2014 IEEE 5th Global Engineering Education Conference

Lightweight Ad Hoc Assessment of Practical Programming Skills at Scale

Thomas Staubitz, Jan Renz, Christian Willems, Johannes Jasper, Christoph Meinel
Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
{thomas.staubitz, jan.renz, christian.willems, meinel}@hpi.uni-potsdam.de
johannes.jasper@student.hpi.uni-potsdam.de

***Abstract*—There is a great demand for hands-on training in engineering education. In the context of a Massive Open Online Course (MOOC), assessing these experiments manually by teaching assistants is not possible owed to the high number of participants and the resulting workload for the teaching team. Systems for machine-based assessment of coding tasks are existing, but not necessarily available publicly, or not prepared to handle the massive amount of users in a MOOC. Definitely, they are not available "ad hoc", but require a certain amount of effort to be integrated in the MOOC platform or to be made available for the students in another way. Time and money to provide the required effort is not always available.**

This work presents a lightweight solution for the assessment of practical programming exercises, based on third party online coding tools. The solution was introduced as a part of openHPI's *Web-Technologies* course. The basic idea is to prepare a task in an available online tool, along with a piece of code that is able to evaluate the participant's solution. In case of success the participant is provided with a password, which in return serves as the answer for a fill-in-the-gap question in a standard quiz as provided by the openHPI MOOC platform, and thus allows for automatic online assessment based on practical coding exercises.

Keywords*—*MOOC, Hands-on Experience, Online Assessment, Javascript, HTML, Scalability

I. Introduction

openHPI is Germany's largest MOOC platform with a specialization in ICT engineering. Run by the Hasso Plattner Institute (HPI) in Potsdam, it has offered seven courses on various ICT topics since September 2012—hosting between 7,000 and 17,000 enrolled users per course.
This paper is mainly based on the experience that has been made while delivering the course *Web-Technologies*, which was offered in June and July 2013 as a follow-up to the course *Internetworking with TCP-IP*. 7,350 users were enrolled at the time of the final exam. 3,172 of these enrolled users were active participants. On openHPI active participants are defined as those users, who have submitted at least one discussion post or one homework [1]. 1,727 of those active participants were eligible for a graded certificate. Like all courses on openHPI, *Web-Technologies* followed a schema of a six-week course with several ungraded self-tests and one graded homework per week. According to this schema, the courses are concluded with a final exam, which also is graded. For each of these graded assignments a certain amount of points can be achieved by the participants. To be eligible for a graded certificate, a participant has to achieve at least 50% of the overall maximum course score [2]. As a part of this course, the teaching team offered HTML and Javascript tutorials of a more hands-on approach in form of screen casts with coding demonstrations and a corresponding transcript. The participation in these tutorials and exercises was optional. In this context, a new way of assessing the participants' learning progress, based on online execution of small coding tasks, using an existing external web tool, has been introduced. The motivation behind this decision will be discussed in section II. Furthermore, the methodology of this approach will be discussed in detail in section III. Section III-D will report about the experience that has been made with this model and some caveats that have to be considered. In section IV the impact that solving the practical assignments had on the users' results in the final exam will be analyzed. Finally, an outlook on the steps that are planned for the future will be given.

II. Assessment of Practical Assignments in a MOOC Context

A. Demand for Hands-on Exercises

Practical experiments are essential parts of on-campus courses. In an international survey amongst computer science academics—conducted in 2003—74% of the participants stated that they use practical work to assess their students [3]. Assessment for these experiments are essential, not only to motivate the students to work on these tasks, but even more to provide them with feedback if their effort was successful or not. Timely feedback is a very important mechanism to encourage participants to keep up their work, particularly if they encounter problems or do not find an instant solution [4].

Although practical assignments can be hard to implement for e-learning solutions and online courses, there is still a great demand. When the instructors of the first course on openHPI asked for missing features and content items, a remarkable number of users explicitly asked for practical experiments and exercises. The teaching team of the second course on "Internetworking" introduced a set of experimental hands-on exercises [5]. Even though, these exercises were voluntarily and resulted only in a small number of extra points (compared to the required effort), between 80 and 85 percent of the active participants took part in each of these exercises. The "Internetworking" course had a total of 2,761 active participants (out of 9,891 registered users at the time of the final exam). In a survey with more than 1,000 participants after the conclusion of that course, 4 out of 5 students evaluated the hands-on tasks as very useful (43.6%) or rather useful [5].

B. Assessing Practical Assignments at Large Scale

xMOOCs, as the openHPI courses, typically implement assessment based on computer-gradable quizzes, such as multiple-choice or fill-in-the-blank questions. This is a necessary limitation, since manual grading done by teaching assistants does not scale for a massive audience[1]. The need for *computer-gradable* exercises is a first challenge when enhancing MOOCs with practical assignments. The second challenge is posed by the requirements concerning *scaling*. Common solutions include running the experiments on the students' own computers or providing a cloud-driven environment. Yet, both approaches make auto-grading even harder. Another disadvantage of experiments on students' machines is the heterogeneous environment. Different operating systems or software stacks cause individual errors and problems. A comprehensive environment for hands-on assessment in the context of MOOCs should also respect the social character of this kind of courses, which makes the assessment prone to cheating since students could easily share results of practical tasks on social networks. During Coursera's Scala course, offered by Martin Odersky in 2012, this scenario became reality. The course staff felt the necessity to tackle this issue in an email announcement to all course participants [6]. For this reason, the practical exercises should be *customized* for each student. The paper at hand presents a lightweight and quick approach for the realization of practical assignments and therefore cannot match all the above requirements.

