12-31-2019

PERSONALIZED COMPUTER SECURITY TASKS WITH AUTOMATIC EVALUATION AND FEEDBACK

Isaac Agudo
Computer Science Department, University of Malaga, isaac@lcc.uma.es

Ruben Rios
Computer Science Department, University of Malaga, ruben@lcc.uma.es

Ana Nieto
Computer Science Department, University of Malaga, nieto@lcc.uma.es

PERSONALIZED COMPUTER SECURITY TASKS WITH AUTOMATIC EVALUATION AND FEEDBACK

Isaac Agudo
Computer Science Department
University of Malaga
isaac@lcc.uma.es

Ruben Rios
Computer Science Department
University of Malaga
ruben@lcc.uma.es

Ana Nieto
Computer Science Department
University of Malaga
nieto@lcc.uma.es

Abstract:

Teaching cybersecurity is very challenging as it requires coverage of many different technologies, protocols, tools, and systems. This paper presents SERA, a modular and highly customizable framework specifically designed for the definition of cybersecurity exercises with automatic evaluation and feedback. The framework defines a simple yet powerful language for the specification of lab exercises, which are instantiated at runtime into personalized tasks for each student. The language supports the definition of checks that should be passed for considering that the exercise is complete. Moreover, it is possible to associate some feedback to the checks with the goal of guiding the student towards the solution in case it is necessary, thus promoting self-learning and students' autonomy. SERA also keeps information regarding students' submissions in order to track students' progress and identify potential difficulties and limitations. Finally, the SERA framework has been integrated with the Moodle learning platform and is currently under beta testing in several computer security courses at the University of Malaga.

Keywords: evaluation, feedback, learn by doing, self-learning, autonomy, Moodle

I. INTRODUCTION

Teachers around the world are in a constant search for new and more dynamic methodologies that make their students more involved in their own learning process. Essentially, the idea is to guide teaching to action [Bot et al., 2007] rather than taking the traditional approach of non-interactive master classes where students become mere consumers of information.

Among the various methodologies that have been developed for enabling the active participation of the students, it is worth mentioning problem-based learning as well as flipped classrooms [Clark et al, 2018]. These methodologies are especially relevant in the context of computer science and engineering, where it is paramount that students face real-world problems and find solutions autonomously. It is precisely in this area of knowledge were the authors of this paper teach most of their classes.

Unfortunately, although these methodologies are undoubtedly beneficial for the student, they imply a considerable amount of effort for the teacher, who must be constantly preparing and correcting exercises so that the students receive timely feedback, which is considered a crucial factor in the learning process [Hattie, 2007]. In addition, the teacher must be available, in the classroom or remotely, whenever a student is stuck with an exercise because they may lose interest if guidance or feedback is not immediate. This highly demanding and time-consuming task puts a lot of stress on the teacher and is only feasible when the number of students in class is relatively small.

We have been working for the last few years on the development of some ad-hoc solutions to automate the evaluation of the most recurring lab exercises. We found out that (i) it was not easy to integrate all our developments with the learning management system at the University of Malaga, currently based on Moodle, and also that (ii) the lab exercises were barely reusable as they were

specifically designed to meet a very concrete goal. Consequently, it was necessary to find a solution that could be integrated with Moodle and, at the same time, was highly configurable for as many types of lab exercises as possible.

After reviewing the literature on automatic evaluation of lab exercises, we realized that most of the tools available were devoted to the evaluation of programming assignments (see [Caiza, 2013] and [Keuning et al, 2019] for interesting surveys). Despite that, we decided to analyze in detail some alternatives, such as SIETTE (*Intelligent Evaluation System through Tests*) [Conejo et al., 2004] and VPL (*Virtual Programming lab for Moodle*) [Rodríguez-del Pino, 2012]. We finally decided to develop our own framework because existing solutions either required too much effort for creating the type of exercises we were aiming at or they did not fully cover our needs. As a result, we present SERA, a framework for the definition and automatic evaluation of lab exercises focused on the context of computer security.

The rest of the paper is structured as follows. First, we introduce the context that motivated the development of the framework alongside with a description of how labs were carried out before SERA and our objectives when designing it. Next, we provide a detailed description of the framework. In Section IV, we describe the language used for the specification of activities, which is one of the core elements of the framework. Section V presents some of the indicators we establish for assessing the performance of the students. Finally, Section VI concludes the paper and indicate some lines of future work.

