Interoperability Issues between Markup formats for Mathematical Exercises

Giorgi Goguadze[1], Manolis Mavrikis[2], Alberto González Palomo[3]

[1] University Of Saarland,D-66123, Saarbrücken, Germany
george@activemath.org
[2] School of Mathematics, The University of Edinburgh,
EH93JZ Edinburgh, UK
M.Mavrikis@ed.ac.uk
[3] DFKI Saarbrücken, D-66123, Saarbrücken, Germany,
alberto@activemath.org

Abstract. In this paper we describe several existing knowledge representation formats for interactive exercises and how these address the representational needs of mathematical exercises. The paper also provides an overview of existing tools that support these formats such as players, rendering exercises and the role of mathematical services assisting these players. Since most of the developers of this family of languages have the same goal, it is now more possible than before, to reach a common representation format. Having this in mind, we discuss features, limitations and the interoperability between them.

1 intro

Introduction
   Authoring exercises, and particularly interactive and intelligent ones for mathematics, is one of the most time consuming processes in the e-learning field. In addition, as [10] describes, authoring is often conducted in a proprietary format of particular systems that hinders reusability, sharing and exchange between interested parties. Efforts are underway to establish standards and specifications that would facilitate authoring, reusability, exchange between educators and interoperability among systems. These efforts often have different perspectives but at least share some common goals making the transformation between data formats more possible than ever before. On the other hand there are still problems, since some formats are not generic enough, the representation is often bound to the underlying technology of the system, and the efforts of transforming such representations fail at these system specific parts.
   This paper reviews briefly these approaches and tools that support them, and describes some important interoperability and transformation problems among them in an attempt to reveal basic common subset of those formats.

2   Data formats for interactive exercises

2.1   QTI and related formats

The IMS Question & Test Interoperability (QTI) specification provides a data model for the representation of questions (and tests), the way to process the user's response and their corresponding results. It has been developed by the non-profit organization IMS Global Learning Consortium, whose mission is to promote the adoption of open technical specifications for interoperable learning technologies worldwide.

QTI's very first version appeared in early 2000. After a number of issues that were raised by implementers and other interested parties the specification was extended and updated a couple of times. Other issues uncovered more fundamental aspects of the specification that would require extensive clarification or significant extension in order to be addressed. As the QTI overview describes, "since the QTI specification was first conceived, the breadth of IMS specifications has grown and work on Content Packaging, Simple Sequencing and most recently Learning Design created the need for a cross-specification review".This review process led to a release of the second version which focuses only on the individual items and does not update those parts of the specification that dealt with the aggregation of items into assessments.

In this second version several question types are supported. MCQs, fill-in-blanks, even matching and other types of graphical interactions. A major change was the introduction of the concept of cloning and question templates which allow the creation of a set of questions based on randomly selected values which can be used in the text and response processing of the question. The new specification supports templates that define common marking schemes that processing engines can support in advance in a way that enables further interoperability. Of course, other than the default response processing templates, authors can declare their own processing instructions. The response processing model has been changed from the quite declarative one of the first version to one that supports more complicated response processing, that consists of a sequence of rules that are carried in order by the response processing engine. Based on some flag-like variables, which are set during the response processing, conditional feedback can be presented to the learner. Although version 2 also allows multiple parts of a question and adaptive items which change according to the response processing of previous steps, these parts are not reusable but bound to the specific item that includes them. In addition, authoring the several conditions and paths becomes very complicated due to the programming-like (yet XML-based) language that the response processing is based on. On the other hand, one could argue that this provides enough flexibility to encode complex and elaborate questions. However, as [1] observes, this greater flexibility inevitably provides multiple ways of implementing a response processing. This way one can use a more complex coding than what is required. Such redundancy does not introduce ambiguity, however, it can hurt interoperability as some engines will not be able to support the more complicated questions.

