ISBN: 978-989-8533-58-6 © 2016

AUTOMATED ASSESSMENT IN MASSIVE OPEN ONLINE COURSES

Dmitrii A. Ivaniushin, Dmitrii G. Shtennikov, Eugene A. Efimchick and Andrey V. Lyamin
ITMO University

ABSTRACT

This paper describes an approach to use automated assessments in online courses. Open edX platform is used as the online courses platform. The new assessment type uses Scilab as learning and solution validation tool. This approach allows to use automated individual variant generation and automated solution checks without involving the course author. The approach implementation is based in XBlock SDK provided by Open edX developers team.

KEYWORDS

MOOC, programming assignments, Scilab, automated assessment

1. INTRODUCTION

Nowadays Massive Open Online Courses (MOOCs) are trending in both e-learning and traditional learning. One of the most popular open education platforms is edX. In June, 2013 the platform's source code was released on Github and everyone willing got an opportunity to deploy his own edX platform instance and create courses there. The open-source version of the platform was named Open edX, indicating being open software.

Typical MOOC consists of video lectures, quizzes, tests and discussion boards. The test items can be represented by various types, for example, text input, drag-n-drop, sequence problems, and virtual laboratories with automatic check. Also some MOOC-providers support more complex problems types, where student should give an answer by uploading an artifact, such as essay, picture, etc. Student should accomplish an assignment following to instructions given by course author and upload the outcome to the system.

Some assignment types are difficult to assess such as programming task or modeling problems. Generally, assessments that check modeling skills are assessed by experts (usually, teachers or assistants), but including those types of problems in massive online courses causes some issues regarding number of students. Since number of students grows, such types of assessments become more difficult to check due to the number of solutions uploaded. It becomes even worse when each student is assigned a randomly parametrized problem, though these type of problems can be graded using determined instructions in an automatic mode.

Scilab is an open source, cross-platform numerical computational package and a high-level and numerically oriented programming language. This package can be used to assess student skill to perform mathematical modeling or skill to perform computing of any kind, for example to calculate matrices parameters. Common assignment types, which can use Scilab packages as a validation tool, are mathematical or physics problems including process modeling, programming problems that use Scilab programming language for modeling purposes.

This paper proposes an approach to assess such assignments using Scilab as student learning tool and student solution verification tool. The main aim of the approach is to provide individual and random parametrization for problems.

2. MASSIVE OPEN ONLINE COURSES PLATFORM

Open edX is a popular platform that provides possibility to hold massive open online courses. Numerous instances are deployed all over the world and its number grows everyday.

Its main components are learning management system (LMS) and content management system (Studio, earlier – CMS). LMS provides the learners (the students) with access to learning materials, such as video lecture, assessments and forums. Students can pass assessments and earn grades depending on their success using this system.

CMS is a system where course author can create courses, configuring its outline and the course structure in general. Authors can also add video lectures using special UI components, configure the due dates of assessment submissions and so on.

Open edX platform support different assessment types. Such assessment types as text input, image input, code input, sequence input, drag-and-drop input are provided. The number of them is continuously growing, and it is possible to extend the quantity of supported components using an application programming interface named XBlock. This interface was created by the Open edX developer team to extend support of custom components which are not supported out-of-the-box. The most common use of this interface is interconnection of different services. Many corporate systems have been created and have collected much data that can be re-used. By now many systems have already developed different solutions to support behavioristic assessments, which are not supported by Open edX, but using XBlock interface makes it possible.

There is no support of Scilab-assessments with source code in Scilab programming languages required as student submission implemented by Open edX team, so it may be implemented via XBlock API.

Usually, when saying the edx-platform combination of CMS and LMS is meant, but there are much more components in there. One of them is XQueue. XQueue is a component which provides messaging services. Messages are text strings in special format. The format of messages is defined by its users: consumer and producer. In this paper producer is LMS and consumer is Scilab-server. The idea behind the queue is that all the requests are performed asynchronously with the first-in-first-out principle. It also makes possible to enqueue such operations as variant generation and submission check, and close browser pages from which those actions were requested, as they still will be performed when the server is ready.