II. CONTEXT

The authors of this paper belong to the area of Telematics Engineering at the University of Malaga, where they teach various courses on Computer Security in different degrees, including:

- *Network Security* in the Degree of Telematics Engineering
- *Computer Security and Digital Forensics* in the Degree of Criminology
- *Network Security and Online Transactions* in the Master of Digital Marketing

Although these courses are highly heterogeneous, both in the number of students and their background knowledge in networks and computers, they are all aimed at introducing the student to the basics of information security. Therefore, even though each of these courses has its own twist specific to the degree, they all share some common concepts which are core to all of them.

As a result, there are some recurrent types of lab exercises which appear in all of these courses. Most of these types of exercises are covered by the modules already integrated in SERA. They include:

- Use of cryptographic algorithms
- Management of digital certificates
- Document encryption and signing
- Secure email communications
- Secure web communications

The focus and level of detail of assignments depend on the degree profile but, it is worth noting that the most complex exercises can be divided into simpler activities. These simpler activities are usually common to all courses, which facilitates the reuse of tasks by concatenating them.

Moreover, there is a large number of tools and platforms that need to be covered in the lab, which complicates the task of tracking and evaluating the exercises automatically. This situation is even worse when one of the requirements is that the framework is fully integrated with Moodle.

Finally, designing a framework capable of dealing with different tools and platforms has an important advantage, that is, flexibility. In case suitable modules are created, our framework could be used for teaching different subjects, even though our initial focus was on security courses.

Traditional Approach

We describe here in detail some exercises that are repeated in many security courses and discuss their drawbacks in order to show the reasons for developing the SERA framework.

Digital Certificates life cycle. Digital certificates are the base form of identification secure web server in TLS protocol, but also for e-Government applications. They allow citizens to sign documents and verify them. It is a recurring concept in all security-related courses. Students need to understand how certificates are created, when to trust a certificate, and how to handle revocation. A traditional lab on digital certificates will instruct students on those aspects by asking them to create a Certification Authority, which in essence is a special kind of certificates with some options enabled. The instructor will have to check all certificates submitted and whether the options were enabled or not. He or she will also have to reinforce and explain again in case a student makes a mistake. The instructor will later ask the student to try again using different parameters to create a certificate for a web server or for protecting email communications (S/MIME). Although the process is very similar, students tend to be distracted when setting the right options for the certificate, and the instructor needs to check which options were selected and, once again, explain to the class when someone makes a mistake in order to reinforce. This is clearly a tedious job and an added problem is that while some students need more guidance, other students understand all the concepts and finish the exercise really fast. In the end, the students have to submit their certificates and the instructor needs to check them again to finally decide who passed the exercise.

Transport Layer Security protocol. Online commerce, e-Banking, and many other critical services over the Internet would not be possible if we had no TLS protocol. Understanding the different parameters of the TLS protocol and learning how to configure it, is a recurring exercise in network security courses. The student is presented with some scenario, the domain of a web server that needs to be protected, the certification authority that should be used for the certificates, the version of the protocol that must be enabled and the required parameters. Then, the students have to configure the web server and test it is secure for themselves while trying to understand any mistakes in the configuration or asking the instructor during the class. When a student is stuck, the instructor helps him/her and reinforces the concepts to the rest of the class. Finally, the students submit their configuration and certificates, and the instructor checks everything to decide who passed the exercise.

Drawbacks of the traditional approach:

- All the students receive and solve exactly the same exercise, which allows them to copy the results from other students.
- The instructor needs to waste time repeating the same concepts multiple times, both to individual students and to the whole class in some cases.
- Sometimes, students think they did it right and do not ask in class, resulting in them not getting negative feedback until the instructor is able to review all submissions. By that time, they might not even understand the feedback because they already moved to a different topic.
- It is difficult for the instructor to recall what was the second or third most common mistake, or to recall the name of all students who required feedback in order to follow up.

Objectives

In light of the aforementioned limitations, we established some design goals for SERA. The overall primary goal was to provide a flexible platform for defining customized lab exercises for security courses and automating their evaluation. Moreover, the platform should be easy to use for both students and teachers, and be fully integrated with Moodle.

Regarding the exercises, they should be different for each student. Students should submit a number of files or answers to different questions. After submitting their solutions, they must be automatically evaluated so that they receive immediate feedback on their errors. This is expected to promote self-learning by guiding the students towards the solution.

Students should be able to access all exercises from their Moodle space directly, without any additional registration or login. All exercises should be presented in a homogenous and familiar way. The status of all exercises should be visible to the students, clearly informing which exercises have been passed and which have not.

Teachers should be able to refine or modify exercises easily, with little to no programming skills. Nonetheless, creating new exercises from scratch might require a deeper knowledge of the platform, and in some cases, writing code. Teachers should also be able to track the students' progress accurately, both individually and globally.

