

## Modular Database Approach to Manage a Large Set of STACK Questions

Jukka V. Paatero<sup>1</sup>, Aleksi Mankonen<sup>2</sup>, Juho Ratava, Barkat A. Bhayo, Paula Immonen, Markku Kuosa, Mikko Äijälä

**Abstract:** This study presents a novel approach to producing, managing, and updating a large set of questions on the Moodle-integrated STACK platform. The proposed method consists of an automatic construction of questions using modular components. This reduces the workload required for mass-authoring and maintenance of questions and makes the authoring more flexible when changes are required. A set of standardized, module-based question templates allow the formulation of new questions with only limited knowledge of STACK, including integration of parametrized graphics. The proposed method has been tested and validated with a large set of physics questions, demonstrating a significant reduction in workload and improvement in the quality and relevance of the questions. The use of modular structures allows for greater flexibility and customization in the design of the questions. Overall, this study provides a valuable framework for educators and instructional designers who seek easy approach to extensive learning materials production.

**Keywords:** STACK questions management, modular database, flexibility, mass-authoring.

### 1 Introduction

Online learning with automated Computer Aided Assessment (CAA) has advanced significantly during recent decades [Broughton 2020]. In particular, the increased distance learning during 2020-2 gave a substantial boost to all forms of off-campus assessments. The main advantage of using CAA is that it potentially enables the management of a large number of students with limited teaching resources [Broughton 2020]. However, the implementation of high-quality self-learning materials and automated CAA requires significant work input.

There exists a large number of CAA systems, and most of them are well suited for teaching and assessing mathematics and physics [Sangwin 2013]. The *System for Teaching and Assessment using a Computer algebra Kernel* (STACK) is one of the popular CAA

---

<sup>1</sup> LUT University, LUT School of Energy Systems, P.O.Box 20, FI-53851 Lappeenranta, Finland, jukka.paatero@lut.fi

<sup>2</sup> LUT University, LUT School of Energy Systems, Mikkulankatu 19, FI-15210 Lahti, Finland, aleksi.mankonen@lut.fi

systems often used with specific Learning Management Systems (LMSs) like Moodle<sup>3</sup> and Ilias<sup>4</sup>. The work discussed here is based on the physics and mathematics teaching with CAA on the Moodle LMS.

The CAA system STACK adds a large set of advanced mathematics (and physics) assessment options to the assessment tools available on Moodle by default. Some of the key advantages of STACK include displaying mathematical texts, student-accessible language for submitting answers in both advanced symbolic and numeric format, the possibility to construct detailed multi-part question-evaluation-feedback progression pathways using potential response tree structures, and randomization of selected task parameters to produce variants of a single question [Sangwin 2013].

It is typical for any CAA system that authoring new questions with good quality online pedagogics is typically time-taking. Thus, authoring many questions requires a major investment of time, while some question components can be effectively manually copied from earlier questions. Similarly, updating large numbers of questions individually via the default STACK interface on Moodle is time consuming. Overall, these same challenges apply to question authoring and updates with STACK. The production and management of a vast multitude of STACK questions form a challenge in terms of invested working time and their sensitivity to errors resulting from misspellings and copying earlier content sections.

In this paper, the authors suggest an alternative approach for mass-scale authoring and updating STACK questions. The main idea is not to work via the Moodle-based STACK interface, but instead to work directly with the XML-format files that are used to import and export STACK questions into and out of the Moodle environment. The leading idea is to limit the freedom in authoring by employing pre-structured STACK question formats with pre-determined choice trees. Such questions can then be divided into modular key components, and all authoring efforts can be focused on the modules that include the parts subject to change between the questions in question. These modules can then be linked into complete STACK questions.

This approach potentially enables streamlining the STACK question authoring and updating by using a modular database of question components and procedural production of the questions from the database. When only the question modules are edited, this allows using a simplified component-based interface. This makes the authoring easier to begin with, and lowers the learning curve required for a person to begin authoring and maintaining questions. Further, one can simply focus on some parts of the questions, like authoring feedback sections.

