Towards Adaptable and Reusable RDF-based Query Patterns for Trace-Based Learner Modelling

Lemya Settouti[1,2], Nathalie Guin[1], Vanda Luengo[2], and Alain Mille[1]

[1] Université de Lyon, CNRS,
Université Lyon 1, LIRIS, UMR5205, F-69622, France
{Lemya.Settouti, Nathalie.Guin, Alain.Mille}@liris.cnrs.fr
[2] Université de Grenoble, LIG, France
Vanda.Luengo@imag.fr

Abstract. This paper defines a framework to describe Learner Modelling (LM) process based on interactions traces. This framework includes an RDF-Based representation of knowledge models that can be used by a LM designer. This framework supports also the description of reusable and adaptable SPARQL-based query patterns. These patterns enables the LM designer to calculate and infer learner profile elements for different TEL systems. We define formally the notion of query pattern and the adaptation algorithm with an illustration of its application in the context of two TEL systems in order to detect the strategy "gaming the system".

1 Introduction

The personalization of Technology-Enhanced Learning systems (TEL) is done through the Learner model, which collects information about the Learner. Learner Modelling (LM) is the process of creating learner model from observed learner's behaviors [5]. Learner's behaviors refer to historic of actions, results of these actions including intermediates ones [15]. To acquire such learner's behaviors, a common approach is to observe and produce trace of learner's interactions with TEL systems.

Learner Modelling (LM) is a complex task that must take into account severals components: (1) Representation of observations about learner's behavior(including learner-system interactions traces and their interpretations), (2) Representation of domain knowledge (including topics concepts, facts, procedures or methods that experts used to achieve task or solve a problem), (3) Diagnosis and reasoning function allowing to infer assumptions about learner and (4) Representations of assumptions about individual learner characteristics (e.g. performance, skills, misconceptions, strategies,interests, etc.).

To deal with such complexity, many LM frameworks have been designed and proposed [5,11,8]. However, most of frameworks do not pay much attention on the reusing issues. Actually, the issue of LM component reusing has been implicitly investigated in the context of *educational ontologies* [7,1,6] (Interaction model, Domain model, Learner model, etc.), however, without addressing the issue of the *diagnosis reusing*. To the best of our knowledge, there is no LM framework supporting the diagnosis reusing specially in the context of their application within TEL systems. The issue of reusing diagnosis is a complex problem mainly due to two reasons :

- From a conceptual point of view, LM is mostly domain-dependent. In the practice, it is hard (and even impossible) to reuse LM domain-dependent components specially diagnosis part. One solution is to identify diagnosis-parts which are domain-independent from domain-dependent ones. We need a framework making a clear separation between such kinds of diagnosis and defining a generic diagnosis model domain-independent which can be exploited in several contexts.
- From a technical point of view, LM diagnosis are encoded in many forms making their reuse complex (or impossible). We can identify two main classes of diagnosis encoding that are used in LM frameworks: (1) Diagnosis encoded with high level programming languages (i.e. the host programming language used for TEL as Java, Php, C++, etc.), (2) Diagnosis encoded with declarative languages equipped with reasoning capabilities. Both forms of encoding are ill-adapted to diagnosis reuse. For the first class of languages, the access and comprehension of diagnosis are difficult and even impossible in most TEL. For the second class, there exist many syntaxes (and particularly severals semantics). Importantly, such diagnosis encoding rely on textual representation making difficult reusing. We need a framework dealing with diagnosis representation facilitating reusing.

To deal with these problems, this paper aims at defining a framework supporting LM construction and LM diagnosis reusing. Our contribution is twofold : firstly, we define a framework based on trace of learner's interactions. While our framework supports description of LM dependent or independent on specific domain [13], we will focus in this paper on LM domain-independent parts. Actually, the separation between models that are domain dependent and independent is key element in our framework, which make reusing diagnosis a tractable issue. Our second contribution to achieve a pertinent reuse of LM components presents the notion of *generic query pattern*. Intuitively, generic query patterns are reusable solutions describing diagnosis rules that can be adapted to a specific context and thus that can be used in several TEL systems. To describe generic query patterns, we use an RDF-Based SPARQL rule representation as encoding language. This representation encourage reusing or sharing.

The remainder of paper is organised as follows: Section 2 describes the Trace-Based Learner Modelling framework. Sections 3 describes formally a notion of query patterns that ca be reused in several TEL Systems. Section 4 presents a generic query pattern for detecting when the learner is gaming with the TEL system and an describes how to adapt this pattern in the context of two TEL systems.

2 Trace-Based Learner Modelling Framework

The Trace-Based Learner Modelling framework is an extension of Trace-Based System (TBS) framework [14]: an approach defined to describe, reason and exploit interaction traces in order to provide TEL personalization. We extend TBS framework by defining new knowledge models to support Learner profile building. The figure 1 shows the building of learner profile that requires carrying out three main processes. Before define the process of building learner profile, we introdice the notions defined within Trace-Based System (TBS) framework over the concept *modeled trace*. Trace-Based SystemTBS collects the observed data from TEL system (e.g. log files) according to the *Trace Model*. the *Trace Model* is a general ontology that defines and structures explicitly content of interaction traces from various activities not specifically the learning ones.The *Modelled trace* is the sequence of observed elements tracked within systems, associated explicitly to a *Trace Model*.

