A QTI Metamodel

Sonja Radenković, Nenad Krdžavac, Vladan Devedžić

FON - School of Business Administration,
Department of Information Systems and Technologies,
University of Belgrade, Serbia
sonjam@fon.bg.ac.yu, devedzic@etf.bg.ac.yu,
nenadkr@galeb.etf.bg.ac.yu

Abstract. Question & Test Interoperability specification (QTI) provides a good starting point for modeling and designing assessment systems. This paper presents a metamodel for developing a QTI-based assessment system using the Model Driven Architecture (MDA) standard. The main reason for applying the MDA standard in development of assessment systems is to make a clear difference between conceptual and concrete modeling in order to automate transfer and sharing of information and knowledge. The first prerequisite for this is a QTI metamodel.

Keywords: Assessment, Metamodeling, Model Driven Architecture

1. Introduction

In order to create a flexible, interoperable assessment system that can be easy to maintain and reuse, the Model Driven Architecture (MDA) standard [7] that comes from software engineering can be applied. By applying this standard, it is possible to utilize the possibilities of traditional assessment, and at the same time to combine additional demands of new types of assessment with limited resources. The core concept here is to change the system's specification rather than implementation using the Unified Modeling Language [9] as the standardized modeling language that most tools provide support for.

This paper proposes a metamodel for development of assessment systems based on the IMS Question & Test Interoperability specification [4] (or just QTI systems, for short) using the MDA standard.

2. IMS QTI Standard – A Short Overview

The QTI standard [4] specifies how to represent question (assessmentItem) and test (assessmentTest) data and the corresponding result reports. These items are the smallest exchangeable assessment object within this specification [5]. An item is more than a 'question' in that it contains the question and instructions of how to be presented, the response processing to be applied to the candidate response(s), and the

feedback that may be presented (including hints and solutions). There are different forms of items, such as simple- or multiple-choice questions or fill-in-the-blank tasks [6]. The presentation provides the structure for defining several possibilities for the same question, in order to be able to present the same question in different ways. Each answer within this question can also have different wordings. The response processing allows the author of a test to predefine how the answers to the test will be evaluated. The results of a test can be expressed using the "result reporting". This definition describes that the results of a test can be recorded so that other systems can make use of it. The feedback component of QTI consists of two types, modal and integrated [5]. There is an exchange of items, assessment, and results between authoring tools, item banks, learning systems and assessment delivery systems. For interchange between these systems, an XMI binding (see Section 5 for explanation) is provided [10].

3. Related Work

There were efforts to develop an assessment system [8] that will conform to the IMS QTI standard in order to support the reuse of both new and traditional assessment types. Some of them tried to deploy a specific methodology in order to improve reusability of assessment systems [2]. They developed an extensible educational model of assessment to provide a broader basis for interoperability specifications for the assessment process from construction to evaluation. The model was cast in the Unified Modeling Language (UML) [9], a standard language for specifying, visualizing, constructing and documenting concepts and artifacts.

This model, however, cannot provide interoperability of assessment systems at all. This model can be used to create Java classes and method bodies, for example, using Java UML profile [15], and then to transform this UML model into Java code, and to complete the Java program using a Java IDE [16]. On the other hand, it cannot be used to create heterogeneous assessment system that interoperates at the level of standard component interfaces.

4. Problem Statement

To create a flexible assessment system capable of presenting and analyzing the students' solutions, it is necessary to define the system requirements precisely. These requirements are:

- The system has to be designed according to standards of modern software engineering, such as the MDA standard.
- It has to provide a well-documented content format for storing items, in a way independent of the authoring tool used to create them;
- It has to support the deployment of item banks across a wide range of learning and assessment delivery systems;
- It has to support the deployment of items and item banks from diverse sources in a single learning or assessment-delivery system;

- It has to provide other systems with the ability to report test results in an intelligent manner; to this end, a good starting point can be the use of description logics (DL) reasoning techniques, such as concept classification and consistency [14].

By creating a system to accomplish the above requirements, one obtains:
- a flexible and quickly developed assessment environment;
- high-level of interoperability between various component systems;
- easy and intuitive testing of students' knowledge, done in an intelligent way;
- an easy-to-extend system that can be improved by including new subsystems.