---

<sup>3</sup> See <https://moodle.org>

<sup>4</sup> See <https://www.ilias.de/en/>

## 2 Methodology

Modifying STACK questions using the drop-down menus and text fields of the online editor can prove an inefficient way of managing a larger set of questions. Many times, questions share standard features that the question author wishes to use in many questions. One way to reuse a part of a question is to save the text rows in a file and copy them to new questions. Saving text rows only works for content that is stored in text fields and cannot be applied to other types of content, such as drop-down menu items. All the STACK question's data, including drop-down menu-controlled settings, can be exported to a XML file where it can be edited easily, for example, using a text editor. The question in this file format is naturally composed of sections that correspond to the different elements edited via the fields and functionalities available on the online editor. The methods applied here are based on editing the questions in XML file format.

### 2.1 Simplifying management by modularity

The naturally occurring sections in the questions can further be organized and separated to form modules in XML file format that correspond to intuitive functionalities and components of the questions. Such modules can, for example, contain general question settings; question structure settings; introduction text of the question; lists of symbols and variables; answers, mathematic equations, and solutions; grading algorithms; and components of student feedback like hints, specific feedback, and general feedback. This kind of modularization is primarily layout-based. [Dí02]

These modules offer a universal way to organize and reuse all question parts and structures when forming new questions. Commonly used data and sections can be stored ready to use, such as logic programming, symbol lists and visualizations. For example, reusing the same potential response tree in authoring several questions becomes trivial and virtually error free even for complex tree structures, while when authoring with the web editor, this is one of the most time-taking and error-prone components.

To effectively use a fleet of question module components, the author needs to maintain a list of modules utilized in each of the questions. Often questions with similar logic and structure can differ in only couple of modules. The modules now form a local body of data that is used to author questions. This data is processed to become an uploadable, viewable, and answerable Moodle STACK question. In Moodle, the questions are stored on the server database as normally. This also allows maintenance of the entire question database with the “single point of change” design principle. [GPT12]

### 2.2 Procedural construction of questions

To finalize questions, the modules needed for a question must be organized into a single XML file that includes all required modules in the correct order. While individual

questions can easily be built manually with a text editor, effective management of many questions and their modules requires an automated approach.

The design for the automated building of questions requires first identifying the repeating structures and refactoring the question data so that the structures common to most questions and changing structures are placed on different modules. In principle, there are three kinds of modules: data that is common to all questions (e.g., common XML structures), data that is common to the questions which share a similar structure (e.g., a question with an equation as an answer) and data that is unique to a question (e.g., question text). By combining the required parts, it is possible to build a complete question or a set of questions; changing a single module file allows updating all questions using that module at once. [GPT12]

### 3 Results

The implementation of the question modularization, their procedural building and management has been carried out on the proof-of-concept level. The main effort has been in demonstrating the idea behind this implementation, while the practical user experience is still very primitive. In practice, the implementation was carried out in the Windows operating environment and Moodle learning management system. This choice was motivated by simple and easy access to the operating environment shared by the authors. The module data is stored as XML data in a form that is readily understood by Moodle, and the build system needs to simply concatenate the data together. In Moodle, the imported XML is stored in the database of the learning management system, and the XML structure will vanish. Both the question data and the XML storage file for the question are designed to be agnostic to the build system. This approach to authoring results in question data that is locally in a very different format than on the Moodle server. Every different question archetype has a dedicated script to manage it. While the initial testing was done with only one question archetype, now several archetypes have been tested successfully.

Using the approach of directly modifying XML files has proven to be incredibly efficient in the centralized updating of questions. When several questions share a module, such as the list of symbols, updating the formatting of the symbol list to any amount of questions can be done in a matter of minutes. The same procedure with the Web-based interface would take days. Practical implementations of the proof-of-concept level authoring and updating questions have already been tested in small and large math and physics courses with 30-300 students.

At this stage, the following modules have been identified as easily adaptable to physics and mathematics questions: the logo of the authoring organization; a list of symbols of variables, their units, explanations, and input instructions; an algorithm for converting a number to be displayed to LaTeX format for students; the algorithm for evaluating numerical answer, order of magnitude, sign and accuracy of the responses are graded; and

general feedback (worked solution) text layout. These modules have been successfully updated in a centralized manner for all questions without a need to access data of individual questions.