TABLE I. NUMBER OF HANDED IN SUBMISSIONS (FOR A MORE DETAILED DESCRIPTION OF THE EXERCISES SEE SECTION III)

Assignment	**# of submissions**	**success-rate**
Bonus 1: HTML: Add list entry	1,240	94%
Bonus 2: HTML: Edit class attributes	1,230	92%
Bonus 3: Javascript: max()	1,198	91%
Bonus 4: Javascript: square()	1,147	93%
Bonus 5: Javascript: sum()	1,108	85%
Bonus 6: Javascript: cookies	1,037	84%
Bonus 7: Javascript: isPrime()	953	85%
Bonus 8: Javascript: fibonacci()	920	85%
Total:	8,833	

During the last couple of years many efforts—such as e.g. WebToTeach, Ceilidh, Assyst, Web-Cat, or ELP to mention just a few—have been made to automate the evaluation of programming tasks. (see e.g. [7] for an overview of these efforts). Enabling automated grading of programming tasks in a MOOC system, such as openHPI, requires time, effort, and money. The two most important reasons why this was not an option for openHPI's *Web-Technologies* course were:

- The openHPI platform does neither provide an environment to run code, nor to assess it.
- The introduction of programming tasks into MOOCS still had an experimental character, lessons had to be learned before integrating such a feature in the platform.
- The development of a full featured online coding environment and its integration into the existing platform would have required a great deal of time. This time was not to be spent before the practicability of hands-on tasks in MOOCs was tested.

So, implementing a customized solution for the *Web-Technologies* course at openHPI—automated or peer-review-based—was out of question in terms of effort and timing. A leaner solution had to be found. Third-party, web-based coding tools recommended themselves as a quick and cheap alternative. Three general categories of available tools have been identified and will be discussed in section III-C.

TABLE II. NUMBER OF USERS THAT SUBMITTED A CERTAIN AMOUNT OF BONUS TASKS

# of submitted tasks	**# of users**
1	20
2	33
3	54
4	40
5	73
6	86
7	56
8	894
Total:	1,256

In the *Web-Technologies* course, an aggregate of 8,833 submissions (see table I for details) for all practical assignments were handed in by the participants. 1,256 participants (39.6%) out of 3,172 active participants submitted a solution for at least one of the practical bonus exercises. 894 (71.2%) out of these 1,256 participants submitted solutions for all eight bonus tasks (See table II). Given that a maximum of 16 points—which is equivalent to the amount of points for a single homework—for all eight bonus tasks was available, and that the participants had to invest considerably more time to earn these points compared to the homework, these numbers are quite good.

III. THE SOLUTION THROUGH EXECUTION ASSESSMENT PATTERN (STEAP)

The basic idea is to prepare a programming problem in a publicly available online tool, along with a piece of code that

[1]Peer reviewing is deliberately not taken into account here.

is able to evaluate the participant's solution. This evaluation code returns a *password* if the participant's solution provided the correct results. The participant can now copy this password and paste it in a fill-in-the-gap question in a standard quiz on the openHPI MOOC platform, which also serves as the starting point for the whole procedure (see figure 1).

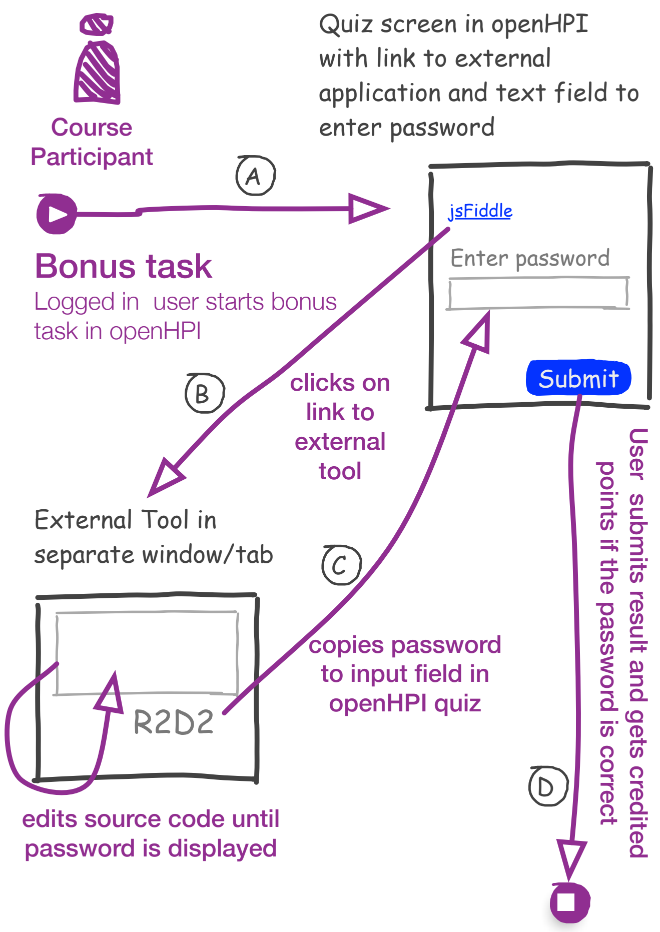

Course Participant
Quiz screen in openHPI with link to external application and text field to enter password
A
jsFiddle
Enter password
Submit
Bonus task
Logged in user starts bonus task in openHPI
B
clicks on link to external tool
External Tool in separate window/tab
C
copies password to input field in openHPI quiz
R2D2
edits source code until password is displayed
D
User submits result and gets credited points if the password is correct

Fig. 1. Schematic view of workflow in STEAP

A. *Implementation*

A simple STEAP task consists of three parts:

- a well-defined task or problem,
- the possibility to solve the given problem by editing and executing code in an instantly easily available online tool,
- a piece of code to evaluate the participant's solution.