III. THE SERA FRAMEWORK

The SERA framework provides a flexible platform for defining lab exercises for security courses and automating their evaluation. As shown in Figure 1, the framework consists of various core components (SERA Engine) that take advantage of the functions defined by a number of modules and utilities for the instantiation of personalized activities and their validation.

The framework defines a language for the specification of activity templates, which are later instantiated into exercises based on some parameters defined in the specification. To prevent cheating, at runtime each student is presented with similar yet different exercises that may require one or more submissions to complete the exercise. The template also defines the checks that should be passed to consider the exercise is correct. In addition, when some of the checks fail the students will receive some feedback that helps them to reach the solution autonomously.

SERA is integrated with Moodle for managing the access of students to the exercises and keeping track of their results. Furthermore, SERA uses an internal database to store relevant information related to the submissions of the students so that the teacher can analyze the performance of the class and react accordingly. Full details about the framework are given below.

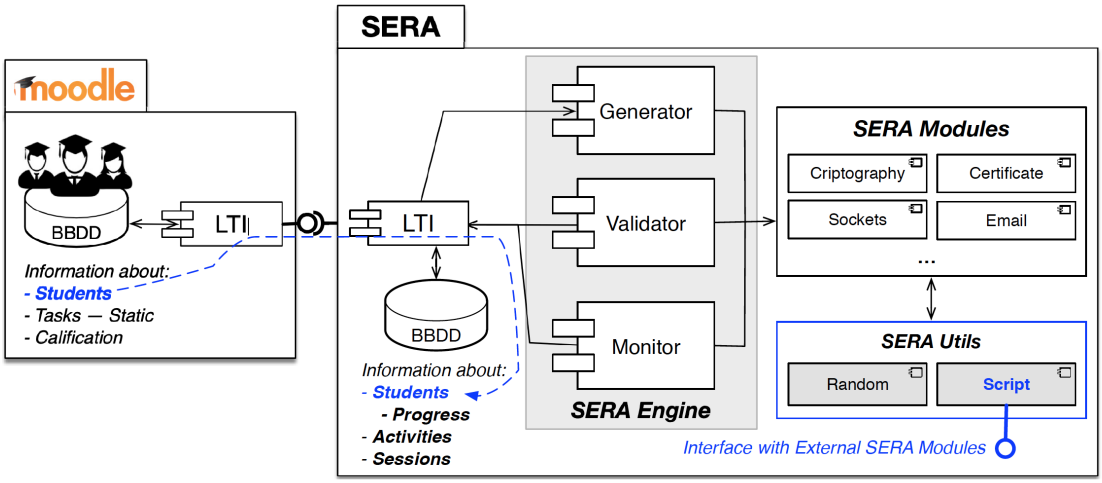

Figure 1: Architecture of the SERA Framework

Engine

The SERA Engine consists of three main components (see Figure 1, SERA Engine):

- **Generator**. It is in charge of the activity generation process by instantiating all the parameters in the activity template.
- **Validator**. It is in charge of checking the conditions specified in the templates according to the student submissions and the activity parameters.
- **Monitor**. It is in charge of checking the conditions of the templates for those activities that require the student to perform some actions in an external system (e.g. sending an e-mail to a given account) rather than submitting a file.

These three components base their operation on a set of specific modules, described in the next subsection, and a set of supporting SERA utilities. We have developed two support libraries for generating random data (*Random util*) and calling external functions (*Script util*) that can be used in the definition of activities as will be shown in section IV.

As shown in Figure 1, SERA can be connected with *Moodle* instances using the LTI (Learning Tool Interoperability[1]) standard, which is widely used by universities around the world. In fact, the LTI standard allows SERA to be connect to most current *Learning Management Systems* (LMS) such as Blackboard, Canvas, Schoology, Brightspace, etc. This standard can help establish a secure connection between the external evaluation tools (i.e., SERA) and the LMS in order to (a) send information about the course and the student to the external tool, and (b) receive feedback and grades in the LMS. We have implemented SERA using *Node.js*, and particularly the *express* framework for web applications, that simplifies session management, and many other tasks.

