Specification of Assessment-test Criteria through ASP Specifications

Giovambattista Ianni, Claudio Panetta, and Francesco Ricca

Departments of Mathematics, Università della Calabria

Abstract. The EXAM system is a complete on-line exam taking portal. Teachers and students are assisted in the whole process of assessment test building, exam taking, and test correction. One of the most interesting features of the portal is the possibility to automatically generate assessment tests based on user defined constraints. A teacher is made able to build up an assessment test template using a simple web-based interface, and then her preferences are translated into a logic executable specification. In this paper, we describe the assessment test generation engine of the EXAM portal, then we look at the internal and formal specification of the problem. A brief system overview is given and, eventually, we discuss about related works.[1]

1 Introduction

Online exam taking systems have recently seen a big proliferation on the web [6]. Such systems (see e.g. [7, 9, 13, 16, 20–23]) meet the needs of both industries and teaching institutions to assess the learning level of, respectively, employees and students. Although this aspect of the e-learning field proves to be of great interest for the market, several related issues deserve more in-depth study. Explicit or implicit assessment of the learner plays a very important role when the learner profile has to be determined and when a specific user learning programme has to be (semi) automatically generated [3], [19], [14]. The above cited commercial systems allow to author a full assessment test through manual selection of single questions from a given database. This task turns out to be very annoying, especially if the (usually huge) given question database cannot be searched in a structured way.

Furthermore, when assessment tests are taken in classrooms, it is often necessary to propose a different test to each learner, keeping anyway the same difficulty level. This avoids mutual collaboration between students. For instance, the Gradiance system [20] allows to generate set of tests containing slight variations (e.g. in the order of possible answers for each question).

Answer Set Programming is a good candidate technology for approaching the above mentioned issues. The adoption of an ASP solver engine may allow to quickly specify in a formal way the combinatorial problem underlying the system; also, the obtained

[1] This work has been partially funded by the EU research projects IST-2002-33570 (INFOMIX), and IST-2001-37004 (WASP), and by FWF (Austrian Science Funds) under the project "Answer Set Programming for the Semantic Web" (P17212-N04).

specifications are easy to be configured and managed, whereas computation times of current solvers can be kept reasonable.

Indeed, in recent years, the answer set programming (ASP) paradigm has emerged as an important tool for declarative knowledge representation and reasoning. This approach is rooted in semantic notions and is based on methods to compute logical models.

One of the main reasons for the increasing popularity of both the answer set semantics as well as the stable model semantics is in large part due to the availability of sophisticated solvers for these languages. On the one hand, the system DLV [17] is a state-of-the-art implementation for the answer set semantics, and, on the other hand, the Smodels system [18] implements the stable model semantics.

Furthermore, in view of its inherent expressibility, the answer set semantics is a suitable tool to serve as a host language for capturing specialized advanced reasoning tasks. Consequently, in accordance to the general methodology of the answer set programming paradigm, ASP solvers can be used as underlying reasoning engines for evaluating such dedicated tasks. Different such tasks have been implemented, e.g., on top of the DLV system. In particular, DLV provides front-ends for planning [11] and diagnostic reasoning [10], as well as computing the semantics of updates of nonmonotonic knowledge bases represented as logic programs [12], or the semantics of inheritance programs [4]. Furthermore, the plp front-end [8] to DLV allows the computation of different preference approaches under the answer set semantics, and extensions of the core DLV syntax allow the natural formalization of optimization problems, in terms of the so-called weak constraints [5]. A similar flexibility and applicability applies to the Smodels system as well, which can also be used as a C++-library called from user programs, or as a stand-alone program together with suitable front-ends. The increasing interest in ASP is also documented by the recent establishment of the Working Group on Answer Set Programming (WASP) which is supported by the European Commission (IST-2001-37004).

Also, the DLV system has already been exploited in [1], where we proposed a system capable of building, in a semi-automatic way, personalized learning paths.

In this paper we introduce a more sophisticated approach in order to address the task of building an assessment test. The user of the EXAM portal is made able to generate groups of assessment tests through the design of a test template. An automatic generation engine is then run; such an engine computes and outputs tests compliant with the input template. This job is performed extracting suitable items from a given question database.