1) Task: The participants were given several programming tasks of increasing difficulty in form of a quiz within the openHPI course platform. The task (see Fig. 2) described a problem that was to be solved as well as the structure of the solution.

2) Solution: The problem description contained a link to a JSFiddle page[2] that had been prepared for these purposes. There, the participants had the opportunity to write and test code to solve the given task in an editor within their web-browser (see Fig. 3-[1]).

3) Evaluation: The concept behind the STEAP tasks is that solving the task requires the student to modify and to execute the code—plain reading and understanding of the code is not sufficient. The participant is forced to do practical, hands-on exercises, experiment with the code snippets, and construct her expertise out of practical experience. The evaluation of the participants' solution is implemented through a series of test cases that ought to allow for all possibilities. This piece of code executes the provided solution and checks whether it behaves as expected. For the given example, the evaluation code called the function *isPrime* with exemplary numbers, each time checking if the participant's code solves the problem correctly. For the test numbers 2, 3, 5 and 7 the *isPrime* function was supposed to say *true*. For the numbers 4, 9 and 15, however, it was supposed to say *false* as these are not prime numbers. As unexperienced programmers often do not check certain edge cases, the evaluation code also checked the numbers 0 and 1, which have to be treated separately in the solution code. If all the test numbers returned the correct result, the password was released. On the other hand, if one or more test cases fail—the *isPrime* function returns an unexpected result—the password stays hidden. There were no restrictions regarding the number of attempts. Participants can run their code as often as required and refine it if necessary. Note, however, that the quiz on openHPI has a time restriction. Only within this period—usually one hour—is the participant able to enter the password in the quiz form in order to gain credit for the exercise.

In order to prevent the participants from only covering the test cases, the evaluation code is obfuscated[3] (see Listing 3-[2]). Note that obfuscated code can be restored, provided a certain effort (see section III-D for a discussion of this aspect). If the participant has solved the task, she is rewarded with the display of a password, which in turn is the solution for the quiz that is containing the task description from step 1. Obviously, this kind of evaluation is binary. Either the participant is able to retrieve the password or not. There is no way to provide points for partial solutions. This aspect will be discussed in more detail later.

Assignments of the more cognitive reading and understanding kind have also been provided in homework and the final exam. The results of two of these assignments have been used to

[2]What JSFiddle is and why it was chosen by the teaching team, will be described in more detail in section III-C. The example given in Fig. 3 can be found online at http://jsfiddle.net/openHPI/nFxVh/

[3]Obfuscated code is still functional and can be executed by a computer. It obscures its functionality to a human reader by using strong compression and complex syntax.

evaluate the impact of the practical programming tasks on the performance of the participants (see section IV).

> Given is the following website, which provides some Javascript code and two HTML form fields on JSFiddle: http://jsfiddle.net/openHPI/nFxVh/
> Write a function *isPrime(num)*, which takes a number *num* and checks if that number is a prime number. The function is expected to return *true* if *num* is a prime number and *false* if not. [...] When you have found a correct solution, a password will be displayed in JSFiddle's result window, which you can enter into the input field below this text.

Fig. 2. Exemplary Javascript programming task in the openHPI course *Web-Technologies* (translation from German original).

```
function isPrime(num) {
   //enter your code here [1]
}

//Obscured evaluation code [2]
eval(function(p,a,c,k,e,r){e=function(c){return
c.toString(a)};if(!''.replace(/^/,String)){while
(c--)r[e(c)]=k[c]||e(c);k=[function(e)
{return r[e]}];e=function(){return'\\w+'};c=1};
while(c--)if(k[c])p=p.replace(new RegExp
('\\b'+e(c)+'\\b','g'),k[c]);return p}('k m()
{8 a=6(2)&&6(3)&&6(5)&&6(7)&&!6(4)&&!6(9)&&
!6(r)&&!6(0)&&!6(1);8 b="l";b="j"+b;8 c=n.
o(\'p\');c.q=(a?"d f g h iöe, s t u "+b:"d f g h v
w iöe")}',33,33,'||||||isPrime||var|||||
Sie|st|haben|die|Aufgabe|gel|BK1k|function|ad87|
check|document|getElementById|solution|innerHTML|
15|das|Passwort|ist|noch|nicht'.split('|'),0,{}))
```

Fig. 3. Javascript coding task (1) and obfuscated evaluation code (2).

B. Advantages of the STEAP Methodology

STEAP enables teachers to easily employ available, free online tools for the purpose of evaluating practical assignments. It can be used in combination with any system that is managing learners. Be it a classical Learning Management System, such as e.g. Moodle or Canvas, or a system that is specialized to manage MOOCS such as e.g. openHPI or edX. The only requirement is that this system provides the possibility to edit automatically graded simple multiple-choice or fill-in-the-gap questions. The methodology—or at least the lightweight implementation presented in this work—should not be employed where fraud resistance plays an important role, however (see section III-D). STEAP enables students to test their practical skills in an environment that has been prepared by the teacher according to the special demands of the given task. Basic, immediate, automated feedback is given to the student. This mechanism needs to be improved in the future, however, to provide a more detailed feedback (see also section III-D). STEAP has its limits in this regard as any feedback needs to be specified in the task itself, and therefore, will cause a high workload for the teachers if a very detailed feedback is required unless there is a community that can help out the learner, as given in a MOOC context.

