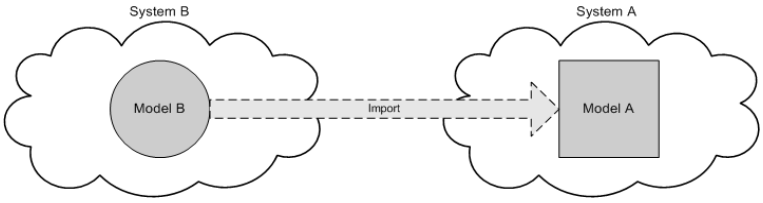

Fig. 3. The basic import scenario

The system which processes the import can access both models. Furthermore, it has access to a set of RDF Schema (RDFS) files which hold the description of the structure of the learner model of system A. The RDF Schema files on which model B is based might be known to a user who supervises the import, but it can not be taken for granted that they are available for the automated system which processes the import.

Different scenarios exist in which a learner model has to be imported. For instance, when two model servers want to merge their data, it is necessary to import one of the models into the other one, either on both sides or centrally. Another scenario would be some sort of browsing system for learner data which

does not require a continuous update of its learner model (which would be possible with the synchronized service mentioned in Section 2) either because it is not necessary or because it does not have permanent internet access. This system might periodically update its data unilaterally with the learner profile data from another source, or bilaterally also submit its own learner profiles to the other source in order to share the data it collected since the last exchange of learner models.

4.2 The Import Process

Several possible problems and conflicts arise from the basic scenario, and therefore learner profile data has to be thoroughly examined regarding its form and content before importing it. Figure 4 depicts the sequence of the necessary checks:

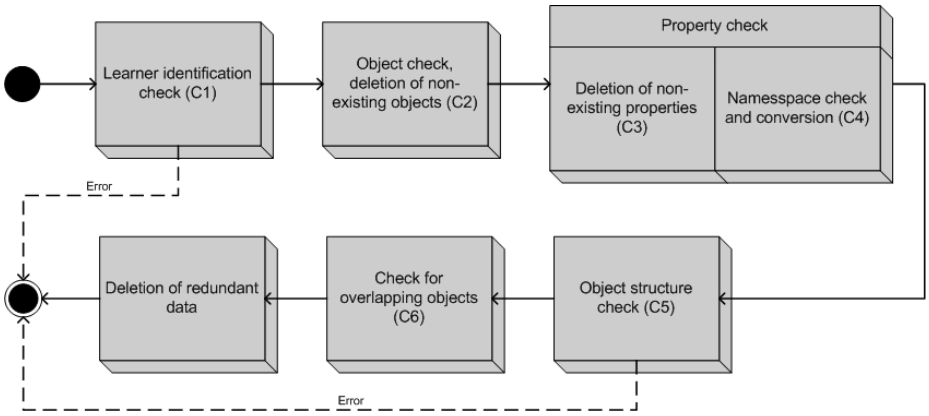

Fig. 4. The import process

The RDF Schema definitions of model of system A will be used in several steps of the import. The schema files can be analyzed to extract information about what objects exist, what properties they have, and what kind of values these properties can hold.

4.3 Import Checks

Learner identification check (C1) The import process starts with the check of the learner identification of each learner in the profiles which are being imported. The identification mechanism of the learner profile described in [5] allows the use of local learner identifiers in different systems. Therefore, when a profile of a learner is being imported from a foreign system B, it is necessary to check the identifier and to map it to the correspondent local identifier in system A. If the

learner does not yet exist in system A, then a new learner will be created with a new local identifier.

When a conflict occurs, it is necessary to abort the import of either the single learner or the complete import process of the model. If the check was successful, the import process will continue with the check of the objects and properties of the learner data.

Object check (C2) A learner profile object could for example be an object of the "papi:Performance" class. If the name of an object of model B is erroneous, and therefore can not be found in the schema definitions of model A, the object can not be imported. Unknown objects could also be system-specific extensions which do not exist in the general learner profile definitions. It is not possible to automatically determine what kind of object this object should be. The data of this object has to be discarded.

The problem that model A and model B have different object structures can for example occur when the two models are based on different versions of the learner profile, or simply through a malformed and wrong creation of one of the models.

Deletion of nonexisting properties (C3) Likewise, if a property can not be found in the structure description, then it has to be deleted. This data would have to be changed and added to model of system A manually by someone who knows both model structures.

Namespace check and conversion (C4) Within limits, it is possible to import properties whose namespaces are invalid. Based on the schema definitions, it is possible to determine which property/ properties a certain object can have in the structure of model A. This can be used to check whether an object's property from model B has a wrong or outdated namespace, or if it is a completely unknown property. If only the namespace is wrong, then the property can be changed accordingly and be imported into model A.

Object structure check (C5) The structure of each object of model B has to be checked in order to ensure that it conforms to the learner profile definitions. The values of each object's properties have to have the format (e.g. string or integer values, or complex objects) which is specified in the RDF schema files.

The structure of the object being imported is checked against the schema definition. If the format, type, or value violates the schema definition, the object is discarded. Such object can only be imported manually; otherwise it might cause critical errors in the learner systems which use this data.

Check for overlapping objects (C6) Overlapping objects can for example occur when both models hold data about the same learner or the same institution. If model A and model B have information about an object which exists in both models, this overlapping has to be analyzed in order to avoid conflicts.

In order to be able to determine whether overlapping objects exist, it is necessary to compare all objects of both models. Furthermore, if an overlapping object exists, it is necessary to analyze the RDF schema definition of this object.