In order to achieve this goal, the internal system features several, significant characteristics: first of all, each question put in the system database is annotated with semantic information extending the IMS-QTI specification for question items [15]. Second, teacher desiderata can be specified through a visual interface and then converted in a formal specification. Such formal, logic, specification is executable through the DLV system [17]. The output of such an execution is a set of candidate assessment tests, built through the extraction of items from the question database. The generated assessments are IMS-QTI standard and fully interchangeable.

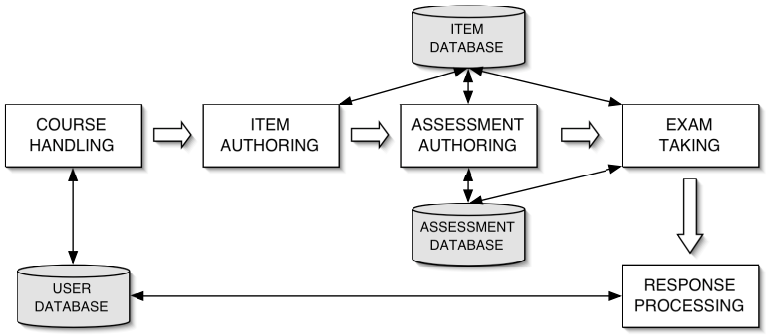

Fig. 1. EXAM Architecture.

The structure of the paper is as follows: we give a system overview in the next section; then in Section 3 it is discussed the relationship between the EXAM system and the IMS-QTI standard; Section 4 discusses about the internal implementation of the Assessment Generation engine, whereas Section 5 describes the Assessment Authoring interface. Eventually we briefly discuss about related work in Section 6 and we draw conclusions in Section 7.

2 System Overview

The EXAM portal is a system conceived in order to ease the student assessment procedure, in all the steps of the corresponding life-cycle, such as question and test editing, exam taking, and outcome evaluation. EXAM adopts the IMS-QTI [15] standard as internal information storage format.

The overall architecture of EXAM is shown in Figure 1. The system is constituted of five different modules:

- Course Handling Module;
- Item Authoring Module;
- Assessment Authoring Module;
- Exam Taking Module;
- Response Processing Module;

The Course Handling Module makes automatic the practical organization of an assessment session. It allows to create a new course, to indict a new assessment session for a given course, and to manage student booking and grouping;

The module is conceived for handling large amounts of students. Exams can be taken either in classrooms or from the web; in the former case the system features automatic test assignment, automatic grouping and session timetabling.

The Item Authoring Module make the teacher able to manage a question database (a question is called “item” in the IMS-QTI specification). Items are built by using a simple graphical interface. The interface supports the most common IMS-QTI item

types such as *True/False, Multiple choice, Fill-in-Blank, Multiple Fill-in-Blank, Multiple choice with Images, Standard with Text-Area, Order with images*.

Moreover, the module implements a non standard kind of item named "Question with File". Teachers can embed inside this kind of item a generic file and the student can upload her answer by using a generic file as well. The Item Authoring Module also implements an item versioning system, allowing teachers to update/rearrange questions, and a simple access policy conceived in order to share items among teachers.

The Assessment Authoring module is the EXAM core module. This module has a graphical interface that allows a teacher to specify an assessment schema. An assessment schema is an assessment specification (i.e. an assessment test template), containing constraints about the quality and quantity of items to be put in a candidate test. Given an assessment schema, the Assessment Builder Module generates an executable specification written in the DLV language [17]. The execution of such a specification outputs a set of assessment tests fulfilling the user desiderata. A teacher can also build, partially or entirely, assessment tests without taking advantage of the automatic assessment building procedure. In this case the teacher manually selects items by herself. Assessments built by the Assessment Authoring module are fully IMS-QTI compliant. Moreover, the Assessment Authoring module implements a versioning system which allows a teacher to update/rearrange previously defined assessment schemas.

The Exam Taking module allows to handle one or more assessment sessions. Teachers can enable students or groups of students to start their assessment session and answer questions.