As part of the proof-of-concept implementation, the question modules have been modified into formats that are more intuitive for text-based editing. This leaves a more significant finalizing role for the building script while it eases the authoring. For example, question text fields have been reorganized into a table-like format with a text field followed by a variable when eligible. Each language version has a separate table with the same variables. In a similar manner, variables used in the question are also submitted in a table-like format. This kind of customized module improves the authoring speed and quality and helps to avoid and find errors. At a later stage, a more advanced editing interface will be utilized, resulting in even more distance between the implemented modules and the corresponding parts of the final question.

A key part of constructing the question is the script that concatenates the needed individual text files (modules) and writes content directly included in the script file. Conditionally formatted content is directly included in the script instead of a separate text file. The script identifies which content is written to the output XML files based on specific rows in the input text files. For example, the number of input variables fields is identified from an input text file and the corresponding number of input fields and potential response trees are written to the output XML.

as weekly exercise sets have been written to the XML file by the script significantly reducing the work of categorization of questions through the Web-based interface.

The features of creating environment variables and using conditional branches of Batch scripting language was used to conditionally add and modify the content of the question. For example, the code for grading numerical answers is not included in the question if there are no numerical questions, thus making the question execution faster and the loading time shorter for the students. In the current stage, the build system will identify the modules present for the question by having each question in a separate folder and checking if a relevant file or included reference exists in that folder.

Questions managed in a centralized manner have been successfully implemented in statistical mathematics and physics courses. The effort of creating new questions with existing modules was significantly reduced as compared to creating new questions in the Web-based interface. The flowchart of building a single question has been demonstrated in Fig 1. In the case of several question archetypes, a similar structure applies for each of the archetypes. However, some of the shared modules are shared between all archetypes. Such modules would include the logo of the authoring organization and an algorithm for evaluating a numerical answer.

The implemented modular approach was shown to effectively support the multi-authoring approach in authoring STACK questions. Multiple authors worked with the questions through an online shared folder in a cloud service instead of the Web-interface that easily

create conflicts. However, the web browser was used for testing the questions. Authors were now able to focus in parallel on specialized components of the questions, like graphical elements, question texts and variables, mathematics of the question, and feedback. The Batch script files were run by all question authors, while the scripts were updated by only one author to avoid confusion.

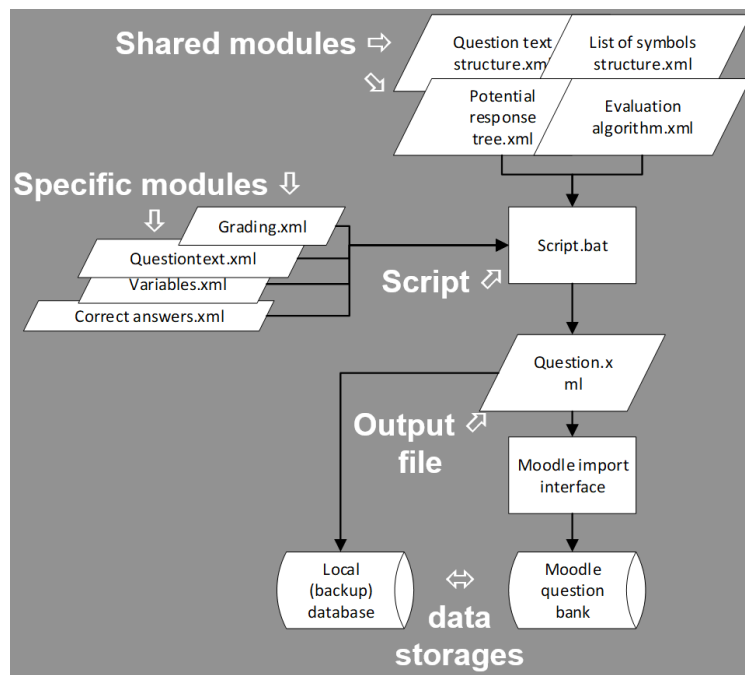


Fig. 1: Flowchart for building a single question based on shared and specific modules.

A separate script has been used for concatenating complete questions as question sets. Questions related to a particular course have been written to a single file and brought to the learning environment Moodle with one drag-and drop action. The question categories

The authoring using several question modules, separates different question components into separate XML files. Fig 2 demonstrates how the content from various modules forms the full questions. The figure has one example from physics and one from mathematics questions. However, the figure only includes modules that contain visual elements connected to the displaying of the question itself.