3. WORKFLOW SCENARIOS

General scenario may be described from two points of view: student's point and course author (instructor) point.

The overall idea that stands behind the process is that the system automatically validates user submissions written in Scilab programming languages. To make it work some restrictions are to be given.

By design each student should receive individual task variant. This is possible when some rules of generation process are provided. Describing the variant generation process is fully up to the course author. To make the workflow easier the generation script is supposed to be written in Scilab programming language thus making generation process and executing student code quite similar.

If no variant has been generated yet, the student receives personal unique variant. So viewing the same problem again and again will not generate more variants for particular student, but still, each student will have his own variant data. This is possible by storing variant data associated with particular student viewing the task. This will also allow the student to leave the page with current task and return to it later, even using another computer.

Student is supposed to create some artifact which represents his solution and submit it to the system. It may be source code written in Scilab language, which may be executed to reproduce some calculations, or models created by student and which are to be validated. Student has limited number of attempts and earns highest grade of all his submissions' grades.

As the student has his solution submitted he is waits until his solution is checked and validated. When the check is performed, he can see his final grade.

Course author, at his turn, provides data and configure module for students. The two most important aspects are task text, which is shown to students, and check instructions. Course author may also provide instructions to generate an individual variant for each student, though this feature is optional. Author can configure the module via special user interface. See table 1 for problem parameters.

Table 1. Problem parameters required to configure module

Parameter	Explanation
Display name	Each task should have a caption which will be used in statistical reports or other UI-elements
Queue name	The component uses XQueue as message broker, this parameter holds the identifier to the queue where all messages are stored
Attempts	The number of attempts allowed to submit solution
Weight	Weight of the task's grade
Instructor archive	File containing generation and check scenarios; a check scenario is mandatory and a generate scenario is optional; in case when the generate scenario is absent no variant generation is performed
Task text	Text showed to student when he opens the task page; it may contain substitution symbols to be templated with random parameters generated by generate script

The system requires special instructor file to be uploaded. This file is simple zip-archive which contains generation and check instructions. As the variant generation is optional, this archive may contain no generation instructions, so the module will not generate random data for students. Thus this module will not communicate with other subsystems to generate a variant, though check scenario is required anyway. Both check and generate scenarios are written in Scilab programming language to simplify the process.

4. IMPLEMENTATION

This paper describes an approach, where Scilab tool is used for assessments in online courses. While attending an online course on modeling basics, students are supposed to come through special assessments besides basic quizzes and tests. Students are to obtain personal task which is randomly generated for each attendee and later fulfill task providing their own solution using Scilab package. The solution may be represented by source code written in Scilab programming language or by a model made in XCos environment, part of Scilab package.

Latter the solution is checked by a special service called Scilab server, using rules provided by the course author. Scilab server is a special environment developed to handle Scilab-based assessments. This environment is an HTTP server that uses XQueue servers as message broker and performs execution of checking scripts.

Scilab server may be used for generating variants or checking solutions. To make server perform any action the message is required to be sent via a message broker. This message contains details on action to be performed with all supplementary details. For example, student script and checking instructions are necessary to assess the solution, and for generating individual variant generating instructions are needed. The message is string in json-format, which is passed via HTTP-request body. A javascript object encoded in this string should contain particular fields, for example, a field describing a method to execute, as “generate” or “check”, and submission identifier.

Scilab-server does not have ant pre-loaded instructions to generate a unique variant or check a users solution. This make the server independent from the course author. When the server receives a message to check or generate, it also receives instruction to perform action requested, so for generation it retrieves generation instructions right in the message, for check it receives student solution along with checking instructions provided by course author. Such implementation makes the server versatile, thus it can be used in different courses no matter of assessment types and aims.

There are two types of messages implemented in Scilab server for now: a message to generate an individual variant or a message to check user solution.

4.1 Unique Variant Generation

The generate-message contains data required to generate an individual variant. When generate-message is received by Scilab server, it launches a generate process which is illustrated in figure 1.