C. Categories of Available Online Tools for Practical Assignments

We have identified three general categories of available tools that more or less fit the purposes of the teaching team.

1) Full-featured Online Courses: Tools in this category, offer complete courses to certain hands-on programming topics. Including text- or video-based explanations, a predefined set of exercises and means to solve these exercises online. CodeAcademy[4] is an example for a tool in this category.

2) Online Tools with a Fixed Problem Set: Tools in this category typically provide a fixed set of problems, which are evaluated by the tool one way or the other, but no additional course-like material. CodingBat[5] is an example for a tool in this category. It offers an impressive list of problems in Java and in Python. The CodingBat evaluates the results based on the results of predefined unit tests. CodeLab[6], the commercial follow up project of WebToTeach [8] is an other tool in this category.

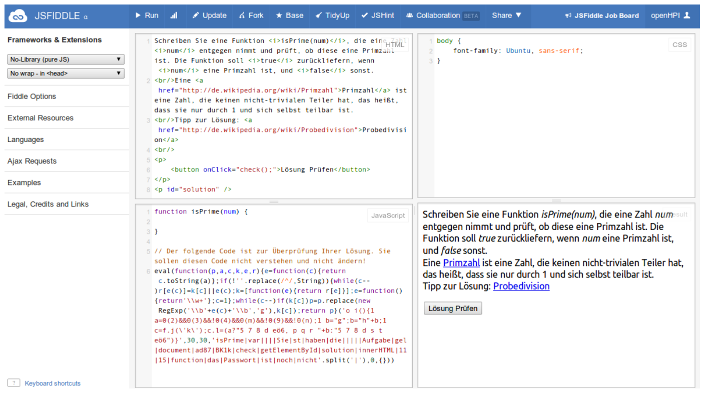
3) Free-form Coding Tools: Finally, tools in this category offer a means to write and run code. There is neither a given problem set nor a course structure attached.

Tools from the first two categories are not suited for the intended purpose. A common problem with all these tools was, that the teaching team had no influence on the problem set. There was no way to inject the evaluation code. These tools provide evaluation mechanisms of their own, which do not provide a means to return the results to the openHPI platform. A possible solution for this issue is the Learning Tools Interoperability (LTI) specification by the IMS Global Learning Consortium, which also signs responsible for e.g. the Common Cartridge (CC) Format. LTI allows teaching/learning tools to interact with systems that are, amongst other duties, managing learners, such as Learning Management Systems or MOOCs. This idea will be taken up again in section V.

Employing a tool that is able to give a more specific feedback to the users, such as CodingBat or WebToTeach, would be desirable. Unfortunately, in the given setting, the usage of such a tool was not manageable for the intended purposes. The teaching team decided in favor of JSFiddle[7], a tool of the third category—free-form coding tools. JSFiddle is a tool that allows participants to quickly and easily write HTML, CSS, and Javascript code online, view and share the results and keep a version-controlled

[4]see http://www.codecademy.com/
[5]see http://codingbat.com/
[6]see http://www.turingscraft.com/
[7]see http://jsfiddle.net

history of their steps. Besides Javascript and CSS it also supports CoffeeScript and SCSS. A broad variety of Javascript libraries, such as jQuery[8] or MooTools[9] can be used. Within the web community JSFiddle has a high reputation and is one of the 2,000 most frequently used websites [9]. Its main window is separated into four areas, three of them serve the purpose of editing HTML, Javascript, and CSS code, while the fourth displays the rendered result when the user finishes editing and clicks the *Run* button (see Fig. 4).



No-Library (pure JS)
No wrap - in <head>

```html
Schreiben Sie eine Funktion <i>isPrime(num)</i>, die eine Zahl
<i>num</i> entgegen nimmt und prüft, ob diese eine Primzahl
ist. Die Funktion soll <i>true</i> zurückliefern, wenn
 <i>num</i> eine Primzahl ist, und <i>false</i> sonst.
<br/>Eine <a
 href="http://de.wikipedia.org/wiki/Primzahl">Primzahl</a> ist
eine Zahl, die keinen nicht-trivialen Teiler hat, das heißt,
dass sie nur durch 1 und sich selbst teilbar ist.
<br/>Tipp zur Lösung: <a
 href="http://de.wikipedia.org/wiki/Probedivision">Probedivisi
on</a>
<br/>
<p>
    <button onClick="check();">Lösung Prüfen</button>
</p>
<p id="solution" />
```

```css
body {
    font-family: Ubuntu, sans-serif;
}
```

```javascript
function isPrime(num) {

}

// Der folgende Code ist zur Überprüfung Ihrer Lösung. Sie
sollen diesen Code nicht verstehen und nicht ändern!
eval(function(p,a,c,k,e,r){e=function(c){return
 c.toString(a)};if(!''.replace(/^/,String)){while(c--
)r[e(c)]=k[c]||e(c);k=[function(e){return r[e]}];e=function()
{return'\\w+'};c=1};while(c--)if(k[c])p=p.replace(new
 RegExp('\\b'+e(c)+'\\b','g'),k[c]);return p}('o i(){1
a=0(2)&&0(3)&&!0(4)&&0(m)&&!0(9)&&!0(n);1 b="g";b="h"+b;1
c=f.j(\'k\');c.l=(a?"5 7 8 d eö6, p q r "+b:"5 7 8 d s t
eö6")}',30,30,'isPrime|var||||Sie|st|haben|die|||||Aufgabe|gel
|document|ad87|BK1k|check|getElementById|solution|innerHTML|11
|15|function|das|Passwort|ist|noch|nicht'.split('|'),0,{}))
```