A set of policy restrictions can be customized in order to prevent unauthorized access and establish the author identity: (IP filtering) A student can start an assessment session only if its machine is registered in the system; (IP binding) After starting an assessment session a student cannot change its machine without teacher consensus; (Filtering) Teachers can enable/disable student connection to the system.

The last module (the Response Processing module), allows to evaluate and to statistically analyze "students responses". The Response Processing module performs, whenever it is possible, an automatic scoring of student answers and it allows, through the user interface, to check and assign manually the remaining students scores. The system can automatically process almost every standard kind of item. For instance Standard True-False or Multiple-Choice items are scored without human intervention, but Fill-in-Blank or "With file" items need explicit teacher supervision. The Response Processing module also helps the teacher to modify the overall scores, either individually or globally, and to perform statistical analysis. Users (administrator, teachers and students) connect to the system by only using a web-browser. No platform-specific software is needed.

User requests are handled by a web application that manipulates relational data through JDBC, whereas XML data are processed through DOM and XSLT. In particular, DOM is used to create and modify IMS-QTI objects and XSLT is used to obtain a corresponding HTML graphic representation. The DLV system is called to perform reasoning during the assessment generation stage. Data about users (User Database in Figure 1) and assessment schemas are stored in standard relational database format.

Data about assessments and questions (Item and Assessment Database in Figure 1) are stored in a hybrid way, combining XML files and relational database tables. In particular, XML files (IMS-QTI compliant) representing items are stored as blob attributes in relational database tables. Such tables also store some extra information about items in relational database format in order to improve system efficiency. Data about assessments is stored in a standard relational database format which can be easily translated in XML format (IMS-QTI compliant). User files and extra information are stored directly in the file system.

3 Exam and the IMS-QTI Specification

The Instructional Management System Question and Test Interoperability (IMS-QTI) specification describes a basic structure for the representation of question (item) and test (assessment) data and their corresponding results reports [15].

Moreover, the specification allows representation (and sharing) of this resources inside (and between) Learning Management Systems, authors and libraries.

IMS adopted the XML language [24], the industry standard markup language used to encode data models and store data, in order to promote the widest adoption of the the QTI specification.

Like all IMS specifications, the IMS-QTI specification provide no specification for user interfaces or pedagogical paradigms. This way the IMS-QTI specification can be easily supported or integrated in commercial Learning Management System or simple e-learning applications.

The technical structure of the IMS-QTI specification is based upon two components:

- The ASI components (describe evaluation objects: Assessment, Section, Item);
- The results reporting objects (contain the results to be reported for an evaluation).

The QTI ASI specification formally define and details in terms of their elements, sub-elements and attributes the important concepts of Assessment, Section and Item.

The Item entity is defined in order to represent question types (e.g. multiple choice, fill in the blank etc.). The Item is the smallest independent unit that can be exchanged using IMS-QTI. Consequently, an Item cannot be composed of other Items. An Assessment is an instance of a test that can be assembled from blocks of Items, also defined Sections. A Section can contain any combination of Sections and/or Items, and one or more Sections can be contained within a QTI-XML instance. Sections are used to support grouping constructs defined by an educational paradigm and/or control the ways in which different sequences of items may be constructed. The section is merely a grouping construct and what a section means in an assessment depends on the author intentions. As previously pointed out, an assessment is composed of one or more Sections which themselves are composed of Items, or of more Sections. Assessment, Section and Item data objects are referred to as ASI (Assessment, Section, Item) structures.

The QTI ASI structures instances are stored by using XML documents. The QTI ASI Information Model document describes the XML metadata that can be used to catalog Items, Sections, and Assessments. A description of such metadata, rich of implementation details, is outside the scope of this section and is omitted. The interested reader can refer to [15].

4 Assessment automatic generation internals

Following the ASI model, an assessment schema is a template for an ASI structure. It is constituted of a set of template sections (possibly nested) and template items. Template sections (resp. template items) are associated to a set of constraints; each constraint specifies some requirement an actual section (resp. item) should have in order to fulfill the schema prerequisites. Constraints may be either strong or weak. In order to be accepted, an ASI instance must respect all the specified strong constraints. Weak constraints express only weak preferences and are satisfied only if possible. A quality order is anyway established between ASI instances (assessments having a greater number of satisfied weak constraints are preferred with respect to instances having more violated weak constraints).