5. A QTI Metamodel

In order to provide the interoperability in both homogeneous and heterogeneous QTI assessment systems, it is useful to apply the MDA standard in QTI assessment system development. The reasons are as follows:
- the interoperability in heterogeneous environments could be achieved via shared metadata of subsystems that create the assessment system.
- the overall strategy for sharing and understanding metadata consists of the automated development, publishing, management and interpretation of models [10].

In other words, the metamodel determines how expressive its models can be, because the models are defined using the concepts defined in the metamodel.

The first step in developing a QTI-based assessment system using the MDA standard is to create a metamodel that captures the main concepts of QTI, i.e. those for creating the models of Item Bank Tool, Test Construction Tool, as well as Learning System Tool. The novelty in creating QTI-based assessment systems here is the improvement of response processing. As mentioned above, the QTI standard specifies that the response processing allows the author of the test to predefine how the test answers will be evaluated. Specific rules can be defined, which allow the exact definition of when an answer to a question is wrong and when it is correct. Furthermore, scoring weights can be defined that allow for a more complex evaluation of a test. We are going a step further; we propose applying DL reasoning techniques in response processing of a QTI-based assessment system in order to check semantic consistency of answers to open-ended questions. Because of that, it is necessary to use a DL metamodel as a part of QTI-based assessment system metamodel.

The section below explains the QTI metamodel. The metamodel is designed using the Posedion UML tool (www.gentleware.com), and according to the IMS QTI standard [6]. The metamodel is a UML-based QTI metamodel. The metamodel is divided in four meta-packages, according to the QTI standard: *Item Bank Tool* , *Learning System Tool* , *Test Construction Tool* and *Description Logics Reasoning Tool* . Due to space limitations, only two of them are discussed in this paper.

5.1 Item Bank Tool Metapackage

According to the QTI standard [4], the following metaclasses may be defined: *ResponseDeclaration*, *TemplateDeclaration*, and *OutcomeDeclaration* (Figure 1). These metaclasses are derived from the *VariableDeclaration* metaclass [5]. The attribute called *Identifier* is the same for the above *declaration* metaclasses .

The core metaclass of the Item Bank Tool metapackage is *AssessmentItem* (Figure 1). The other metaclasses are connected to this metaclass. *Identification* of any assessmentItem model, at the `M1 level is defined using the *identifier* attribute in the metaclass.

According to [5], the *ItemBody* metaclass represents the text, graphics, media objects, and interactions that describe the item's content and information about how it is structured. The body is presented by combining it with stylesheet information, either explicitly or implicitly using the default style rules of the delivery or the authoring system [5], (Figure 2). The *ItemBody* metaclass is linked to the *FeedbackBlock* metaclass (Figure 2). The *FeedbackBlock* metaclass is very important in case of presenting any material to the students.
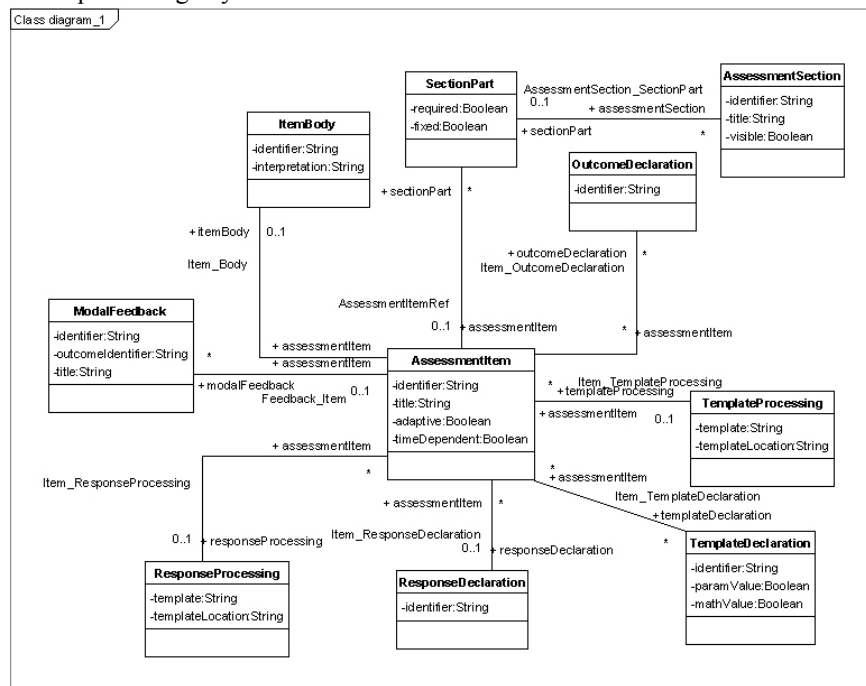