### Mathematics question

Module: Logo

Module: question text

Insert the composite functions  $f \circ g$  and  $g \circ f$  when  $a = 2$ ,  $b = 1$  and  $f = 4 \sin(x)$ .

What is the algebraic expression for composite function in terms of  $x$ ?

$f \circ g =$   Module: algebraic input field

What is the algebraic expression for composite function in terms of  $x$ ?

$g \circ f =$   Module: algebraic input field

Function	Latex	Symbol	Unit	Input instruction
addition	$a + b$	$a+b$		
sine	$\sin(a)$	$\sin(a)$		
multiplication	$ab$	$a*b$		
division	$\frac{a}{b}$	$a/b$		
independent variable	$x$	$x$		
function	$g(x)$	$g(x)$		
function	$f(x)$	$f(x)$		
composite function	$f \circ g$	$f \circ g$		
composition function	$g \circ f$	$g \circ f$		

Module: List of symbols

### Physics question

Module: Logo

Module: question text

When the switch  $K$  in the shown circuit is open, a current of  $I_0 = 11.8000$  A flows in the circuit. Once the switch is closed the current is  $I_c = 3.9000$  A. The resistances of the circuit are  $E = 4.9000$  V and  $R_1 = 9.4000$   $\Omega$ . Calculate the source voltage and internal resistance of the voltage source.

Module: illustration

What is the resistance?

$R_2 =$    $\Omega$  Module: numerical input field

What is the internal resistance?

$r =$    $\Omega$  Module: numerical input field

Insert only 3 most significant digits of your answer without rounding.

Quantity	Symbol	Unit	Input instruction
current	$I_0$	A	$I_0$
current	$I_c$	A	$I_c$
source voltage	$E$	V	$E$
resistance	$R_1$	$\Omega$	$R_1$
resistance	$R_2$	$\Omega$	$R_2$
internal resistance	$r$	$\Omega$	$r$

Module: List of symbols

Fig 2: The visibility of graphical module elements in the final STACK questions.

## 4 Conclusions & Discussion

Online learning material for Moodle/STACK platform consists of frequently occurring code/text sections and sets of options. The authors have demonstrated that the question

contents can be effectively accessed and modified through XML files, making it possible to manage the content more efficiently. Content that can be reused effectively is stored in a text file and included in an XML file with a script. The automatization of trivial tasks with a script was proven efficient especially in updating a large set of similar questions. The modular management of questions proved very suitable for multi-authoring and minimizing time spent dealing with typing errors and minor updates.

Using the scripts in updating and importing the questions makes it possible to introduce new learning content and modify the existing content faster. New technical solutions of question management of this research give question authors the possibility to focus on creating desired learning content instead of spending time on trivial copy-replace tasks.

The separation of question text simplifies also adding different localization of the questions. Now adding language versions requires only touching the files containing question texts. Ideally, the build system would be developed to check file dates since last build to automatically update only those questions which contain changed data.

## Bibliography

- [BHR20] Broughton, S.J.; Hernandez-Martinez P.; Robinson, C.L.: The effectiveness of computer-aided assessment for the purposes of a mathematical sciences lecturer. In: (I. Management Association Ed.) Learning and Performance Assessment: Concepts, Methodologies, Tools, and Applications. IGI Global, Hershey, PA, pp. 639-655, 2020.
- [Di02] Díaz, O.; Irastorza, A.; Rodríguez J.J.; Anfurrutia F.I.: An overview on XML initiatives to bring modularization to web application development. In: (Isaias, P. ed.) Proc. IADIS International Conference WWW/Internet, Lisbon 2002. IADIS Press, Portugal, pp. 435-443, 2002.
- [Sa13] Sangwin, C.: Computer aided assessment of mathematics. OUP Oxford, 2013.
- [GPT12] Giunta, R.; Pappalardo, G.; Tramontana, E.: AODP: refactoring code to provide advanced aspect-oriented modularization of design patterns. In Proc. 27th Annual ACM Symposium on Applied Computing. pp. 1243-1250, 2012.