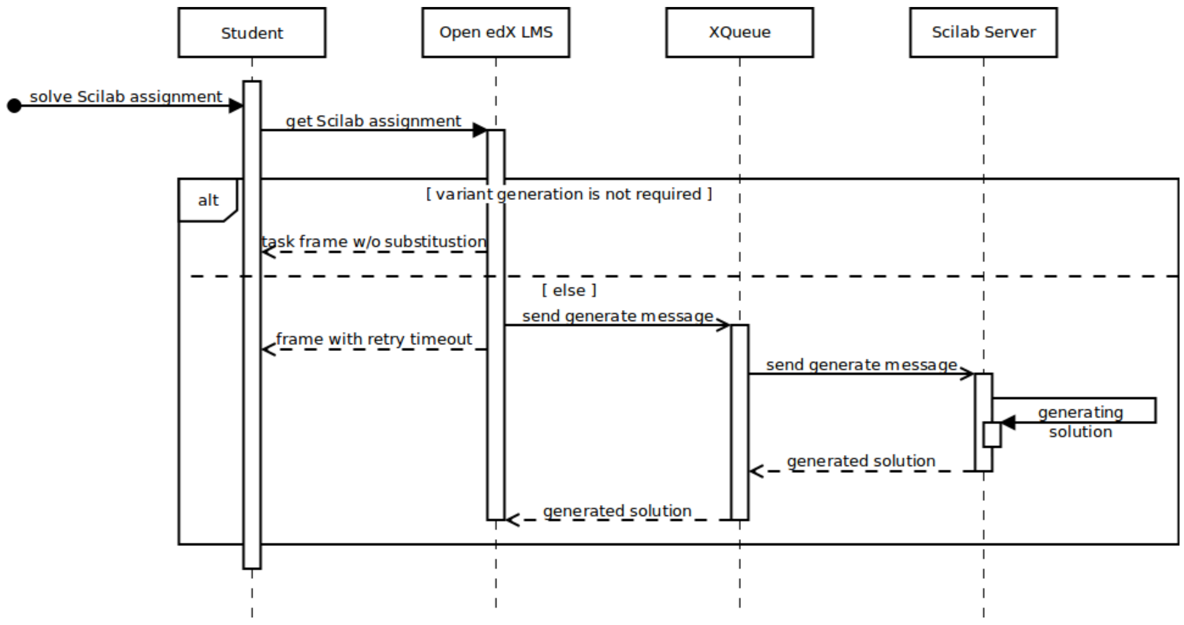

Figure 1. Unique variant generation sequence scheme

When the student opens a problem in LMS, the system decides whether the problem need a random variant to be generated or not. If not, the problem text is displayed as it-is without any template substitution routines. Otherwise the system generates a new variant for the student if no variant was generated before. Then the LMS system creates generate-message, which is sent to Scilab-server via XQueue.

First, the server extracts the contents of the instructions archive in a temporary directory on the server as the message is received. Then the Scilab server executes the generation scenario from the archive. The scenario name is always the same for any task. This scenario is provided by the course author and should produce an artifact: a text file with a variant data. This file is later used to display task data in LMS and to check a user solution. The generation itself is optional, so the Scilab server may never receive this type of message for specific tasks.

The generation artifact is a simple text file. It may contain several text lines. Each line represent a single parameter of the variant. For example, it may be random coefficients to solve an equation. The system does not care about the number of parameters, as it is an agreement between the generation script and the checking script. This text file will be passed as-is to the check script later. Also all the parameters listed in this file will be shown to the student in the problem text with the rules described below.

As the random variant is generated it is sent back to LMS with XQueue as a broker.

4.2 Problem Text Display

The problem text is a crucial element of a problem as it describes the guidelines to follow while solving the task. It is always up to the course author to give instructions which are comprehensible to students.

As far as the kind of problems proposed support parametrization with unique personal variant variables, the task text should support parametrization too. Thus a simple templating system is implemented.

The variant generation should produce a text file. It may contain any number of text lines where each line represent a single parameter. So, if the generate script creates a text file containing two lines, the subsystem decides that the random variant has two parameters.