Fig. 1. Item Bank Tool's metaclasses

The metaclasses that are in charge of processing are *OutcomeProcessing*, *ResponseProcessing* and *TemplateProcessing* (Figure 3, Figure 4, Figure 5, and Figure 6). Response processing consists of a sequence of rules that are carried out, in a predefined order, by the response processor [5]. The *ResponseProcessing* metaclass has two attributes: *template* and *templateLocation* (Figure 4, Figure 5) [5].

ResponseProcessing depends on *ResponseConditions* and *ExitResponse* (Figure 4).

Response processing involves the application of a set of *responseRules* including the testing of *responseConditions* and the evaluation of expressions involving the item variables. The *ResponseProcessing* metaclass (Figure 4) represents this involvement. Response processing is the process by which the Delivery Engine assigns outcomes based on the candidate's responses [5]. The OutcomeProcessing metaclass is linked to *OutcomeRules* (Figure 3). The metaclasses *OutcomeCondition* and *ExitTest* extend *OutcomeRule*.

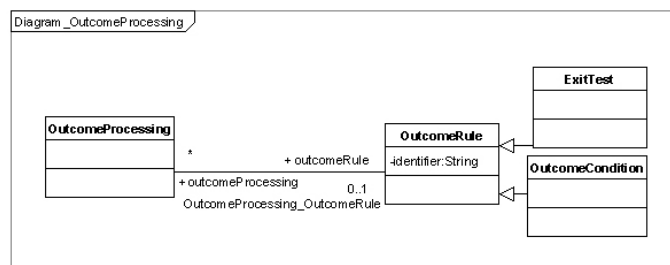**Fig. 2.** The ItemBody metaclass in the QTI Metamodel

Fig. 3. The OutcomeProcessing metaclass in the QTI metamodel

The *feedback* that the QTI system provides to the student (candidate) conditionally includes the material presented. This is based on the result of *responseProcessing* . In other words, the feedback is controlled by the values of outcome variables [5] In the QTI metamodel, feedback is represented with the *FeedBackBlock* metaclass (Figure 2). There is, also, *integrated feedback* , which is embedded in the *itemBody.*