If an object exists in both, model from system A and model from system B, this is not necessarily a problem. Actually, this will be the standard case in the context of learner systems which cooperatively create and use learner profiles. It may be that the objects hold exactly the same data, and therefore no conflict exists. However, if the two objects contain different information, the properties of the objects have to be further analyzed. For each affected property, the RDF schema definitions of the learner profile have to be checked whether the property allows multiple values. If this is the case, the additional value for this property from model B can directly be added to the corresponding object in model A. If the property only allows one value, then a preset rule has to be specified. Possible rules are:

1. Date: The property value which has the newer creation date will be used. However, this requires that all properties of the learner profile contain this information, and that each learner system which creates and changes learner data always sets this property. This can not be guaranteed when heterogeneous learner systems provide the learner models.
2. Priority: One of the models (either model from system A or model from system B) is defined to have a higher priority. The data of the model with the lower priority will be discarded; thus only non-overlapping data will be imported. Although this is a rule which can be easily implemented, this approach can cause inconsistencies in the model data, because it is possible that correct data will be replaced by outdated data or newer data will be discarded and not imported. But in a scenario of a totally automated import system which works with data from different learner systems it is the only feasible rule.
3. Error: The entire import will be aborted.

Deletion of redundant data In addition to the already specified checks, the deletion of all redundant data is necessary at the end of the process. It would not cause any errors or problems if unused data would be included in model A, but it is a waste of memory and processing time if the data would be imported.

4.4 Consequences of a model import

As it can be seen, the possible consequences of a model import depend largely on the configuration of the different checks and rules.

In a simple browsing system which does not allow editing of the data, the rules can be set so that model being imported has the higher priority. This scenario does not contain any critical conflicts, as the data from model B will always replace the existing data.

In an e-learning system which regularly exchanges data with another system, it is quite probable that overlapping data exists. Whether this will cause inconsistencies between the two data sets depends on the chosen rule for this problem and on the available data, like date properties, with the last change.

In a fully automated system, this problem can only be addressed by reporting the problems and conflicts so that they can be solved manually by a supervisor. If the import was started manually, these problems can be reported directly during the import and can be solved on-the-fly.

5 Conclusions and Further Work

We have described a framework which utilizes standards to make learner profiles interoperable. A user model server similar to the one described in [8] is implemented by making use of the framework. A Java and web service API was implemented making use of the framework to allow other systems to plug into the standard based learner modeling component. We have described mappings between learner models when they are to be imported into one system and they use schema similar to the one proposed in the paper.

There are still two main issues which remain unsolved in general: how to deal with anonymous nodes, and how to map between schemas if they are more heterogeneous. In both cases, schema integration approaches and semantic mappings between ontologies as described for example in [11] should be studied.

Such schema integration approaches are especially useful in P2P environments similar to the one [12, 6, 9].

References

1. Susan Bull, Paul Brna, Sonia Critchley, Koula Davie, and Corina Holzherr. The missing peer, artificial peers and the enhancement of human-human collaborative student modelling. In S.P. Lajoie and M. Vivet, editors, *Proc. of International Conference on Artificial Intelligence in Education*, pages 269–276. IOS Press, 1999.
2. Susan Bull and Theson Nghiem. Helping learners to understand themselves with a learner model open to students, peers and instructors. In Paul Brna and Vania Dimitrova, editors, *Proc. of Workshop on Individual and Group Modelling Methods that Help Learners Understand Themselves, International Conference on Intelligent Tutoring Systems*, pages 5–13, 2002.
3. Susan Bull and Helen Pain. Did I say what I think I said, and do you agree with me? In J. Greer, editor, *Proceedings World Conference on Artificial Intelligence in Education, AACE*, pages 501–508, 2002.
4. Vania Dimitrova. Style-olm interactive open learner modelling. *International Journal of Artificial Intelligence in Education*, 13, 2002.
5. Peter Dolog. Identifying relevant fragments of learner profile on the semantic web. In *Proc. of SWEL'2004 — International Workshop on Semantic Web for eLearning, International Semantic Web Conference 2004*, Hiroshima, Japan, November 2004.

6. Peter Dolog, Nicola Henze, Wolfgang Nejdl, and Michael Sintek. Personalization in distributed e-learning environments. In *Proc. of WWW2004 — The Thirteen International World Wide Web Conference*, New Yourk, May 2004. ACM Press.
7. Peter Dolog and Wolfgang Nejdl. Using UML and XMI for generating adaptive navigation sequences in web-based systems. In Perdita Stevens, Jon Whittle, and Grady Booch, editors, *Proc. of UML 2003 — The Unified Modeling Language. Model Languages and Applications. 6th International Conference*, volume 2863 of *LNCS*, pages 205–219, San Francisco, CA, USA, October 2003. Springer.
8. Jozef Fink and Alfred Kobsa. User modeling in personalized city tours. *Artificial Intelligence Review*, 18(1):33–74, 2002.
9. Alon Y. Halevy, Zachary G. Ives, Dan Suciu, and Igor Tatarinov. Schema mediation for large-scale semantic data sharing. *VLDB J.*, 14(1):68–83, 2005.
10. Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Micahel Sintek, Ambjoern Naeve, Michael Nilsson, Matthias Palmér, and Tore Risch. EDUTELLA: a P2P Networking Infrastructure based on RDF. In *In Proc. of 11th World Wide Web Conference*, Hawaii, USA, May 2002.
11. Natasha Noy and Heiner Stuckenschmidt. Ontology alignment: An annotated bibliography. In Y. Kalfoglou, M. Schorlemmer, A. Sheth, S. Staab, and M. Uschold, editors, *Semantic Interoperability and Integration*, number 04391 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2005. <http://drops.dagstuhl.de/opus/volltexte/2005/48>[date of citation: 2005-01-01].
12. Julita Vassileva, Gordon McCalla, and Jim Greer. Multi-agent multi-user modelling in I-Help. *User Modeling and User-Adapted Interaction*, 2002.