Schreiben Sie eine Funktion *isPrime(num)*, die eine Zahl *num* entgegen nimmt und prüft, ob diese eine Primzahl ist. Die Funktion soll *true* zurückliefern, wenn *num* eine Primzahl ist, und *false* sonst.
Eine Primzahl ist eine Zahl, die keinen nicht-trivialen Teiler hat, das heißt, dass sie nur durch 1 und sich selbst teilbar ist.
Tipp zur Lösung: Probedivision

Fig. 4. Screenshot of the JSFiddle programming environment. The left column allows for settings of the used software libraries. The upper-left window is used to edit HTML code, in the upper-right window CSS rules can be defined. Javascript code can be edited in the lower-left window. The final result is displayed in the lower-right window.

Hereby, the practical programming exercises actually run on the students' computers—not causing any performance bottleneck on the server—the training environment, however, is still homogeneous due to the mandatory use of a 3rd party, web-based tool.

D. Lessons learned

1) Version history in JSFiddle: One of the problems that the teaching team faced was that JSFiddle automatically stores the version history of a project. JSFiddle provides the user with an *Update* and a *Run* button. A click on the *Update* button is not required for solving the task—a click on the *Run* button would be sufficient to unveil the password if the solution has been found, but it can also not be avoided, as the teaching team obviously had no influence on a third party tool's GUI. The history of a JSFiddle project contains all updates of all users. Different versions can be accessed by simply changing the version number in the URL. The version number is appended to the identification of the code snippet. The original code from the example above had the URL http://jsfiddle.net/openHPI/nFxVh/**1**/, a later version would have the URL http://jsfiddle.net/openHPI/nFxVh/**2**/. The teaching team made use of this feature in one very special case of an advanced object-oriented Javascript exercise. In this exercise where participants could not earn points at all, the history was used to provide them with hints to the solution. Generally, however, this is not what you want in such a context, as each update operation of a participant creates a new version of the task, which then also can be accessed by every other participant—if they experiment hard enough with the URLs.

2) (Missing) Internationalization of external tools: As the Web-Technologies course was offered in German, JSFiddle's missing internationalization was another issue that had to be faced. In the context of a dedicated programming course, this does not necessarily pose a problem as a certain proficiency in English is required anyway in this field. In the given context of an introductory course on a variety of web-technologies, some of the participants had minor problems with that. This is not considered to be a major problem in the context of programming, however.

3) Availability of external tools: More seriously was the fact that some participants reported problems with JSFiddle's availability. Some spot checks by the teaching team confirmed these reports. Especially, in close view of an approaching deadline this turned out to be problematic. In this case, it can be taken for granted that it was not caused by load peaks on our side. The maximum number of participants working in parallel on the hands-on tasks was 314[10]. If the number of participants working in parallel on hands-on tasks is expected to be higher, this is a factor that needs to be considered. Contacting the provider of the chosen tool and discussing the plan might be a good idea as not every tool that is available online is necessarily designed to handle large amounts of users.

4) Browser compatibility: When third party tools are used in the context of the WWW, browser compatibility always is an issue to be considered. Nevertheless, jsFiddle has turned out to be a tool that is compatible with a wide variety of browsers. Obviously, Javascript needs to be enabled in the user's browser and certain browser plug-ins, such as e.g. NoScript or Internet Security tool suites need to be disabled in this context. Content authors that aim to incorporate "fancy" or exotic libraries into their experiment should test the setup with the 3rd-party tool extensively.

5) Fraud resistance: The solution presented in this work is not very fraud resistant. As shown above, participants were easily able to find the results of other participants with a little effort. Unsurprisingly, participants with a more advanced knowledge of Javascript also have been able to de-obfuscate the evaluation code and come into possession of the solution passwords[11]. Sharing results via social networks, also was easy to do as all participants worked on identical problems. It was decided to ignore these issues for the time being, as

- it was assumed that participants were more interested

[8]see http://jquery.com
[9]see http://mootools.net/

[10]This is the number of participants that started to work on the task on openHPI within 1 hour. We had no way to measure how many users were actually using JSFiddle during this time period
[11]see e.g. the following discussion (in German language) on openHPI: https://openhpi.de/courses/7/discussion_topics/3205

in completing a challenge and learning rather than in earning points[12],

- participants were only able to earn a few bonus points,
- participants with the technical expertise to restore the original evaluation code would also be skilled enough to solve the questions at hand.

6) Binary assessment: This type of assignment allows only a binary assessment of the assignments. Either it works, and the participant can retrieve the password, or it does not and she cannot retrieve the password. Neither is it possible to grant points for partially correct solutions nor is it possible to assess the quality of the found solution.