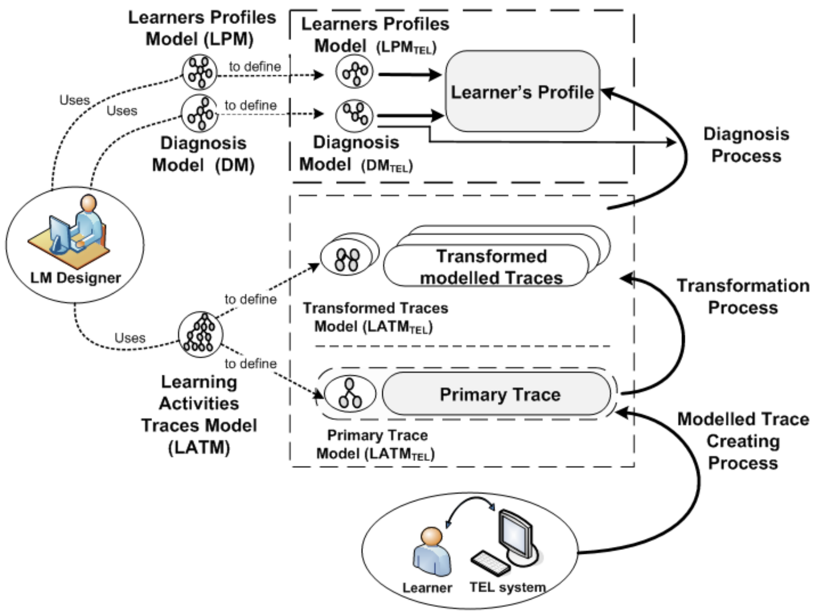

Learners Profiles Model (LPM)
Learners Profiles Model (LPM_{TEL})
to define
to define
Uses
Uses
Learner's Profile
Diagnosis Model (DM)
Diagnosis Model (DM_{TEL})
Diagnosis Process
LM Designer
Transformed modelled Traces
Transformation Process
to define
Uses
to define
Transformed Traces Model ($LATM_{TEL}$)
Learning Activities Traces Model (LATM)
Primary Trace
Primary Trace Model ($LATM_{TEL}$)
Modelled Trace Creating Process
Learner
TEL system

Fig. 1. The Trace-Based Learner Modelling Framework

The first process consists in creating a *primary modelled trace* according to TBS framework which collects the observed data from TEL system (e.g. log files). *Modelled trace* is the sequence of observed elements tracked within TEL system, associated explicitly to a *Trace Model*[14]. *Trace Model* refers to the ontology that defines and structures explicitly content of an interaction trace [14].

We defined within our framework the Learning Activities Trace Model (LATM) as an extension of the *Trace Model*. LATM is an RDF vocabulary which enables to describe activities observed by several TEL systems supporting individual learning. LATM models interaction traces by specifying how observed elements are typed and categorized, what relations may exist between them and what attributes further describe each of them. LATM assumes that observe a learning activity session consists in observing the several actions performed by the learner and in observing learning objects (problem, problem solution, assessment, etc.) manipulated by actions. Learning session observation may also include information about learner's identification and information about the TEL system. To create the*primary modelled trace*, the LM designer defines the Learning Activities Trace Model of a specific TEL system ($LATM_{TEL}$) by using (LATM), which elements of $LATM_{TEL}$ are instances of elements of $LATM$ and the *modelled trace* is the sequence of observed elements associated to $LATM_{TEL}$ elements. *The primary modelled trace* is the first modelled trace created according to ($LATM_{TEL}$).

The second process consists in performing operations on traces like applying filters, aggregating and computing elements, etc. The result of this process is *transformed modelled traces* which can be more reusable and exploitable than the primary trace. Actually, the collected information within primary trace are low-level of abstraction to be exploited in order to infer the learner profile. The LM designer defines the *transdormed trace model* by using LATM in order to transform the primary modelled trace, then he/she use query and transformation languages defined by TBS in order to interpret and improve the primary trace.

The last process consists in performing diagnosis. The diagnosis process enables to construct the learner's profile by querying modelled traces, calculated data and inferring learner's strategies and/or learner's knowledge. The learner's profile is built from: the *modelled trace* elements (primary or transformed), the specific TEL system Diagnosis Model elements (DM_{TEL}) and from the Learner Profile Model of a specific TEL system (LPM_{TEL}). The LM designer defines the LPM_{TEL} by using the Learner Profile Model (LPM) proposed within our framework. LPM is an RDF vocabulary which enables to describe the structure of learners profiles. LPM is composed of: (1) *Calculated Data* that describes data derived directly from $LATM_{TEL}$ or from other calculated data (e.g. average time spends on solving a problem), (2) the *Strategy* performed by the learner in order to solve problems and (3) qualitative or quantitative estimation about the learner *Knowledge*.

The LM designer defines the DM_{TEL} by using the Diagnosis Model (DM) proposed within our framework. DM is composed of *domain knowledge* elements and *queries and rules diagnosis* elements. The *domain knowledge* refers to the knowledge used in both overlay and buggy LM approaches. The *queries diagnosis* refer to a RDF representation of SPARQL queries. Instead of relying on the textual representation of SPARQL expressions, we define an RDF Schema that can be easily converted to textual SPARQL queries. The main advantage of RDF-based SPARQL representation is that queries expressed by the LM designer can be stored as RDF together with the knowledge models proposed within our framework. This enables linking models elements with the associated SPARQL queries as well as easy sharing and reuse of rules and queries as part of learner modelling framework.