Template processing consists of one or more *templateRules* (Figure 5) that are followed by the cloning engine or delivery system in order to assign values to the template variables [5]. Template rules are described by the *TemplateRule* metaclass. The *SetTemplateValue* and *SetCorrectResponse* metaclasses extend the *TemplateRule* metaclass.
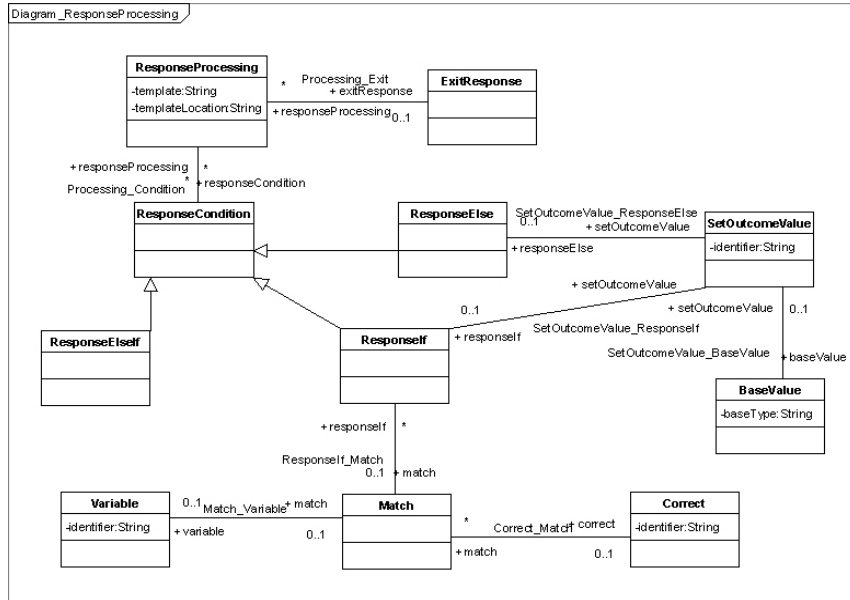
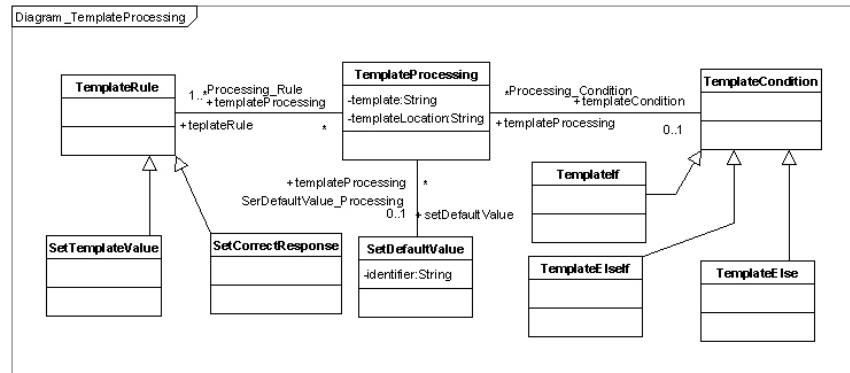**Fig. 4.** The ResponseProcessing metaclass in the QTI metamodel

Fig. 5. The TemplateProcessing metaclass in the QTI metamodel

5.2 Test Construction Tool Metapackage

The central metaclass in the Test Construction Tool metapackage is *AssessmentTest*. This metaclass aggregates the *TestPart* metaclass and the associated *ScoreReport* and *TimeLimits* metaclasses (Figure 6).
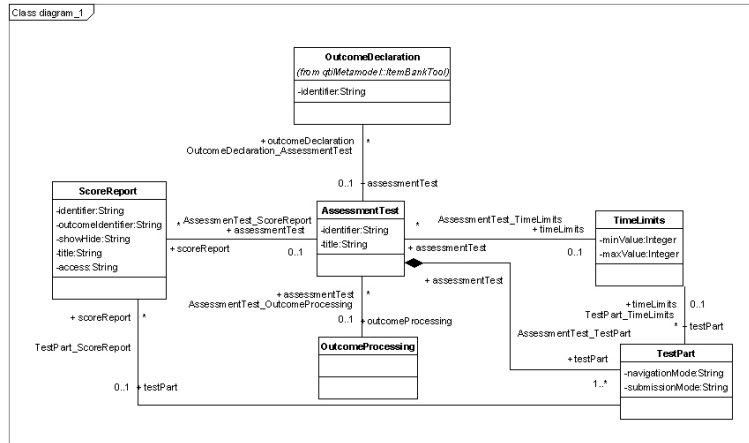
Fig. 6. Metaclasses in the Test Construction Tool Metapackage

6. QTI System Development Using the MDA Standard

The essence of QTI system development using the MDA standard is the transformation from Platform Independent Model (PIM) to Platform Specific Model (PSM) [7]. PIM specifies the system's functionality [10]. PSM is the specification of implementation details in a concrete technical environment. In the case of QTI system, the PIM is defined as a QTI metamodel and the models of different subsystems. The PSM can be generated using: Java Metadata Interfaces (JMI) [1], Ecore classes of the Eclipse Modeling Framework [12], and the QVT standard.