E. Transferability

1) Transferring the programming exercises to other programming languages: The chosen approach is based on two major features, namely the opportunity to edit and run code in a homogeneous online environment. This means that participants are not required to install any programming tools on their local machines. This aspect becomes increasingly important when considering other programming languages as many of these require a great deal of configuration. The basic concept of editing the code and presenting the results online is transferable to any programming language. Also running the code on the server is possible, as any soft- and hardware can be simulated in virtual environments. Websites with online editors and compilers already exist[13] and could be used for the STEAP approach. Note, however, that in contrast to Javascript, most programming languages cannot be interpreted by the users' browsers, but have to be *compiled* on the server. The created programs would therefore run on that server and use its resources (e.g. memory or processing time). As discussed in section III-D, the availability of such online services could be affected in times of high usage.

Another means to ensure the homogeneity of the coding environment is the use of virtual machines. Such machines can be pre-configured by the teaching team to meet the students' needs and provide any necessary tool for the task at hand. In contrast to server sided evaluation, however, virtual machines are downloaded and run on the users' local machines, thus preventing availability problems.

2) Transferring the methodology to be used for non-programming tasks: The concept of STEAP—provide a task in a third party environment and test the result with a password—is well applicable not only to programming challenges.

[12]Even considered that the certificate that is issued by openHPI does not have an “official value", such as ECTS points, etc. this admittedly turned out to be a rather starry-eyed assumption. Concluding from often arising discussions about points in the forums or the help desk, points are valued rather high amongst certain subsets of our user community.

[13]see e.g. compileonline.com/

In the described openHPI course, a topic in the lectures was HTTP[14]. In order to provide a deeper understanding, the teaching team decided to provide practical exercises on HTTP headers. A HTTP header is meta information that is sent along every HTTP request to web servers. In tutorials accompanying the course, we taught how to manipulate HTTP headers and what effects this has. The HPI-owned site tele-task.de was prepared to react to the word “openHPI” in the participant's *user-agent*, which is a HTTP header designed to provide information about the web-browser in use (e.g. indicating a mobile operating system). If a course participant accessed the tele-TASK website with such a manipulated user-agent string, the site issued a password that should be used to answer a related question in the openHPI quiz environment.

In order to design a task that follows the STEAP concept, it must only produce an unambiguous result—not necessarily electronically—that can be entered into a quiz form. Other examples for hands-on exercises conducted with the STEAP pattern are described in [5]. The course participants had to understand headers of e-mails, the responses on a query to a DNS server, or analyze network traffic on their own computer.

IV. Evaluation and Feedback

Within the *Web-Technologies* course, week four was dealing with the topic web programming. Comparable questions in the homework of week four and the final exam have been selected to evaluate if working on the STEAP tasks had any influence on the users' performance. Week four's homework had to be submitted before the STEAP tasks were available. The final exam was available after the STEAP tasks had to be submitted. Fig. 5 and 6 show the questions that have been compared.

Our assumption was, that those users who worked their way through all the STEAP tasks would have better results for the monitored question in the final exam—compared to their colleagues who did not work on the STEAP tasks and/or compared to the monitored control question in the homework of week four.
To prove this assumption the amount of points that the users achieved for these questions have been compared.

The following tables show the results of this evaluation. Only the results of those users that submitted both—homework of week four and the final exam—have been considered. In the following, this subset of participants will be called the *test group*. The columns in all following tables are to be read as:

- **col1**: The number of submitted STEAP tasks, per user in col2 to col4,
- **col2**: The number of users who sported better results for the questions in the final exam than for the equivalent questions in the homework of week four. The number in parentheses shows the percentage of the number of

[14]The Hypertext Transfer Protocol is used for requesting web-sites from web servers.

users in this column vs. the sum of all columns in this row,

- **col3**: The number of users whose results for the equivalent questions in homework four and final exam were the same,
- **col4**: The number of users whose results in the final exam were worse, compared to their results for the equivalent questions in homework four.

As the vast majority of the participants in the test group have taken either no bonus task or all bonus tasks we aggregated this value in two groups:

- Participants who submitted less than four bonus tasks (control group),
- participants who submitted four or more bonus tasks (experimental group).

We started by comparing all the users in the test group. Quite contrary to our assumptions, an even larger percentage of participants that did not work on any STEAP tasks improved their performance than those who did.

TABLE III. COMPARISON OF THE RESULTS OF ALL PARTICIPANTS IN THE TEST GROUP

col1	col2	col3	col4
< 4	246 (36%)	341 (49%)	105 (15%)
≥ 4	265 (25%)	705 (65%)	109 (10%)

Then, to figure out if the participants background in IT had any impact on the results, we evaluated the participants separated by this value. Background in IT is a profile setting on openHPI, which optionally can be specified by the participants. The options for this setting are:

- Undefined, None, Beginner, Advanced, and Expert.

The groups *Undefined* and *None* have been merged as the second group was of negligible size.

Tables IV to VII show the results of the users in the test group separated by their background in IT.