2.2   Mathematical exercises and QTI

It is worth noting here that QTI (especially its first version) was not explicitly developed with mathematics in mind (see discussions in [3,10,16]) and ignores many significant problems that authors of mathematical activities face. These have been discussed in several meetings and workshops of the community [12,14,13] and mailing lists[5,6]. The main issues are summarized here:

- Answers to mathematical questions, more often than not, involve mathematical entities which need to be understood by the system, processed and evaluated appropriately.
- Mathematical questions especially for Science and Engineering often test precision and accuracy of real values and therefore feedback and scoring needs to be adapted to the students' answers.
- It is not possible to have mathematics in all of the parts of the question (e.g title, statement, feedback) in an interoperable, platform-independent and system-understandable way.
- It is often necessary and faster to author mathematical questions, describe their response processing and the feedback they provide, by using mathematical constructs like variables that can also be randomized.
- QTI is weighted towards "teacher provided" response questions in which a student is required to make a certain kind of selection from responses provided by a teacher (for example MCQs). Actually, a teacher more usually would prefer to establish that the student's own answer satisfies certain (mathematical) properties.
- Mathematical questions often require the learner to provide an answer for which the form (e.g matrix, fraction, parts of a formula) is already provided (in order, for example, to test specific aspects or even to make it easier for the student to answer).
- There are many examples of question structures which research has shown to be useful and effective. These include such things as conditional hints, multiple parts and partial marking. Such features are now reasonably well established in mathematical CAA and their absence provides less flexibility during the question processing and leads to less effective feedback.

2.3   MathQTI as an extensions to QTI

The aforementioned limitations were some of the inspiration behind the JISC-funded "Serving Mathematics in a distributed e-learning environment"[4] project which aimed to develop open-source software and tools to address the special requirements of Mathematics in the context of e-learning and e-assessment. Part of the project looked into extending the IMS Question & Test Interoperability (QTI) specification in an way as compatible and convenient as possible, in an attempt to enable the exchange of questions with mathematical content between question engines and authoring tools.

---

[4] http://maths.york.ac.uk/serving_maths/

A result of this process was a first draft of the MathQTI specification[9] and the decisions to:

- use presentation MathML for questions statements and feedback.
- employ OpenMath[18] with a restricted content dictionary for the mathematical expressions in the template and response processing. An attribute determines how the OpenMath expression is dealt with (is it evaluated to a number, to an expression, simplified, or left unevaluated).
- introduce a new tag to test for syntactical equivalence in response processing.
- allow an alternative way of including interaction elements in question statements and feedback, which works via id's and the "for" attribute, further separating content from presentation and allowing mathematical expressions to be substituted by interactive elements.

Most of these aspects were inspired from other formats (like the ones in WaLLiS and ActiveMath, which are discussed in the following sections) in an attempt to bring them close and enable interoperability. On the other hand, as the format is based on QTI it benefits from its features but also suffers some of its limitations in relation to multiple parts, help or hint requests and adaptive response processing. Although it is appreciated that there will always be a tension between writing a specification which includes interesting "features" and the complexity of writing platforms which comply with emerging standards, features such as conditional hints, multiple parts and partial marking are now reasonably well established in mathematical Computer-Assisted Asessment (CAA) and their absence from a common specification would hinder its widespread approval.

2.4   ACTIVEMATH Exercise Format

The interactive learning environment ACTIVEMATH [15] possesses its own representation format for interactive exercises, which extends the OMDOC [17] knowledge representation. As in OMDOC, OPENMATH standard [18] for representation of mathematical formulas is used. This format was first defined in [5], then some refactoring was made in cooperation with the WALLiS project [10].

Some further changes were made over the last few years, based on experience gained from testing the capabilities of such a representation with authors in realistic conditions. The expressivity of the exercise language was improved based on real demands from an extensive test and study at a school (70 6th grade pupils) in Saarbrücken, a customisation as Matheführerschein[5], and a large study with 250 students at the Institute of Education of the University of London. These changes were mainly fuelled by the goal to reuse exercise encodings for several learning situations and different tutorial strategies.

The knowledge representation of ACTIVEMATH, as described below, was the basis for defining the less expressive knowledge representation for interactive exercises in the LEACTIVEMATH project. The current version of the ACTIVEMATH

---

[5] http://mfonline.activemath.org:8080/MatheOnline

exercise knowledge representation is described in [6]. In this format, a fully authored exercise is a finite-state machine consisting of a graph of `interaction`s, which represent the nodes of the exercise graph and transitions that connect different nodes in this graph. The basic structure of an interaction element is shown in Figure 1. An `interaction` contains one or more `feedback` elements providing textual[6] or graphical feedback to the learner. Each feedback is followed by an `interactivity assigment` specifying which interactive elements should be used in the `interaction` (e.g. fill-in-blanks, multiple-choice-questions). Each `interaction` can have one or more interactive elements of different types.