3 Query Patterns for Learner Modelling

We propose the notion of *query pattern* in order to deal with reusability, . We define a query pattern as a general and reusable query that describes how to use, process and interpret learners' interactions in order to identify relevant information. In our framework, the goal of query patterns is to obtain on the one hand learners' individual observable behaviors during their learning activities and on the other hand to infer learners' competences and problem-solving strategies. To be generic, query patterns must be able to be used in a wide range of TEL systems. In our approach, using query patterns requires that a TEL system should have a model of traces, a model of diagnosis and a model of learner profile consistent with the models defined in our framework (i.e. LATM, LPM and DM models). To be applied in different contexts and domains, query patterns have to describe, beyond the way to compute the needed information, the manner to be adapted to the specific models of a TEL system. In this section, first we define the notion of *query pattern*, then we present the generic algorithm for adapting query patterns to specifics TEL systems.

Formally, a query pattern is defined as couple (Q_p, δ) where

– $Q_p = \{q_0, ..., q_n\}$ is an ordered set of SPARQL queries. SPARQL is used to express queries on RDF graphs (i.e. matching the [where-part] of SPARQL query) in order to obtain RDF graphs in a given context (SPARQL query [construct-part]). The defined SPARQL queries that compose query patterns use elements of knowledge models defined in our framework (i.e. elements of LATM, LPM and DM models).
– δ is an adaptation function enabling to reuse SPARQL queries by constructing a new queries Q'_p (such as $Q'_p = \delta(Q_p)$) adapted to a specific TEL system. More precisely, δ adapt $\{q_0, ..., q_n\}$ which are described in terms of elements of knowledge models (i.e. elements from LATM, LPM, DM) by constructing a set of new SPARQL queries $\{q'_0, ..., q'_n\}$ which are described in terms of a specific TEL system models elements (i.e. elements from $LATM_{TEL}$, LPM_{TEL}, DM_{TEL}).

The algorithm 1 describes the process of rewriting general SPARQL queries to specific SPARQL queries. This process substitutes each element from general models $LATM$, LPM and DM by elements from the specific models (respectively $LATM_{TEL}$, LPM_{TEL}, DM_{TEL}). In this paper, for the sake of simplicity, we will show just the part of δ enabling the substitution of $LATM$ elements by the corresponding elements of $LATM_{TEL}$.

Firstly, the input data for the algorithm 1 which reify the adaptation function δ are respectively: (1) an ordered set of general SPARQL queries $\{q_0, ..., q_n\}$, (2) an ordered set of couples (c_i, c'_i) meaning that each concept c_i of $LATM$ model should be substituted by the corresponding concept of $LATM_{TEL}$ c'_i (with c'_i is an instance of c_i), (3) an ordered set of couples (p_i, p'_i) meaning that each attribute or relation p_i of $LATM$ should be substituted by the corresponding attribute or relation of $LATM_{TEL}$ p'_i. The output data for δ is an ordered set of adapted SPARQL queries $\{q'_0, ..., q'_n\}$.

Algorithm 1: Adaptation function δ for a Query Pattern

```
Input  : - An ordered set of SPARQL queries $Q_p = \{q_0, ..., q_n\}$
         - An ordered set $S_c = \{(c_0, c'_0), ..., (c_n, c'_n)\}$ where $c'_i$ is an obseltype
           $\in LATM_{TEL}$ substitutes an obseltype $c_i \in LATM$
         - An ordered set of $S_p = \{(p_0, p'_0), ..., (p_m, p'_m)\}$ where $p'_i \in LATM_{TEL}$
           substitutes an attribute or relationtype $p_i \in LATM$
         - A list $L(q_i)$ of triple patterns associated to a $q_i \in Q_p$
Output: - An ordered set of adapted SPARQL queries $Q'_p = \{q'_0, ..., q'_n\}$
1  begin
2      $Q'_p \leftarrow \emptyset$
3      foreach $q_i \in Q_p$ do
4          $q'_p \leftarrow \emptyset$
5          $L(q) \leftarrow \emptyset$
6          foreach $< subject_i, predicate_i, object_i > \in q_i$ do
7              if $< subject_i, predicate_i, object_i > \notin L(q_i)$ then
8                  if $predicate_i = rdf{:}type$ then
9                      if $object_i = c_k$ and $(c_k, c'_k) \in S_c$ then
10                         if $\exists < subject_j, predicate_j, object_j > \in q_i$ where
                           $object_j = subject_i$ then
11                             - $q'_p \leftarrow q'_p \cup < subject_j, predicate_i, c'_k >$
12                             - Put $< subject_j, predicate_j, object_j >$ in the list
                               $L(q_i)$.
13                     if $object_i = p_h$ and $(p_h, p'_h) \in S_p$ then
14                         if $\exists < subject_j, predicate_j, object_j > \in q_i$ where
                           $predicate_j = subject_i$ then
15                             - $q'_p \leftarrow q'_p \cup < subject_j, p'_h, object_j >$
16                             - Put $< subject_j, predicate_j, object_j >$ in the list
                               $L(q_i)$.
17                 else
18                     $q'_p \leftarrow q'_p \cup < subject_i, predicate_i, object_i >$
19         $Q'_p \leftarrow Q'_p \cup q'_p$
20 return $Q'_p$
```

The general principle of the algorithm 1 can be described abstractly as follow: for each q_i composing the query pattern (line 3), the algorithm substitutes every triple pattern $< subject_i, predicate_i, object_i > \in q_i$ (line 6) that is linked to elements from $LATM$ and which has not been processed by the algorithm previously (line 7), by triple pattern with elements related to a specific TEL system $LATM_{TEL}$. This substitution consists in either replace the concept c_k of $LATM$ by the corresponding concept of $LATM_{TEL}$ c'_k (lines 9-12), and replace the attribute or relation p_k of $LATM$ by the corresponding attribute or relation of $LATM_{TEL}$ p'_k (lines 13-16).