TABLE IV. COMPARISON OF THE RESULTS OF THOSE PARTICIPANTS IN THE TEST GROUP WITH AN UNDEFINED OR NO BACKGROUND IN IT.

col1	col2	col3	col4
< 4	48 (33%)	79 (54%)	20 (14%)
≥ 4	57 (22%)	171(66%)	30 (12%)

TABLE V. COMPARISON OF THE RESULTS OF THOSE PARTICIPANTS IN THE TEST GROUP WITH A BEGINNER'S BACKGROUND IN IT.

col1	col2	col3	col4
< 4	62 (38%)	68 (42%)	32 (20%)
≥ 4	63 (37%)	84 (49%)	25 (15%)

TABLE VI. COMPARISON OF THE RESULTS OF THOSE PARTICIPANTS IN THE TEST GROUP WITH AN ADVANCED BACKGROUND IN IT.

col1	col2	col3	col4
< 4	111 (40%)	130 (46%)	40 (14%)
≥ 4	111 (24%)	308 (67%)	43 (9%)

TABLE VII. COMPARISON OF THE RESULTS OF THOSE PARTICIPANTS IN THE TEST GROUP WITH AN EXPERT BACKGROUND IN IT.

col1	col2	col3	col4
< 4	25 (25%)	64 (63%)	13 (13%)
≥ 4	34 (18%)	144 (76%)	11 (6%)

The most significant observation here was that the majority of users, regardless of their background, achieved the same amount of points in the final exam as in the homework of week four. Therefore, as a final step, we removed those users from the evaluation that had achieved the full amount of points for the probed question in homework four yet, again distinguished by the users' background in IT.
The consideration behind this move was that those who earned the full amount of points for the first monitored question yet, had no headroom to improve their results anyway and, therefore, rather distorted the result. Tables VIII to XI show the results of this operation. While the absolute numbers still do not differ too much between those users who have taken the STEAP tasks and those who did not, the percentages provide a more optimistic view here.

TABLE VIII. COMPARISON OF THE RESULTS OF THOSE PARTICIPANTS IN THE TEST GROUP WHO HAD NOT ACHIEVED THE POSSIBLE AMOUNT OF POINTS FOR THE REVIEWED QUESTION IN HOMEWORK FOUR WITH AN UNDEFINED OR NO BACKGROUND IN IT.

col1	col2	col3	col4
< 4	48 (61%)	18 (23%)	13 (16%)
≥ 4	57 (73%)	9 (12%)	12 (15%)

TABLE IX. COMPARISON OF THE RESULTS OF THOSE PARTICIPANTS IN THE TEST GROUP WHO HAD NOT ACHIEVED THE POSSIBLE AMOUNT OF POINTS FOR THE REVIEWED QUESTION IN HOMEWORK FOUR WITH A BEGINNER'S BACKGROUND IN IT.

col1	col2	col3	col4
< 4	62 (59%)	23 (22%)	20 (19%)
≥ 4	63 (72%)	10 (11%)	14 (16%)

TABLE X. COMPARISON OF THE RESULTS OF THOSE PARTICIPANTS IN THE TEST GROUP WHO HAD NOT ACHIEVED THE POSSIBLE AMOUNT OF POINTS FOR THE REVIEWED QUESTION IN HOMEWORK FOUR WITH AN ADVANCED BACKGROUND IN IT.

col1	col2	col3	col4
< 4	111 (74%)	18 (12%)	22 (15%)
≥ 4	111 (80%)	13 (9%)	15 (11%)

These data are not supporting our assumption very strongly. Obviously, it cannot be expected that participants get more than

TABLE XI. COMPARISON OF THE RESULTS OF THOSE PARTICIPANTS IN THE TEST GROUP WHO HAD NOT ACHIEVED THE POSSIBLE AMOUNT OF POINTS FOR THE REVIEWED QUESTION IN HOMEWORK FOUR WITH AN EXPERT BACKGROUND IN IT.

col1	col2	col3	col4
< 4	25 (69%)	3 (8%)	8 (22%)
≥ 4	34 (87%)	2 (5%)	3 (8%)

a very brief introduction to the world of programming in a one week crash course, covering about an hour of video and a handful of simple programming tasks. We did not conduct a formal survey to evaluate the participants' opinion about these tasks, but in a discussion category in the course's forum which explicitly broached the issues of bonus material and the STEAP tasks, the majority of users reacted very positively[15]

V. Outlook on Future Development

For the coming new openHPI platform better integrated tools, compatible with the Learning Tools Interoperability (LTI) specification are in the pipeline—to enable more complex programming tasks, better feedback mechanisms, and better fraud protection. LTI enables the communication between systems that, amongst others, manage users, such as LMS or MOOCs via web services. The principle is more or less an automated version of our workflow. The MOOC or LMS provides a link to the connected tool. When the link is clicked, the user is sent to the external learning tool, works on a task there and, on submit, the result—a value between zero and one—is returned to the MOOC, where it is added to the users progress. This solution affords, however, that both tools are compatible with the LTI specification and support its feature set.

For the more distant future, research on non-binary assessment of programming tasks, based on software metrics and unit tests, is planned.

VI. Conclusion

In terms of cost-value ratio the STEAP Experiment proved to be a success under the given circumstances. As these circumstances are quite common, we are sure that our experiments might prove to be valuable for others as well. The improvised connection to a tool, such as JSFiddle, via the generation of passwords is easy to setup and very cheap to implement. If the teaching team is aware of several pitfalls, its usage is overall rather uncomplicated. The possibilities are very limited, however. Although we were not able to show a significant difference in the performance of those users that submitted solutions for the STEAP tasks, we are still convinced of the importance to provide more practical exercises for MOOCs. The generally positive feedback that we received from the participants supports this assumption.