Finally a `transition map` element follows, containing transitions. Each transition connects two `interaction` nodes of the exercise. It contains a condition that has to be satisfied in order to jump from the current `interaction` to the target of the transition. Conditions contain comparisons of different types, that have to be applied to the user's input. A comparison can be syntactic — without simplifying the terms in the user's input, numeric — comparing numeric expressions, and semantic — where the external mathematical services are asked to perform the calculations in the given context.

Each `interaction` node can have metadata, specifying mathematical and educational properties of an `interaction`, such as *difficulty* of a step w.r.t. the particular *learning context*, *competency* to be trained in the step, *concepts* that this step is elaborating, specifying whether the *hint* is given, and so on.

Each transition can have diagnosis information in case the user's input satisfies the condition of a given transition, such as the *achievement* of the learner w.r.t. the task, *relevance* of the answer and/or *misconceptions* the learner had in the step.

Fig. 1. ACTIVEMATH Exercise Subsystem architecture

Importantly, due to its compactness and extensibility, the ACTIVEMATH exercise format makes it easy to build more complex representations. This has often been requested over the last few years of the application of ACTIVEMATHby authors and users. Therefore a property of the system is that exercises can also be fully or partially generated and the graph of interactions in such exercises can be dynamically changed. Tutorial strategies applied to a compactly encoded exercise will enhance the exercise by introducing additional nodes, cloning existing ones, adding more connections between them.

---

[6] A recent development allows ACTIVEMATH to read the feedback text aloud.

2.5   WaLLiS format

WaLLiS is a web-based system developed at the School of Mathematics of the University of Edinburgh. A detailed description can be read at [10]. WaLLiS employs interactive exercises that in a way have a predetermined context as they are designed for a specific learning situation. Yet they are adaptive in the sense of what kind of feedback they provide. These interactive exercises are authored in an XML format which is very similar to the initial ActiveMath one [5], whose main characteristic was that it also employs OpenMath to represent the formulas needed for representing the response processing conditions. This format was a direct consequence of early experience from the system's use and authoring content and needs of the context where it was used. Several of the issues mentioned in section 2.1 actually arose during this period.

In addition, in an attempt to address the reusability and ease of maintenance of multiple parts, it was decided that it was better to link rather than nest items and their parts. Thanks to the collobaration between the authors of this paper the two formats are now isomorphic yet their players follow complete different architectures, demonstrating the power of employing generic formats that are not bound to implementation. ActiveMath has a clear client-server architecture (described at section 5.1) and the WaLLiS system follows a series of XSLT transformations that result to an XHTML interactive document which is presented to the learner. Briefly, the transformation results appear on a browser page that includes a form for the interaction, hidden elements (eg. the solution or the hints) and embedded JavaScript that takes care of the interaction and the communication with the server to evaluate the learner's response, without the learner ever leaving or submitting the page. More details are provided at [11,?]. The main point here is that despite the differences the isomorphic formats have been used to share exercises by transforming from one to the other and the fact that both were adequate for two complete different implementations provides positive indicators towards realising the ideal situation of having a common format or at least translating automatically between them without loss of functionality.

2.6   LeActiveMath Exercise Language

The MathDox language [8] is developed in RIACA [7] (TUE) and is based on the former representation, influenced by DocBook [8] and OMDoc languages, called MathBook (RIACA) [7]. OpenMath is used for mathematical formulas here as well. The language is mainly closer to the QTI one with more flexibility (but also increased complexity) in the response processing where specialised tags ("if", "then") are used to represent the response processing.

The LeActiveMath exercise language is developed in RIACA within the LeActiveMath project and is based on the ActiveMath exercise language

---

[7] http://www.riaca.win.tue.nl

[8] http://www.docbook.org

with some additions, inspired by MathDox. For instance, it makes heavy use of state variables, called "dynamic context" in MathDox. In addition the language allows MONET queries [16], sent to the external mathematical systems.