The transformation from PIM to PSM can be made in the Atlas Transformation Language (ATL) [3] that is used for model transformation. In case of a QTI system, it means that the assessment items will be transferred from Learning model to the corresponding Item Bank model (Figure 7), and then to the corresponding Test Construction Model in the process of evaluating a student's answer. The DL reasoning machine will check the consistency of the answer [14].
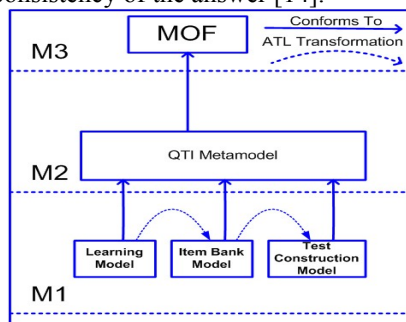

Fig. 7. The process of analysis the students' solutions in MDA

7. Discussion

In order to present the transformation from the QTI metamodel to the corresponding QTI model we used a simple choice example from the QTI standard, Figure 8. The part of the QTI metamodel that is relevant for this example is shown in Figure 9. This part of the QTI metamodel is transformed into the equivalent Ecore-based metamodel (EMF repository) using Eclipse plug-in. The part of QTI metamodel that refers to the simple choice item is shown in Figure 10. The result of this transformation is model given in Figure 11.

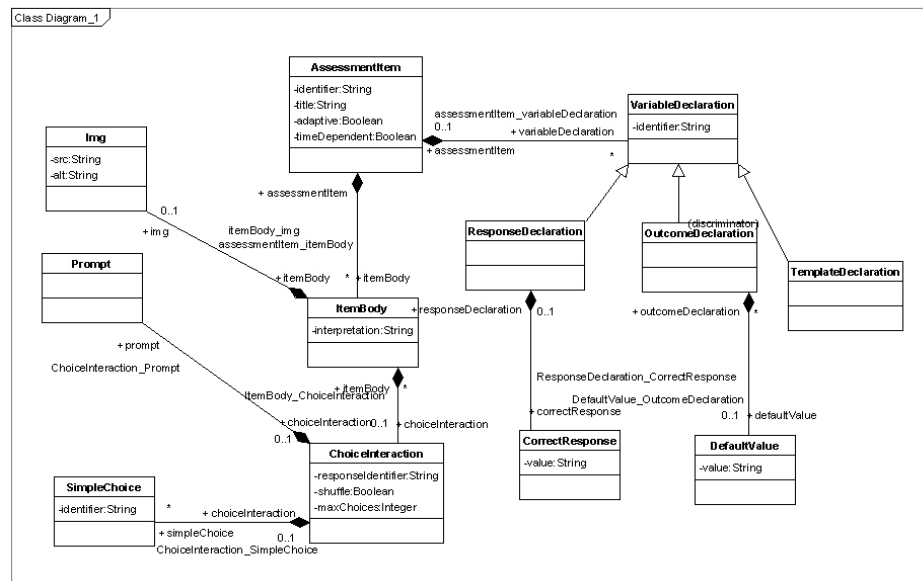**Fig. 8.** The example of simple choice item in QTI metamodel [17]

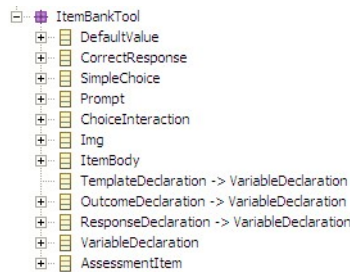**Fig. 9.** The QTI metamodel for the simple choice item

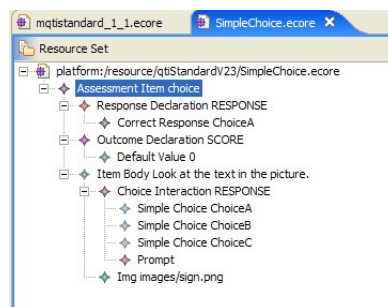**Fig. 10.** The QTI metamodel for simple choice item

Fig. 11. The QTI model for simple choice item

Using the model above, it is possible to use Java in creating, updating and accessing instances of this model.

The QTI metamodel is created in order to help assessment system developers to develop the QTI systems using Model Driven Engineering (MDE). The metamodel itself was cast in terms of UML, which means that it is understandable and useable for experts in assessment domain with knowledge of UML. This metamodel is, also, machine-readable because it is transformed from an MOF-based QTI metamodel into an ECore-based QTI metamodel by using Eclipse plug-ins. This metamodel could be extended for assessment systems that are not QTI-based.