We have defined in our framework several query patterns (about 50 patterns) enabling to obtain relevant information about : (1) learner's observable behaviors (consult course before to solve a problem, request help before to answer, ask for assessment after given a solution, etc.), (2) learner's performances (total learner attempts on problems, success rate on problem, average time spent on solving a problem, etc.) and (3) learner's strategies as *gaming the system*. The adaptation function δ is generic since it is reusable for all query patterns defined in our framework. Actually, the adaptation function make query pattern generic by supporting the adaptation of queries to specific TEL systems.

The patterns defined in our framework can be reused in several TEL Systems. In this paper, we present as example the query pattern enabling to detect when the learner is gaming with a TEL system. We exemplify our proposal by showing how to reuse and adapt the query pattern to detect gaming the system within TEL systems used in different domains, like copex-chimie[4] and AMBRE-add[10] TEL systems.

4 Query Pattern to design gaming the system

According to Baker et al [2], gaming the system is defined as *attempting to succeed in an educational environment by exploiting properties of the system rather than by learning the material and trying to use that knowledge to answer correctly.* Gaming the system is a strategy that is a sign of learner's lack of motivation, or a sign of difficulties encountered by the learner in the problem resolution steps. Generally, gaming the system can be detected from incorrect answers given quickly by the learner with ask for help. For example, when the learner asks for help quickly and in a repetitive way until the system gives him the good answer or when he/she gives quickly and automatically answers to the system until the system identifies the good answer and thus allows the learner to advance in the exercise.
We present in this section how to adapt the query pattern to detect gaming the system starting from learner's interactions with a TEL system. We begin by presenting trace models of the two TEL systems: copex-chimie and AMBRE-add, then we show the process to adapt such query pattern for both of them.

4.1 Trace models for AMBRE-Add and copex-chimie TEL systems

copex-chimie is a Web-based TEL system offering learners the means to design experimental procedure in the context of chemical practices. The learner has for example to determine the concentration of the red dye in a grenadine syrup using spectrophotometry by writing an experimental procedure [4]. The experimental procedure is structured into three steps (preparation of a series of standard solutions; obtain the points of the standard curve; determine the concentration of the dye in the grenadine syrup). For each step, the learner selects adequate actions among a list of eight actions (rinse an equipment, make a dilution, etc.). For each action added in the experimental procedure step, the parameters describing the action have to be set by the learner. The learner can ask the system for assessment of the procedure.The system evaluates the procedure, step by step and it points out to the learner the errors detected in the procedure. To observe a Copex-chimie learning session, several elements are tracked. The figure 2 (B) shows a part of copex-chimie trace model corresponding to the observation of the first step of an experimental procedure (preparation of series of standard solutions). This model is composed of : `Procedure`, `ProcedureStep`, `ProcedureAction` and `ProcedureStepEvaluation` as observed elements types, `idProcedureStep` as attribute and `hasProcedureStep`, `hasAction` and `hasEvaluation` as relationships.