In particular, it is possible to select items in terms of: topic; type; minimum and maximum duration; minimum and maximum difficulty level; version and author of the item; age (date a question was last used in a different assessment by the same teacher). Moreover, a template section can be constrained in terms of: overall minimum and maximum time for completion of the section; overall minimum and maximum score of the section.

As an example, imagine we want to design an assessment schema containing two template sections A and B. The section A might contain two strong constraints prescribing *i)* that A must contain five items; *ii)* that the topic of each item in A must be "Computer Science";

In the second template section, we may specify a strong constraint about the number of items (e.g. we might require at least three items to be put in B) and, about the topic (e.g. we might need to have items about "Computer Networks" only). Furthermore, we might introduce a weak constraint telling that the difficulty level of the items inside B *should* be above 3 out of a maximum level of 5.

After the automatic generation engine is triggered the assessment schema at hand is converted in an executable specification written in the DLV language.

4.1 Problem specification in ASP

In formal terms, the assessment building generation task is a combinatorial problem that can be easily proven to be NP-hard. Roughly speaking, it is given an assessment schema coupled with a database of questions and it is asked to find some assessment matching the given assessment schema.

As previously pointed out, in order to specify the assessment building problem, we used the DLV language, taking advantage of the ASP semantics enriched with weak constraints and aggregate functions.

The database of questions has been modeled through the logic predicate `item(co,ar,ty,te,du,sc,ve)`, where *co* is the item code, *ar* is the topic of the item, *ty* is its type, *te* the item question, *du* the item duration, *sc* the item score and *ve* the item version.

The assessment schema is modeled by using a set of logic predicates. In particular, let *AS* be an assessment schema, we assert:

- the fact `section(as)` to encode the assessment;
- a fact `section(s)` for each section *s* in *AS*;
- a fact `contains(i,j)` for each couple of sections *i* and *j* s.t. *i* contains *j* in *AS*;
- a fact `manuallySelectedItem(s,i)` for each manually selected item *i* contained in section *s*.

The search space is specified by the following rules:

```
inSection(S,I) v notinSection(S,I) :- section(S), S<>as,
       item(I,_,_,_,_,_,_), not manuallySelectedItem(S,I).
inSection(S,I) :- manuallySelectedItem(S,I).
```

A strong constraint selects solutions which contain the same item only once:

```
:- inSection(S1,I), inSection(S2,I), S1!=S2.
```

Section properties are encoded by using the following auxiliary predicates:

```
enclosedIn(S,I) :- inSection(S,I).
enclosedIn(S,I) :- inSection(S1,I), containsClosure(S,S1).
containsClosure(S,S1) :- contains(S,S1).
containsClosure(S,S2) :- contains(S,S1),
       containsClosure(S1,S2).
```

where `containsClosure(s,i)` computes all the items directly and indirectly enclosed in a given section.

Finally, for each different kind of section property, a suitable couple of constraints (one weak and one strong) is specified. As an example, the encoding of a constraint on the minimum number of items in a given section is reported below;

```
:-#count{I: enclosedIn(S,I)}<NumMin,
       minNumberItemsSection(S,NumMin,strong).
:~#count{I: enclosedIn(S,I)}<NumMin,
       minNumberItemsSection(S,NumMin,weak).
```

Notice that the specified constraints are the strong and the weak version of the same property, and only one will be activated, depending on the user choice. Any other section property can be encoded in a similar way.

It is worth pointing out that, although the formal specification of the program is given in terms of a non-ground program, such a program is instance independent. The grounded version of the program is polynomially larger only with respect to the originating program. Furthermore in the presence of a large question databases, usually only a small portion of data is selected and taken into account.