8. Conclusions and Future Work

A QTI metamodel can help all developers of assessment systems who want to apply the MDA standard in creating a flexible, robust and, interoperable assessment system. Once they have a QTI metamodel, they need to create transformations, for example in ATL, to transform their PIMs to PSMs. By making a clear difference between conceptual and concrete modeling, developers can successfully select mechanisms to automate transfer and sharing of information and knowledge. It allows development to move smoothly between the levels of abstraction, with much of the work being automated. Each time a new type of assessment system appears, it means that it is necessary to change, or to enrich the QTI metamodel rather than the implementation.

At the same time, it is possible to improve a specified subsystem individually, without changing the other subsystems.

Future work will be focused on the implementation of transformations from TestConstructionModel to OWL model. The target model will use a description logic reasoner for checking the satisfiability of students' answers.

References

1. R. Dirckze, (spec. leader): *Java Metadata Interface (JMI) API Specification ver. 1.0,* 2002. `http://jcp.org/aboutJava/communityprocess/final/jsr040/`
2. Desirée Joosten – ten Brinke*, Jan van Bruggen, Henry Hermans, Ignace Latour, Jan Burgers, Bas Giesbers & Rob Koper *Modeling Assessment for Re-use of Traditional and New Types of Assessment* [Online]. Available: `http://dspace.ou.nl/bitstream/1820/355/4/modelling-assessment.pdf`
3. F. Jouault, I. Kurtev, *Transforming Models with ATL,* In Proceedings of the Model Transformations in Practice Workshop at MoDELS, Montego Bay, Jamaica, 2005. Available: `http://sosym.dcs.kcl.ac.uk/events/mtip/submissions/jouault_kurtev__transforming_models_with_atl.pdf`
4. *IMS Question and Test Interoperability Overview,* Version 2.1 Public Draft Specification, IMS Global Learning Consortium, Inc. [Online]. Available: `http://www.imsglobal.org/question/qti_v2p1pd/imsqti_oviewv2p1pd.htm`
5. *IMS Question and Test Interoperability Information Model,* Version 2.1 Public Draft Specification, IMS Global Learning Consortium, Inc, February 2006, [Online]. Available: `http://www.imsglobal.org/question/qti_v2p1pd/imsqti_infov2p1pd.html`
6. *ELENA Creating a Smart Space for LearningTM,* Available: `www.elena-project.org`
7. `MDA Guide Version 1.0.1,` Available: `www.omg.org/docs/omg/03-06-01.pdf`
8. Petros Lalos, Symeon Retalis, Yiannis Psaromiligkos, *Creating personalised quizzes both to the learner and to the access device characteristics: the Case of CosyQTI,* 3rd International Workshop on Authoring of Adaptive and Adaptable Educational Hypermedia (A3EH), 2005, Amsterdam.
9. *UML 2.0 Infrastructure Overview,* Available: `http://www.omg.org/issues/UMLSpec.pdf`
10. *OMG XMI Specification, ver. 1.2,* OMG Document Formal/02-01-01, 2002. Available: `http://www.omg.org/cgi-bin/doc?formal/2002-01-01.pdf`
11. Biggs J. B. (1999). *Teaching for Quality Learning at University,* Buckingham: Society for Research in Higher Education & Open University Press.
12. *Discover the Eclipse Modeling Framework (EMF) and Its Dynamic Capabilities,* Available: `http://www.devx.com/Java/Article/29093`
13. Dochy F., Segers M., & Sluijsmans D. M. A. (1999). *The use of self-, peer-, and co-assessment in higher education: a review,* Studies in Higher Education, 24, 331-350.
14. F. Baader, U. Sattler: *An Overview of Tableau Algorithms for Description Logics,* Studia Logica, 69(1), (2001), 5–40.
15. *Catalog of UML Profile,* Available: `http://www.omg.org/technology/documents/profile_catalog.htm`
16. *Eclipse J2EE IDE,* Available: `www.myeclipseide.org`
17. *IMS Question and Test Interoperability Examples Version 2.1 Public Draft Specification,* IMS Global Learning Consortium, Inc, February 2006, [Online]. Available: `http://www.imsglobal.org/question/qti_v2p1pd/examples/items/choice.xml`