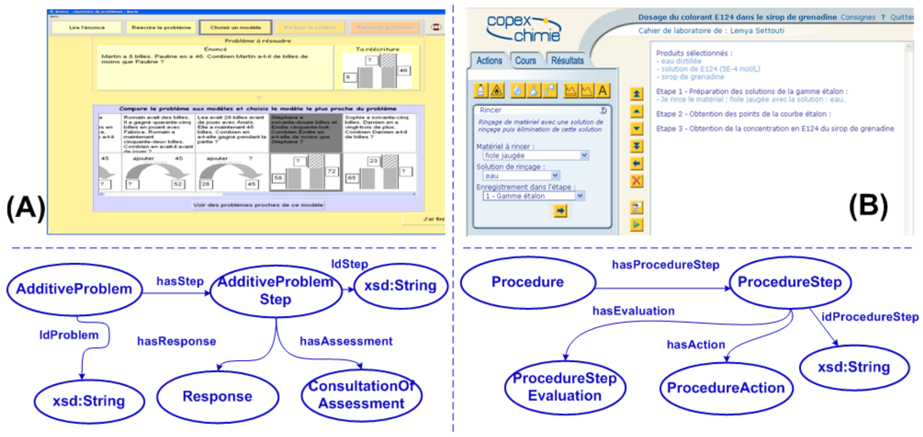


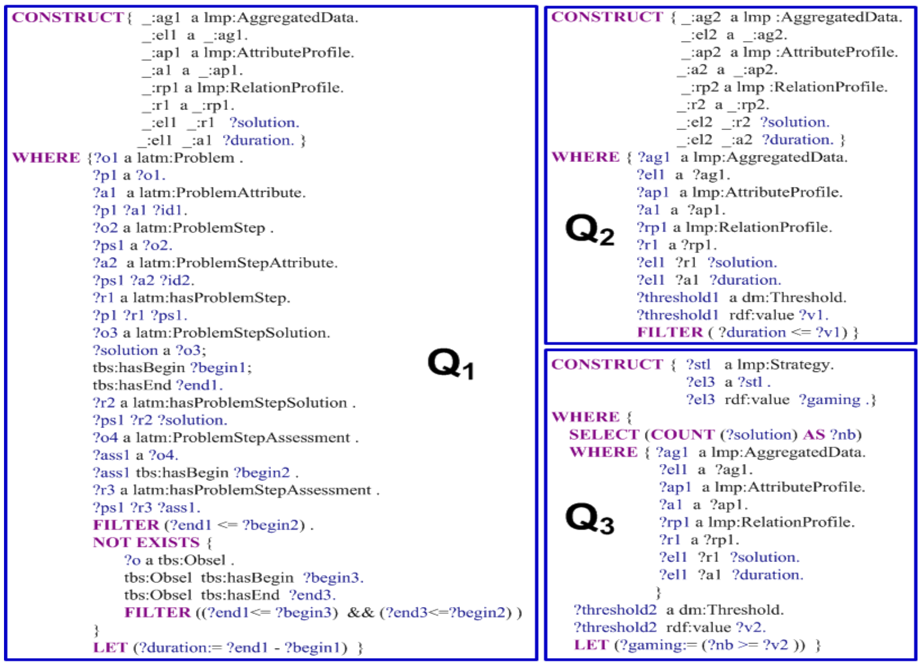
Fig. 2. (A) AMBRE-add Trace model (B) copex-chimie Trace Model

AMBRE-add is an ITS that enables learners to learn methods based on a classification of problems in order to solve additive problems [10]. During a AMBRE-add learning session, the learner begin by observing typical examples of solved problems, then he/she tries to solve problems in five steps. In the first step, the learner reads the problem statement and tries in the second step to reformulate the problem to be solved by identifying the relevant features of problem statement. The third and fourth steps consist in choosing among the examples that he/she has seen before, the problem that seems the nearest to the problem to be solved and adapt the solution. In the last step, the learner stores the new problem with a typical problem that represents a group of existing problems of the same class. For each step, the learner can ask the system for help or to evaluate his/her responses, then the system gives him/her appropriate feedbacks.

As Copex-chimie, AMBRE-add is fully tracked system and several elements are observed. The figure 2 (A) shows a part of AMBRE-add trace model corresponding only to the step 3 that consists in choosing an example of a typical problem. This model is composed of: `AdditiveProblem`, `AdditiveProblemStep`, `Response` and `ConsultationOfAssessment` as observed elements type, `idProblem` and `idStep` as attributes and `hasStep`, `hasResponse` and `hasAssessment` as relationships.

The two defined trace models are both derived from the learning activity trace model $LATM$ and are used to detect when learners are gaming with these two systems.

4.2 Query patterns to detect gaming the system



```
CONSTRUCT{ _:ag1  a lmp:AggregatedData.
           _:el1  a  _:ag1.
           _:ap1  a lmp:AttributeProfile.
           _:a1  a  _:ap1.
           _:rp1 a lmp:RelationProfile.
           _:r1  a _:rp1.
           _:el1  _:r1   ?solution.
          _:el1  _:a1  ?duration. }
WHERE {?o1 a latm:Problem .
       ?p1 a ?o1.
       ?a1  a latm:ProblemAttribute.
       ?p1 ?a1 ?id1.
       ?o2 a latm:ProblemStep .
       ?ps1 a ?o2.
       ?a2  a latm:ProblemStepAttribute.
       ?ps1 ?a2 ?id2.
       ?r1 a latm:hasProblemStep.
       ?p1 ?r1 ?ps1.
       ?o3 a latm:ProblemStepSolution.
       ?solution a ?o3;
       tbs:hasBegin ?begin1;
       tbs:hasEnd ?end1.
       ?r2 a latm:hasProblemStepSolution .
       ?ps1 ?r2 ?solution.
       ?o4 a latm:ProblemStepAssessment .
       ?ass1 a ?o4.
       ?ass1 tbs:hasBegin ?begin2 .
       ?r3 a latm:hasProblemStepAssessment .
       ?ps1 ?r3 ?ass1.
       FILTER (?end1 <= ?begin2) .
       NOT EXISTS {
           ?o a tbs:Obsel .
           tbs:Obsel  tbs:hasBegin  ?begin3.
           tbs:Obsel  tbs:hasEnd  ?end3.
           FILTER ((?end1<= ?begin3)  && (?end3<=?begin2) )
       }
       LET (?duration:= ?end1 - ?begin1)  }
```

```
CONSTRUCT { _:ag2  a lmp :AggregatedData.
            _:el2  a  _:ag2.
            _:ap2  a lmp :AttributeProfile.
           _:a2  a  _:ap2.
            _:rp2 a lmp :RelationProfile.
           _:r2  a _:rp2.
           _:el2  _:r2  ?solution.
           _:el2  _:a2  ?duration. }
WHERE { ?ag1  a lmp:AggregatedData.
        ?el1  a  ?ag1.
        ?ap1  a lmp:AttributeProfile.
        ?a1  a  ?ap1.
        ?rp1 a lmp:RelationProfile.
        ?r1  a ?rp1.
        ?el1  ?r1  ?solution.
        ?el1  ?a1  ?duration.
        ?threshold1  a dm:Threshold.
        ?threshold1  rdf:value ?v1.
        FILTER ( ?duration <= ?v1) }
```

```
CONSTRUCT {  ?stl   a lmp:Strategy.
             ?el3  a ?stl .
             ?el3  rdf:value  ?gaming .}
WHERE {
  SELECT (COUNT (?solution) AS ?nb)
  WHERE { ?ag1  a lmp:AggregatedData.
          ?el1  a  ?ag1.
          ?ap1  a lmp:AttributeProfile.
          ?a1  a  ?ap1.
          ?rp1 a lmp:RelationProfile.
          ?r1  a ?rp1.
          ?el1  ?r1  ?solution.
          ?el1  ?a1  ?duration.
         }
   ?threshold2  a dm:Threshold.
   ?threshold2  rdf:value ?v2.
   LET (?gaming:= (?nb >= ?v2 ))  }
```

Fig. 3. General SPARQL Queries to detect Gaming the system

To detect gaming the system, we defined the gaming query pattern denoted by P_g. The gaming pattern P_g is composed of an ordered set of SPARQL queries $\{q_1, q_2, q_3\}$.