3 Tools for Authoring and Interpretation of the Exercise Languages

3.1 The ACTIVEMATH Tools

The main component of the ACTIVEMATH learning environment is the exercise subsystem, developed by two of the authors. It comprises several components communicating within a modular architecture. Describing this in detail is beyond the scope of this paper. Therefore we describe only those parts of the exercise subsystem architecture which deal with interpretation of the knowledge representation of the exercise steps and their generation.

Briefly, the core of the exercise subsystem consists of an *Exercise Manager*, *Interaction Generator* and *Evaluator/Diagnoser* components. The Exercise Manager is responsible for communicating with the database and the client (typically a web browser). It receives the problem statement of the step and the user input, forwards them further to the Interaction Generator and asks it to return the next `interaction`. The Interaction Generator queries the Evaluator/Diagnoser for diagnosis of the user's action, and builds a new `interaction` based on the result of this diagnosis.

The Interaction Generator base class provides an architecture for defining different Interaction Generators. The default Interaction Generator used in ACTIVEMATH is called *Static Interaction Generator*. It is designed for retrieving `interaction`s in manually authored exercises. In this case, the Evaluator has to check whether the user's input satisfies one of the manually authored conditions provided for the current `interaction`, and the conditions point to the `interaction`s to be executed in the next step in case they are satisfied. The benefit of this approach is that, instead of the Static Interaction Generator, other specific generators can be employed. Each Interaction Generator can define it's own custom Evaluator/Diagnoser class, extending the basic Evaluator/Diagnoser for more advanced queries of the mathematical services.

Since the exercise subsystem of ACTIVEMATH allows for so-called virtual exercises, in which the `interaction`s are generated, writing such interaction generators can give birth to whole classes of automatically generated exercises. There already exist several custom interaction generators such as the "Randomizer", which extends the static one by randomizing variable values in the manually authored exercise. Another generator defines a custom Evaluator that queries the available services such as Domain Reasoners or CAS in each step, in order to receive the diagnosis on the user's actions. Based on the result of this diagnosis, the next `interaction` is generated.

In order to facilitate reusability of exercises in different learning situations, additional generators can be applied to an authored or generated exercise, in a

similar way as the randomizer mentioned before. These generators implement different tutorial strategies which define the sequence in which the steps are rendered, the frequency of hints and other parameters that may vary in different learning situations. Such a flexible system of cascading interaction generators is the result of a careful design of an exercise language and the corresponding engine, interpreting and rendering this language in a modular and extensible way.

3.2   RIACA Exercise Repository Tools

The Exercise repository, developed within the LeActiveMath project, provides an authoring tool for exercises that can be encoded in the LeActiveMath exercise language. This authoring tool allows for constructing exercises with fill-in-blanks and multiple-choice-questions. Exercises can have multiple steps and connect to CAS for checking the answers of the learner. Certain randomization of exercises is also possible. The repository itself is provided as a web-service that serves interactive exercises that can be used directly in the web or to be sent and rendered in another system (for example LeActiveMath).

3.3   The MathQTI engine

The MathQTI engine[9] is a module of WaLLiS that interprets a MathQTI exercise in a similar way as described in section 2.5 but in way that is separated from the rest of the system, with the hope of reusing it as an engine for other systems. It is worth mentioning here that for interpreting the OpenMath formulas and handling the response processing with the Yacas system (which is used at the background for evaluating the learner's answer) the RIACA JSP (Java Server Pages) tag libraries[10] are used. With simple extensions and some additional in-house coded libraries the math-qti engine provided a proof-of-concept system for further developments of WaLLiS and other systems. This demonstrates, once again, that once based on common standards (such as OpenMath) at least a partial interoperability and code reusability is possible.