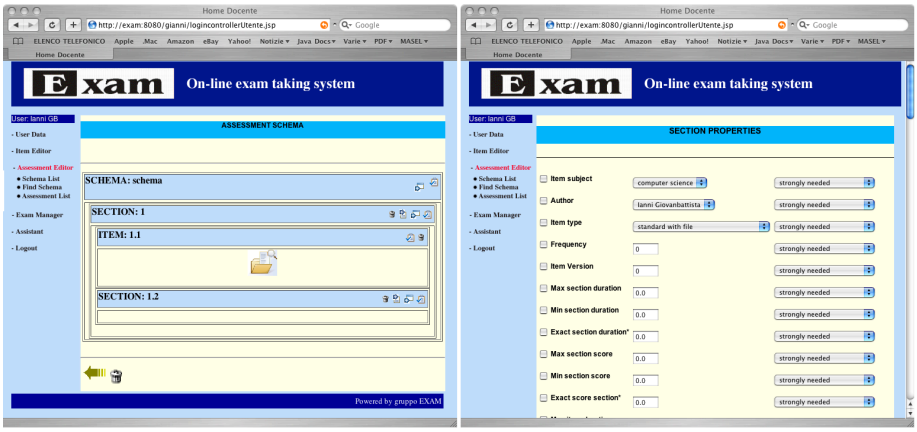

Fig. 2. The Assessment editing interface and the property editing page.

5 A web based interface to build assessments

We describe in this section the EXAM Assessment Authoring interface.

Assessment schemes can be built either manually or automatically by using a common interface.

Some screen-shots of the assessment schema editing tool are shown in Figure 2. New sections and items can be added or removed to/from the schema, by operating on the visual interface. Each section template has a property form, from which it is possible to operate on the list of the available constraints.

This way the teacher can deal with the large number of options available, in order to control the assessment creation process (see Figure 2).

It is important to note that the automatic generation process can be partially overridden by choosing manually from the item database. This feature can be very useful if a teacher definitely want to put a specific item in all generated assessments complying with the schema.

Assessments created by using this interface are stored in the assessment data base in order to be used by the exam taking module.

Once the assessment schema editing stage is complete, it is possible to activate the automatic generation engine.

6 Related Works

At the moment of writing, there are several electronic assessment systems available on the market, such as:

Knowledge Presenter (Deakin KM) [16], Exam Engine (Platte Canyon Multimedia Corp) [7], Rapid Exam (XStream Software) [21], Testcraft (Ingenious Group) [22], Lumenix ExamBuilder (ExamBuilder) [13], ForceTen EEDO Knowledgeware (EEDO) [9], TRA (automatic e-Learning) [23] Gradiance (Gradiance Corporation)[20].

All of them make the author able to edit new items (questions), to put them in a repository; They support the author during the item selection phase by search tools and classification tools; but no automatic assessment generation tool is supported.

In the literature there are examples of systems allowing a sort of automated assessment generation by using a random selection and item shuffling features. This kind of automatization allows to obtain several equivalent assessments producing a permutation of the given set of questions. This effort allows to decrease the possibility of cheating but does not offer support to the author during the assessment authoring phase. A better support can be given to the assessment building phase by specialized tools that can automatically build questions and assessments for a given (well supported) subject, like Mathematics [2]. But this kind of specialization limit the use of this techniques in a general context. Objective tests and quizzes are among the most widely used and well-developed tools in higher education; for a most comprehensive survey (updated to 1999) please refer to [3].

7 Conclusive Remarks

At the moment of writing, the EXAM system is undergoing a thorough extensive beta testing by teachers and students of the Computer Science Curriculum at Università della Calabria. Experimental results will be published in an extended version of this paper.