The first query q_1 (figure 3) matches the $LATM$ elements (in the where part) in order to calculate durations of solutions which are followed directly by an asking for assessments. This query constructs an intermediate RDF graph derived from `CalculatedData`, the element of LPM that is composed of solutions and their durations. The second query q_2 (figure 3) matches the result of the first query q_1 in order to filter only solutions which their durations are less than some threshold duration specified by the LM designer (`?threshold1`). This query constructs an other intermediate RDF graph derived from the element `CalculatedData` representing the short solutions and their durations. The last query q_3 (figure 3) matches the result of the second query q_2. This query q_3 enables to calculate the number of solutions which their durations are less than some threshold duration, and compare it with an other threshold defined by LM designer (`?threshold2`). If the solutions number is higher than the specified threshold then the gaming is detected by constructing as `Strategy` a `gaming the system` element with the value *true* in the learner profile.

In order to detect the gaming within TEL systems, the LM designer reuse the general queries (q_1, q_2, q_3) composing the gaming pattern P_g. The reuse of SPARQL queries is possible by adapting $Q_p = (q_1, q_2, q_3)$ to the specific models elements according to adaptation function 1. The adaptation function generates a new ordered set of SPARQL queries described elements of trace model and learner profile model of a specific TEL system.

4.3 Adapting Gaming the System Query Pattern

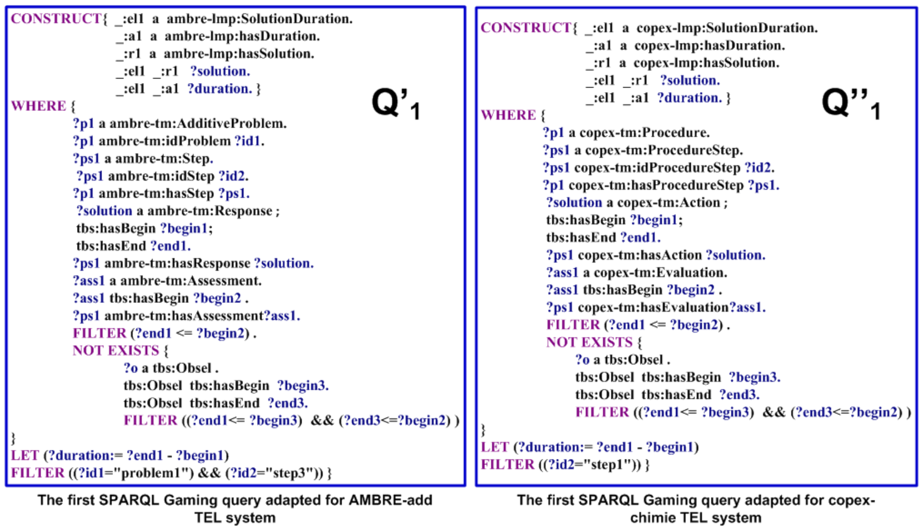
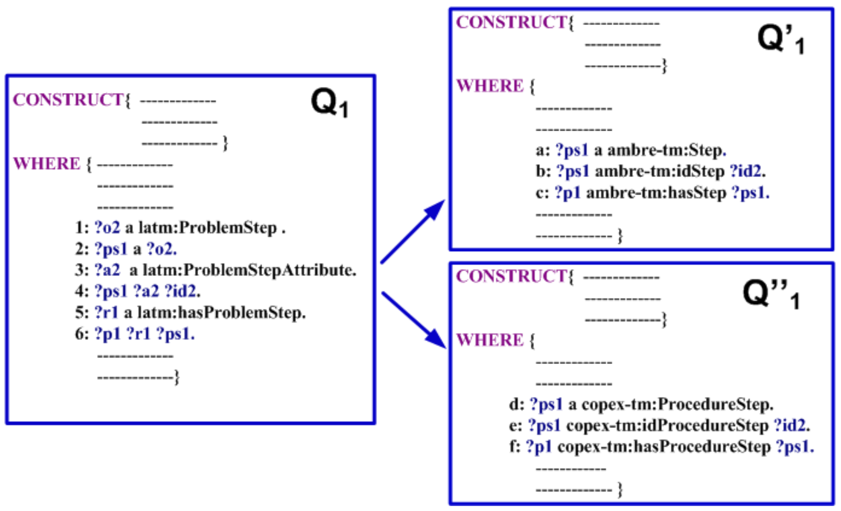

```
CONSTRUCT{ _:el1  a  ambre-lmp:SolutionDuration.
           _:a1  a  ambre-lmp:hasDuration.
           _:r1  a  ambre-lmp:hasSolution.
           _:el1  _:r1   ?solution.
           _:el1  _:a1  ?duration. }
WHERE {
        ?p1 a ambre-tm:AdditiveProblem.
        ?p1 ambre-tm:idProblem ?id1.
        ?ps1 a ambre-tm:Step.
         ?ps1 ambre-tm:idStep ?id2.
        ?p1 ambre-tm:hasStep ?ps1.
         ?solution a ambre-tm:Response ;
         tbs:hasBegin ?begin1;
         tbs:hasEnd ?end1.
        ?ps1 ambre-tm:hasResponse ?solution.
        ?ass1 a ambre-tm:Assessment.
        ?ass1 tbs:hasBegin ?begin2 .
        ?ps1 ambre-tm:hasAssessment?ass1.
        FILTER (?end1 <= ?begin2) .
        NOT EXISTS {
                ?o a tbs:Obsel .
                tbs:Obsel  tbs:hasBegin  ?begin3.
                tbs:Obsel  tbs:hasEnd  ?end3.
                FILTER ((?end1<= ?begin3)  && (?end3<=?begin2) )
}
LET (?duration:= ?end1 - ?begin1)
FILTER ((?id1="problem1") && (?id2="step3")) }
```
Q'1
The first SPARQL Gaming query adapted for AMBRE-add TEL system
```
CONSTRUCT{ _:el1  a  copex-lmp:SolutionDuration.
           _:a1  a  copex-lmp:hasDuration.
           _:r1  a  copex-lmp:hasSolution.
           _:el1  _:r1   ?solution.
           _:el1  _:a1  ?duration. }
WHERE {
        ?p1 a copex-tm:Procedure.
        ?ps1 a copex-tm:ProcedureStep.
        ?ps1 copex-tm:idProcedureStep ?id2.
        ?p1 copex-tm:hasProcedureStep ?ps1.
         ?solution a copex-tm:Action ;
         tbs:hasBegin ?begin1;
         tbs:hasEnd ?end1.
         ?ps1 copex-tm:hasAction ?solution.
         ?ass1 a copex-tm:Evaluation.
         ?ass1 tbs:hasBegin ?begin2 .
         ?ps1 copex-tm:hasEvaluation?ass1.
         FILTER (?end1 <= ?begin2) .
         NOT EXISTS {
              ?o a tbs:Obsel .
              tbs:Obsel  tbs:hasBegin  ?begin3.
              tbs:Obsel  tbs:hasEnd  ?end3.
              FILTER ((?end1<= ?begin3)  && (?end3<=?begin2) )
}
LET (?duration:= ?end1 - ?begin1)
FILTER ((?id2="step1")) }
```
Q''1
The first SPARQL Gaming query adapted for copex-chimie TEL system