3.4   Protocols and web services

Within the LeActiveMath project, generic web-services support is developed in the ActiveMath system. It consists of an OpenMath broker component, which receives mathematical queries from the ActiveMath exercise system, and distributes these among available web services, capable of providing an answer. Among the services connected to such a broker, are Computer Algebra systems (in our case Yacas, WIRIS and Maxima) and other mathematical systems such as Domain Reasoners, written in Prolog, that all communicate with the ActiveMath broker in a variety of ways, including complete SOAP (Simple Object Access Protocol)[11] queries, XML-RPC (Remote Procedure Call), and

---

[9] http://sourceforge.net/projects/walis

[10] http://www.riaca.win.tue.nl/products/taglib/

[11] http://webservices.xml.com/pub/a/ws/2001/04/04/webservices/index.html#soaphead

local system processes via pipes, all of them transparent to the rest of the system and to the content authors and learners. The queries defined there have the following parameters:

– the name of the query
– one or more formulas in OPENMATH format
– context of the query defined by a set of concepts in OPENMATH format
– depth of the rule application

On one hand such queries can be used for semantic comparison of the user's input within the condition in the transition map of the authored exercise. On the other hand, several other queries can be sent for receiving more detailed diagnosis of the learner's actions and, based on such diagnosis, construct the next node in the generated exercise.

It is worth noting that the LEACTIVEMATH exercise language of RIACA, used for the separate repository of interactive exercises outside the ACTIVE-MATH system, directly defines MONET queries inside the content of the exercises. These queries can specify the name of the concrete CAS to be used for evaluation. In the opposite case the available CAS is selected automatically. Therefore, the exercises which specify the concrete CAS to be used are not guaranteed of reusability with other CAS. On the other hand, it can be argued that this way authors of the exercise can be sure that the result of their query will be as expected and therefore pedagogically more sound.

Finally, it should be mentioned that efforts are underway to define common protocols for exercise players. For example, the Remote Question Protocol (RQP[12]) that allows e-learning systems to use external services for presenting and evaluating a question. Other query languages like the MONET Mathematical Query Ontology, as well as the developments of mathematical web services will enable more and more Interactive Learning Environments (ILE) to employ facilities such as elaborate question players, Computer Algebra systems or other processing engines, thus enhancing their functionalities and allowing more elaborate questions not limited by the features of an specific system.

4 Transforming Between Formats

We have mentioned before that several transformations are possible between formats. For example. because of the close collaboration between the authors, back-and-forth transformations between the WaLLiS and the ActiveMath formats is possible. At least for now we are interested in the main framework of the exercise rather than all the medatata and additional information that is included in both of them. It is more important, at least at this phase, to be able to automatically convert the questions statement and response processing even if that does not entail the fully fledged result reporting and metadata annotations that are still perhaps dependent on the context and use of the exercises.

---

[12] http://mantis.york.ac.uk/moodle/course/view.php?id=14

Similarly, exercises from QTI have been transformed to the WALLIS format. This was certainly easier for the first version of QTI (v1.2) which was more descriptive in nature than v2. Since we are able to transform between the WALLIS format and the ACTIVEMATHand LEACTIVEMATHone it is hoped that in the future transformations from QTI to these formats will also be possible. It has to be said here that some exercise types (such as matching and puzzles) are worth investigating further as until now they were not our primary target. On the other hand we have found that only simple exercises from the WaLLiS formats (without multiple steps and interactions) were transformable to the QTI one. In addition, exercises from QTI v2 are easily transformed only when these use the predefined response processing templates. Otherwise, we have found that exercises that were authored using the full flexibility of QTI v2 were impossible to transform in a generic way (e.g. an all-purpose XSLT) mainly due to the prescriptive nature of QTI v2. Once extra knowledge on the rationale behind the exercise, the flags it uses, and the naming conventions are included to the transformation process the task is simplified. Unfortuanately the fact that human intervention is required complicates and makes the process time consuming. This is certainly a matter worth investigating further.

Some mappings that are certainly planned and partially made include the back-and-forth mapping between MathDox, ACTIVEMATH, WALLIS and Math-QTI. For example, one of the expected outcomes of the LEACTIVEMATH project is the interoperability of ACTIVEMATH and LEACTIVEMATH exercise languages. The mappings are mostly straightforward, but not all of the formats are isomorphic. Some of them are bound to a particular implementation rather than designed with reusability in mind. It is certain that the notion of dynamic states and the variable flags used in MathQTI require additional knowledge of the context making the automatic translation even harder. To avoid these problems it seems necessary to avoid encoding information which provide additional knowledge such as pedagogical knowledge rather than just the exercise itself. It is in the benefit of everyone who is involved in the process of authoring and deploying content in real situations, to be able to author and use exercises in different educational situations and goals. We believe that as with other cases where interoperablity was achieved mainly due to the separation of layers (such as the early days of XML and the separation of content and presentation) the dream of interoperablity will be achieved if more formats avoid as much as possible to mix pedagogical knowledge or other information with the response processing.