References

1. F. Ricca A. Garro, N. Leone. Logic-Based Agents for E-Learning. In *Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems, IJCAI 2003 (Eighteenth International Joint Conference on Artificail Intelligence), Acapulco Mexico, pp. 36-45*, 2003.
2. Franco Bagnoli, Fabio Franci, Francesco Mugelli, and Adrea Sterbini. WebTeach in Practice: The Entrance Test to the Engineering Faculty in Florence. In *Proceedings of the IASTED International Conference WEB-BASED Education (WBE2004) February 16-18, Austria* , pages 274–275. ACTA Press, 2004.
3. Peter Brusilovsky and Philip Miller. Web-based Testing for Distance Education. In *Proceedings of WebNet 99 - World Conference on the WWW and Internet, Honolulu, Hawaii, USA, October 24-30*, pages 149–155. in P. De Bra, John J. Leggett (Eds.), 1999.
4. Francesco Buccafurri, Wolfgang Faber, and Nicola Leone. Disjunctive Logic Programs with Inheritance. *Journal of the Theory and Practice of Logic Programming*, 2(3), May 2002.
5. Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Enhancing Disjunctive Datalog by Constraints. *IEEE Transactions on Knowledge and Data Engineering*, 12(5):845–860, 2000.
6. Training Web Conference. Atlanta usa, 2004.
7. Platte Canyon Multimedia Corp. Exam Engine. `<URL:http://www.vbtrain.net/examengine.aspx>`.
8. Jim Delgrande, Torsten Schaub, and Hans Tompits. plp: A Generic Compiler for Ordered Logic Programs. In Thomas Eiter, Wolfgang Faber, and Mirosław Truszczyński, editors, *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-01)*, number 2173 in LNCS, pages 411–415. Springer, 2001.
9. EEDO. Forceten. `<URL:http://www.eedo.com/english/products/forceten.html>`.

10. Thomas Eiter, Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. The Diagnosis Frontend of the `dlv` System. *AI Communications – The European Journal on Artificial Intelligence*, 12(1–2):99–111, 1999.
11. Thomas Eiter, Wolfgang Faber, Nicola Leone, Gerald Pfeifer, and Axel Polleres. System Description: The DLV$^{\mathcal{K}}$ Planning System. In Thomas Eiter, Wolfgang Faber, and Mirosław Truszczyński, editors, *Logic Programming and Nonmonotonic Reasoning — 6th International Conference, LPNMR'01, Vienna, Austria, September 2001, Proceedings*, number 2173 in Lecture Notes in AI (LNAI), pages 413–416. Springer Verlag, September 2001.
12. Thomas Eiter, Michael Fink, Giuliana Sabbatini, and Hans Tompits. An Update Front-End for Extended Logic Programs. In Thomas Eiter, Wolfgang Faber, and Mirosław Truszczyński, editors, *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-01)*, number 2173 in LNCS, pages 397–401. Springer, 2001. System Description (abstract).
13. ExamBuilder. Exam Builder e-learning solutions for professionals. <URL:http://www.exambuilder.com>.
14. William Fone. A Topology and Framework to Aid the Design of Automated Assessment. In *Proceedings of the The 3rd IEEE International Conference on Advanced Learning Technologies (ICALT-03)*, pages 274–275. IEEE Computer Society, 2003.
15. IMS. IMS Question and Test Interoperability v2.0 Public Draft. - IMS Global Learning Consortium, inc., 2003. <URL:http://www.imsglobal.org/question/index.cfm>.
16. Deakin KM. Knowledge presenter. <URL:http://www.deakinkm.com>.
17. Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Christoph Koch, Cristinel Mateis, Simona Perri, and Francesco Scarcello. The DLV System for Knowledge Representation and Reasoning. Technical Report INFSYS RR-1843-02-14, Institut für Informationssysteme, Technische Universität Wien, A-1040 Vienna, Austria, October 2002.
18. Ilkka Niemelä, Patrik Simons, and Tommi Syrjänen. Smodels: A System for Answer Set Programming. In Chitta Baral and Mirosław Truszczyński, editors, *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning (NMR'2000)*, Breckenridge, Colorado, USA, April 2000.
19. Leonid Pesin. Knowledge Testing and Evaluation in the Integrated Web-Based Authoring and Learning Environment. In *Proceedings of the The 3rd IEEE International Conference on Advanced Learning Technologies (ICALT-03)*, pages 268–269. IEEE Computer Society, 2003.
20. Gradiance Education-Support Services. The Gradiance system. <URL:http://gradiance.com>.
21. XSTream Software. Rapid exam. <http://www.xstreamsoftware.com>.
22. TestCraft. TestCraft. - Testcraft Assessment Software. <URL:http://http://testcraft.com>.
23. TRA. TRA E-Learning. <URL:http://www.tra.com>.
24. W3C. The xml markup language specification. <URL:http://www.w3.org/XML>.