Fig. 4. Adaptation of the general SPARQL gaming query q_1 to two TEL systems

To demonstrate the concept of query pattern, we describe the results of application of δ over q_1 (figure 3) for AMBRE-add and copex-chimie TEL systems. The result (figure 4) represents two SPARQL queries q'_1 and q''_1 ready to be used with the two systems.

In order to explain how δ adapts q_1 to AMBRE-add and copex-chimie TEL systems, we need first to remind the general idea behind the process defined by an query pattern. In our framework, the modeling process of learner's interactions is based on three levels of abstraction.The first level (general level) concerns the meta-model $LATM$ used in order to describe traces models for different TEL systems. The second level (specific level) concerns $LATM_{TEL}$ which is an instance of the meta-model $LATM$ describing learner's interactions with the specific TEL system. The last level concerns the *modelled trace* which is the sequence of observed elements tracked within the specific TEL system and associated to $LATM_{TEL}$ elements.



```
CONSTRUCT{ ------------
           ------------
           ------------ }
WHERE { ------------
        ------------
        ------------
     1: ?o2 a latm:ProblemStep .
     2: ?ps1 a ?o2.
     3: ?a2  a latm:ProblemStepAttribute.
     4: ?ps1 ?a2 ?id2.
     5: ?r1 a latm:hasProblemStep.
     6: ?p1 ?r1 ?ps1.
        ------------
        ------------}
```

```
CONSTRUCT{ ------------
           ------------
           ------------}
WHERE {
        ------------
        ------------
     a: ?ps1 a ambre-tm:Step.
     b: ?ps1 ambre-tm:idStep ?id2.
     c: ?p1 ambre-tm:hasStep ?ps1.
        ------------
        ------------ }
```

```
CONSTRUCT{ ------------
           ------------
           ------------}
WHERE {
        ------------
        ------------
     d: ?ps1 a copex-tm:ProcedureStep.
     e: ?ps1 copex-tm:idProcedureStep ?id2.
     f: ?p1 copex-tm:hasProcedureStep ?ps1.
        -----------
        ------------ }
```

Fig. 5. Adaptation of the first gaming query pattern to AMBRE-add and copex-chimie

The assumption around the notion of query pattern consists in applying the process of querying to both general and specific levels. In fact, the query pattern is based on two levels of specification used in the description of SPARQL queries. In the example showed above (figure 5), the triple pattern `?o2 a latm:ProblemStep` from q_1 (line 1) enables to match from $LATM$ the observed elements of type `ProblemStep`. The following pattern `?ps1 a ?o2` (line 2) enables to match the elements from $LATM_{TEL}$ which are instance of the observed element type `ProblemStep`. During the adaptation, those two patterns are no longer needed and only the level of $LATM_{TEL}$ is required. This mechanism is at the base of adaptation algorithm, which enables to prune the level of $LATM$ (the general level) and keep only the level of $LATM_{TEL}$ (the specific one).