5   Conclusion

It is clear from this paper that further work is necessary in order to identify the most generic exercise format suitable for most of the applications and try to join the efforts of the involved communities and produce such a uniform format. This seems to be reasonable, since all the formats considered in this paper (except QTI) not only share common properties, but are not far from being isomorphic. By reaching this common agreement, all the involved communities would gain

variety of tools and players, developed by each side, that will become reusable for others but more importantly a common pool of exercises and content that can be reused and shared between them.

Acknowledgement

References

1. D. Bacon. Review of QTI v2.0. Available online:
   <http://support.imsglobal.org/question/qtiSupportIndex.html>
2. A. Cohen, H. Cuypers, D. Jibetian, M. Spanbroek, LeActiveMath Exercise Language, Deliverable D7., LeActiveMath Project, FP6 Framework, 2005.
3. A. Cohen, H. Cuypers, E.R. Barreiro, H. Sterk Interactive Mathematical Documents on the Web. Algebra, Geometry, and Software Systems 2003, pages 289–307, 2003.
4. Global Learning Consortium. IMS Question & Test Interoperability Specification: A Review <http://www.imsglobal.org/question/whitepaper.pdf>
5. G. Goguadze, E. Melis, C. Ullrich and P. Cairns, Problems and Solutions for Markup for Mathematical Examples and Exercises. In Proceedings of the Second International Conference on Mathematical Knowledge Management, MKM03, Andrea Asperti (ed.). <http://www.ags.uni-sb.de/~ilo/articles/EncodingExoExa.pdf>
6. G. Goguadze, A.G. Palomo, E. Melis, Interactivity of Exercises in ActiveMath. In Proceedings of the 13th International Conference on Computers in Education (ICCE2005), 2005.
7. MathBook Standard, Research Institute for Applications of Computer Algebra <http://www.riaca.win.tue.nl/products/mathbook/>
8. MathDox Website <http://www.mathdox.org>
9. M. Mavrikis. MathQTI draft specification overview. Available online at <http://www.maths.ed.ac.uk/mathqti/docs/v0p3/mathqti_v0p3.pdf>
10. M. Mavrikis and A. Maciocia. Wallis: a web-based ILE for science and engineering students studying mathematics. *Supplement Proc. of the International Conference on AIED*, 2003.
11. M. Mavrikis and A. González Palomo. Mathematical, Interactive Exercise Generation from Static Documents. *Electronic Notes in Theoretical Computer Science*, 93:183–201, 2004.
12. C. Miligan. Question and test interoperability (QTI): Extending the specification for mathematics and numerical disciplines. *LTSN maths-CAA series*, Nov 2003. Available online at <http://ltsn.mathstore.ac.uk/articles/maths-caa-series/nov2003/>

13. M. Mavrikis. Thoughts on MathQTI (see "documents on MathQTI" section of the "serving maths" project: <http://maths.york.ac.uk/serving_maths/mod/resource/view.php?id=103>). Technical report.

14. Minutes of 1st MathQTI workshop. Available online: <http://maths.york.ac.uk/serving_maths/mod/book/view.php?id=154>

15. E. Melis, E. Andrès, J. Büdenbender, A. Frischauf, G. Goguadze, P. Libbrecht, M. Pollet, and C. Ullrich. ACTIVEMATH: A generic and adaptive web-based learning environment. *International Journal of Artificial Intelligence in Education*, 12(4):385–407, 2001.

16. MONET: Mathematics on the Net, EU Project <http://monet.nag.co.uk>

17. OMDoc: An Open Markup Format for Mathematical Documents, <http://www.mathweb.org/omdoc>

18. OpenMath Society Website, <http://www.openmath.org>