[15] see e.g. https://openhpi.de/courses/7/discussion_topics/3154 or https://openhpi.de/courses/7/discussion_topics/3217

Given the following HTML and Javascript:

```
<!DOCTYPE html>
<html>
 <head>
  <title>Wieder eine knifflige Aufgabe</title>
  <script type="text/javascript">
   var d = document;
   function task1() {
    var one = d.getElementById("Eins");
    var two = d.getElementById("Zwei");
    if (one.style.display == "block") {
     one.style.display = "none";
     two.style.display = "block";
    } else {
     two.style.display = "none";
     one.style.display = "block";
    }
   }
   function task2() {
    var teleboard = d.getElementsByTagName("div")[0];
    var newTitle = d.createElement("h1");
    var newText = d.createTextNode("TeleBoard");
    newTitle.appendChild(newText);
    teleboard.appendChild(newTitle);
   }
  </script>
 </head>
 <body>
  <div id="Eins" class="tele" style="display:block">
    Eins
    <a href="http://tele-board.de">Eins</a>
  </div>
  <div id="Zwei" class="task" style="display:none">
   Zwei
   <a href="http://tele-task.de">Zwei</a>
  </div>
  <p><a href="#" onClick="task1();">Aufgabe1</a>
  <p><a href="#" onClick="task2();">Aufgabe2</a>
 </body>
</html>
```

Read the source code and try to determine what behavior will be triggered by the javascript functions task1 and task2. Please, assign the events (A) to (F) to the first two clicks on the links *Aufgabe1* and *Aufgabe2*

- (A) The <div>-element with id="Eins" becomes visible.
- (B) The <div>-element with id="Zwei" becomes visible.
- (C) An <h1>-element will be added to the <div>-element with id="1".
- (D) An <h1>-element will be added to the <div>-element with id="1" only if this <div>-element is currently visible.
- (E) A second <h1>-Element will be attached to the <div>-element with id="1".
- (F) Nothing will happen as an <h1>-element has been attached yet.

Fig. 5. Question in week four's homework of the openHPI course *Web-Technologies* (translation from German original).

Given the following HTML and Javascript:

```
<!DOCTYPE html>
<html>
 <head>
  <title>Schon wieder eine knifflige Aufgabe</title>
  <script type="text/javascript">
   function task1() {
    var one = document.getElementById("Eins");
    if (one.childNodes[0].style.color == "red") {
     one.childNodes[0].style.color = "green";
    } else {
     one.childNodes[0].style.color = "red";
    }
   }

   function task2() {
    var two = document.getElementById("Zwei");
    for (var i = 0; i < two.childNodes.length; i++) {
     if (two.childNodes[i].style) {
      two.childNodes[i].style.color = "red";
     }
    }
   }
  </script>
 </head>
 <body>
  <div id="Eins" class="tele"><a href="http://tele-task.de"
         style="color:red">Eins</a>
  </div>
  <ul id="Zwei">
   <li>Erster</li>
   <li>Zweiter</li>
  </ul>
  <p><a href="#" onClick="task1();">Aufgabe1</a>
  <p><a href="#" onClick="task2();">Aufgabe2</a>
 </body>
</html>
```

Which of the following statements apply for the given code example?

- Clicks on the link *Aufgabe1* color the link in the <div>-element with id="Eins" red or green in turn.
- Clicking on the link *Aufgabe1* has no effect, as javascripts cannot be called via event-handlers, such as *onClick*. To do this an <input type="button" /> is required.
- A click on link *Aufgabe2* colors all list elements in the list with id="Zwei" red.
- Clicks on link *Aufgabe2* color all list elements in the list with id="Zwei" red and green in turn.

Fig. 6. Question in the final exam of the openHPI course *Web-Technologies* (translation from German original).

References

[1] C. Willems, J. Renz, T. Staubitz, and C. Meinel, "Reflections on enrollment numbers and success rates at the openhpi mooc platform," in *Proceedings of the European MOOC Stakeholder Summit 2014*, PAU Education, 2014, pp. 101–106.

[2] C. Meinel and C. Willems, *OpenHPI : The MOOC offer at Hasso Plattner Institute, Hasso-Plattner-Institut für Softwaresystemtechnik: Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam ; Nr. 80*. Potsdam, Germany: University Press, 2013.

[3] J. Carter, K. Ala-Mutka, U. Fuller, M. Dick, J. English, W. Fone, and J. Sheard, "How shall we assess this?" *SIGCSE Bull.*, vol. 35, no. 4, pp. 107–123, Jun. 2003, ISSN: 0097-8418.

[4] D. M. Teague, A. Poppleton, P. G. Bancroft, and P. Roe, "Online feedback for novice programmers," in *OLT 2005 Conference: Beyond Delivery*, QUT Gardens Point, Brisbane, 2005.

[5] C. Willems, J. Jasper, and C. Meinel, "Introducing hands-on experience to a massive open online course on openhpi," in *Proceedings of IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE 2013)*, Kuta, Bali, Indonesia: IEEE Press, 2013.

[6] private communication, [Announcement sent by Scala Course Staff, Scala course on Coursera], 2012.

[7] K. M. Ala-Mutka, "A survey of automated assessment approaches for programming assignments," *Computer Science Education*, vol. 15, no. 2, pp. 83–102, 2005.

[8] D. Arnow and O. Barshay, "Webtoteach: An interactive focused programming exercise system," in *Frontiers in Education Conference, 1999. FIE '99. 29th Annual*, vol. 1, 1999, 12A9/39–12A9/44 vol.1.

[9] http://www.alexa.com/siteinfo/jsfiddle.net, [Online; accessed 26-09-2013].