In fact, where the general pattern compute and infer informations based on all instances matched by $LATM$, the adapted pattern compute only elements specified by LM designer. In this context, the role of LM designer consists in defining the parameters of the algorithm by specifying the elements involved by the specific elements which substitute the general ones. In this example, the adaptation of q_1 to AMBRE-add and copex-chimie TEL systems takes form by construct new queries q_1' and q_1''. To obtain each one, δ prunes line 1 from q_1 and adapts line 2 by substituting `?o2` respectively by `ambre-tm:Step` and `copex-tm:ProcedureStep` which are the parameters defined by AMBRE-add and copex-chimie LM designers. Indeed, we obtain line a for q_1' and line d for q_1''. With the same mechanism of adaptation, δ supstitutes the general attribute `latm:ProblemStepAttribute` from $LATM$ (lines 3-4) by the spesifics attributes `ambre-tm:idStep` in line b and `copex-tm:idProcedureStep` in line e from AMBRE-add and copex-chimie TEL systems. Also, δ supstitutes the general relation type `latm:hasProblemStep` from $LATM$ (lines (5–) by the spesifics relations `ambre-tm:hasStep` in line c and `copex-tm:hasProcedureStep` in line f from AMBRE-add and copex-chimie TEL systems.

5 Related Work

The work presented in this article is closely related to research in several areas. We review them briefly. Recent research works offer services to describe the way to calculate behavioral patterns from interaction traces in order to identify the learning style of the students [12,3]. These patterns are calculated from a specific educational hypermedia system and it is difficult to reuse them to an other TEL system. In area of Learner/user modelling, many works have address the issue of reusability in the context of Ontology-Based LM (e.g. [1,16,9,7]) however without dealing with diagnosis support and reuse. Even if they use semantic web ontology languages as RDF and OWL which offer many advantages (formal semantics, easy reuse, easy portability, availability of effective design tools, etc.) such frameworks lack to support LM designer task specially to offer him the reusing services.

6 Conclusion and future work

In This paper we presented a framework to describe Learner Modelling (LM) process based on interactions traces. This framework includes an RDF-Based representation of knowledge models that can be used by a LM designer. The first model enables the LM designer to describe observations about learner's interactions with a TEL-system. The second model enables the LM designer to describe the structure of learner's profile. This framework supports also the description of reusable and adaptable SPARQL-based query patterns. These patterns enables the LM designer to calculate and infer learner profile elements for different TEL systems . We defined formally the notion of query pattern and illustrate its application in the context of two TEL systems in order to detect gaming the system. In the future, we aim to define more query patterns and to experiment them for other TEL systems.

References

1. Aroyo, L., Dimitrova, V., Kay, J. (eds.): PerSWeb-2005 Workshop. Personalisation on the Semantic Web at 10th International Conference User Modelling 2005 (2005)
2. Baker, R.S.J.D., Corbett, A.T., Koedinger, K.R., Evenson, S., Roll, I., Wagner, A.Z., Naim, M., Raspat, J., Baker, D.J., Beck, J.E.: Adapting to when students game an intelligent tutoring system. In: Proceedings of the 8th International Conference on Intelligent Tutoring Systems. pp. 392–401. Springer-Verlag (2006)
3. Bousbia, N., Labat, J.M., Rebai, I., Balla, A.: Indicators for deducting the learners' learning styles: Case of the navigation typology indicator. pp. 385–389 (2009)
4. d'Ham, C., Girault, I.: Scaffolding the students' activity of experimental design with a dedicated software: copex-chimie. In: Proceeding of ESERA 2009 (2009)
5. Dillenbourg, P., Self, J.: A framework for learner modelling. Interactive Learning Environments 2(2), 111–137 (1992)
6. Dolog, P.: Engineering Adaptive Web Applications. Ph.D. thesis, Doctoral Thesis, Leibniz University of Hannover, Hannover, Germany (March 2006)
7. Kay, J., Lum, A.: Ontologies for scrutable student modelling in adaptive elearning. In: Proceedings of the Adaptive Hypermedia and Adaptive Web-Based Systems Workshop on Semantic Web for E-Learning (2004)
8. Kobsa, A.: Generic user modeling systems. User Modeling and User-Adapted Interaction 11, 49–63 (March 2001)
9. Munoz, L.S., Palazzo, J., Oliveira, M.: Applying semantic web technologies to achieve personalization and reuse of content in educational adaptive hypermedia systems. In: Proceedings of the SWEL Workshop at Adaptive Hypermedia. pp. 348–353 (2004)
10. Nogry, S., Jean-Daubias, S., Guin, N.: Its evaluation in classroom: The case of ambre-awp. In: Intelligent Tutoring Systems. pp. 511–520 (2004)
11. Paiva, A., Self, J.: Tagus a user and learner modeling workbench. User Modeling and User-Adapted Interaction 4, 197–226 (1995), 10.1007/BF01100244
12. Popescu, E.: Learning styles and behavioral differences in web-based learning settings. Advanced Learning Technologies, IEEE International Conference on pp. 446–450 (2009)
13. Settouti, L., Guin, N., Mille, A., Luengo, V.: A trace-based learner modelling framework for technology-enhanced learning systems. IEEE International Conference on Advanced Learning Technologies pp. 73–77 (2010)
14. Settouti, L.S., Prie, Y., Marty, J.C., Mille, A.: A trace-based system for technology-enhanced learning systems personalisation. In: IEEE International Conference on Advanced Learning Technologies. pp. 93–97 (2009)
15. Sison, R., Shimura, M.: Student modeling and machine learning. IJAIED International Journal of Artificial Intelligence in Education 9, 128–158 (1998)
16. Winter, M., Brooks, C., Greer, J.: Towards best practices for semantic web student modelling. In: Proceeding of the 2005 conference on Artificial Intelligence in Education. pp. 694–701 (2005)