When a student should sees a problem text, the subsystem substitutes all special character sequences “%s” with generated parameters. To be more precise, the first occurrence of “%s” in problem text will be substituted with first line of the pregenerated file, the second occurrence of "%s" will be substituted with the second line of the file, etc.

To avoid conflicts, in case when there were some errors with comparability, for example, when the task text was unintentionally changed or a wrong generation script was uploaded earlier, no substitution will be performed if the number of substitution symbols and the number of generated parameters do not match.

4.3 Student Solution Check

The check process is similar to the generation process but still has some differences. It is also initiated by the message received from a message broker, but has a different type descriptor. The message contains a check scenario that should determine a grade for a student solution. The check scenario is a Scilab script which uses unique variant data and a user solution. Though the generation is optional, the check scenario may use the student solution only. The whole check process is illustrated in figure 2.

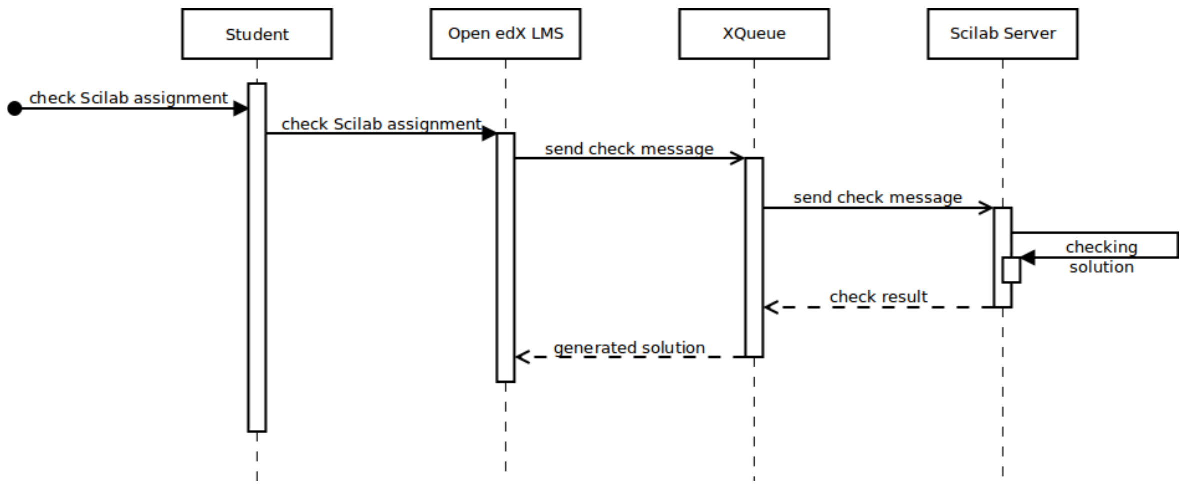

Figure 2. Solution check sequence scheme

When the student uploads his solutions, the checking process starts. LMS creates a message containing data describing type of the message and student solution to be assessed. This message is sent to the XQueue server and later to the Scilab-server.

When the check message is received the server extracts a user solution to temporary directory. If the student solution contains any executable Scilab script, it is executed, though the student solution may contain no scripts at all. That may happen when the task outcome is model created with XCos, then no user script is executed. After that Scilab server creates a file with pregenerated data, if this task needed generation, otherwise this step is skipped. Finally the check scenario is executed. It is written by the course author so it may optionally use pregenerated unique variant data and must validate the user submission. A grade allowed is a real number between 0 and 1.

As the server generates the answer message containing students grade and some additional data, for example, a feedback message, it is sent back to LMS via XQueue, in the reverse way as it was received.

4.4 Examples

For example, there is a task requiring a student to solve a linear equation ax+b=0. Each student is provided by randomly generated coefficients a and b. The task is to submit a file containing source code written in Scilab programming language that calculates variable x within specific precision and outputs this value into a file.

Generation instructions are shown below.

```
// Generation instructions
// Initialize random generator
seed = getdate("s");
rand("seed", seed);
// Generate individual variant
a = rand() * 2 - 1;
b = rand() * 2 - 1;
// Save variant
fd = mopen("./generate_result", "w");
mfprintf(fd, "%f\n%f", a, b);
mclose(fd);
```

The generation script randomly generates real numbers for a and b between -1 and 1 and outputs them into a file. The file created is latter used as variant data in the checking scenario or in LMS to display the task text to student.

A sample of the task text is shown below.

```
<p>Solve an equation [mathjaxinline]ax+b=0[/mathjaxinline], where:
[mathjaxinline]a = %s[/mathjaxinline], [mathjaxinline]b =
%s[/mathjaxinline]</p>
<p>Submit a <i>*.zip</i>-archive as an answer. It should contain file
<i>solution.sce</i> with Scilab source code. This script should create
file <i>output</i> in working directory. The file should contain saved
variable [mathjaxinline]x[/mathjaxinline].</p>
<p>To save data from Scilab script use <i>save</i> function.</p>
```

The task text contains substitution symbols %s which are replaced one by one using pregenerated variant data. There are two substitution parameters, so there are two parameters required to be generated with the generator script.

Check instructions are shown below.

```
// Check instructions
// Read data from individual variant
f = mopen('./generate_result);
[n, a, b] = mfscanf(f, "%f\n%f");
mclose(f);
// Load data generated by user solution
load('./output', 'x');
// Grade user solution
score = 0;
eps = 1e-5;
// Set full grade if user solution performed calculations
// within precision required
if abs(a * x + b) < eps
  score = 1;
end
// Save student score
write('./check_output', string(score))
```

The check scenario first reads pregenerated data provided by the generate scenario. There must be exactly two parameters, and we are sure that the file contains both of them. This file was previously generated by the generate script and it is created along with the check script as-is.

Then the check scenario reads the user output. The user scenario was executed earlier and it should have created output a file as the task text requires. The check script reads the value and compares it with the correct one. If the user's answer lies within given precision, the answer is treated as the correct one, and the student is awarded with the highest grade possible.

5. CONCLUSION

This paper describes an approach and implementation of automated assignments, which allow to check students' skill to perform calculation and modeling using Scilab computational package. The module developed was deployed to one of edX instance and is now successfully used in online courses.

This approach was used in two courses on National Platform of Open Education, the MOOC platform co-founded by 8 Russian universities. The courses are “Elements of control systems” and “Linear control systems”. Both use the developed module to assess students' learning results.

REFERENCES

Rodriguez, C., 2012. MOOCs and the AI-Stanford like courses: Two successful and distinct course formats for massive open online courses. *European Journal of Open, Distance and E-Learning*

Lisitsyna, L.S., Lyamin, A.V., 2014. Approach to Development of Effective E-Learning Courses. *Frontiers in Artificial Intelligence and Applications.* Vol. 262, pp. 732-738

Efimchik E.A., Lyamin A.V., Chezhin M.S., 2015. Automation of Variant Preparation and Solving Estimation of Algorithmic Tasks for Virtual Laboratories Based on Automata Model. *E-Learning, E-Education, and Online Training*

Lisitsyna L.S., Pershin A.A., Kazakov M.A., 2015. Game Mechanics Used for Achieving Better Results of Massive Online. *Smart Innovation, Systems and Technologies.* Vol. 41, pp. 183-193

Lyamin A.V., Vashenkov O.E., 2009. Virtual environment and instruments for student olympiad on cybernetics. *Proceedings of 8th IFAC Symposium on Advances in Control Education.* Kumamoto, Japan.

Belashenkova N.N., Cherepovskaya E.N., Lyamin A.V., Skshidlevsky A.A., 2015. Protection Methods of Assessment Procedures Used in e-Learning. *13th International Conference on Emerging eLearning Technologies and Applications*. P. 27-32.

Ivaniushin D.A., Lyamin A.V., Kopylov D.S., 2016. Assessment of Outcomes in Collaborative Project-Based Learning in Online Courses. *Smart Education and e-Learning 2016.* pp.351-361