Modules

As previously mentioned, the actual generation of parameters and validation of submissions is performed using functions provided by the specific modules. Then, all modules need to implement a common interface for parameter generation, submission parsing and evaluation conditions or checks, although not all the interfaces need to be implemented by all modules, e.g. there may be modules that are only used to generate parameters. In other words, every SERA module may include:

[1] Learning Tools Interoperability: http://www.imsglobal.org/activity/learning-tools-interoperability

- **A generation function**. The generator component will call the generation function of the corresponding module for each parameter in the activity template in order to generate the customized activity instance for each student. Note that every module defines its own types.
- **A parsing function**. Model parsing functions allows us to handle file submissions in a more specific way. The parsing function of each module is in charge of extracting relevant information from the file submission and could also perform some basic type checking. This information can be used later in the evaluation of the activity.
- **A set of check functions**. These functions accept as inputs some submissions and parameters and return feedback to the student in case there is some error, in order to help them find the right answer for the activity.

Additionally, we have developed two supporting libraries that are implemented as the modules *Random* and *Script*, in order to generate random data for the activities (numbers, text, cryptographic keys, etc.) and to call external commands, respectively. Apart from that, we have fully implemented a module for the generation and validation of X.509 documents called *Certificate* and partially developed some modules for secure communications (*TLS* module), for checking vulnerabilities in web servers (*Web* module), and for secure email communications with S/MIME (*Email* module).

Activities

The lifecycle of activities consists of three phases (see Figure 2). The instructor starts by defining an activity template, which is a general description of the task to be performed by the students. Before students can work in the resolution of an activity, the template is transformed into a different activity instance for each student thanks to a number of parameters. Lastly, every time a student tries to solve an activity instance, a new activity submission is created.

Figure 2: Activity Lifecycle

Therefore, SERA works with three types of documents related to activities:

- **Activity Templates**. The activity template is the base document for defining activities. All relevant information is included in the template: title, description, parameters, submissions, and checks (see Section IV). Templates can be shared and reused by different courses and remain static in the system, although new templates can be built upon existing ones, for example by composing activities, to adapt activities to each course. The instructor is responsible for defining and maintaining activity templates.
- **Activity Instances**. Once the student starts an activity, all the parameters defined in the template are instantiated, i.e., they are generated using the corresponding generation functions. This information is stored in the activity instance. For every template, there will be many instances, one per each student that has started to work on it. Once an instance has been created, the student will be always presented with the same parameters unless the instance is manually deleted by the instructor and it needs to be recreated by SERA.
- **Activity Submissions**. The student will try to solve the activity instance based on the description provided and the parameters generated. Every time the student tries to solve the activity by submitting a solution, the system will generate an activity submission. The activity submission includes a reference to the activity instance the student is trying to solve along with the solutions submitted, the results of each check and a timestamp. This is stored to keep track of the student's progress (see Section V).

IV. SERA ACTIVITY SPECIFICATION LANGUAGE

The SERA framework defines a flexible language for the specification of activity templates, which are later instantiated into different tasks for each student. For the definition of an activity template, the teacher must include the following elements:

- **Title.** Each activity template must include a short title that will be shown in the activity menu. This way, instructors can easily select activities for their course and students can also see the status of their activities in a meaningful way.
- **Description**. The description is specified using the Embedded JavaScript templating language[2] for enabling the inclusion of customized parameters. All parameters generated for the activity are available as variables and included in the activity instance.
- **Submissions**. Although not mandatory, activities typically require the submission of some text or files by the student using an HTML form. When no submission is specified, a monitoring function should be provided instead to trigger validation. For each requested submission, a parsing function may be specified.
- **Parameters**. In order to generate different activities for each student, the activity template can include a set of parameters that are generated using some the available modules or built-in functions. There are three types of parameters:
  - ***Literal Values***. Strings that might be form based on other parameters using some EJS tags. They can also include all details about the student and course provided by Moodle in the LTI context.
  - ***References***. We can reference parameters or even submission from other activities, so that they became available for the new activity. This type of parameters enables the construction of complex activities by composing simpler activities.
  - ***Generated Values***. Those parameters are generated using specific modules.
- **Checks**. The activity template must also include some conditions to check if the student submissions are satisfactory. Checks are essentially functions that take as arguments submissions and parameters and return some feedback.

In Figure 3, we provide an example of an activity specification where the student is asked to submit a Certificate Signing Request (CSR)[3] with some particular properties. The description of the activity is customized for each student using several parameters. The parameters may include information about the student provided by Moodle through LTI such as the Student's Full Name, Moodle internal User ID or course ID. We also generate a random parameter *r* and activity-specific parameters *n* and *type*. The random parameter is generated using the *Random* module and, in this case, will result in a number between 0 and 99 which will be different for each student.

As for the submissions, a parsing function for the type *csr* will first check if the file submitted is indeed a CSR, e.g. by checking for the PEM header. In case it is not, it will return some feedback to the student, otherwise, it will extract relevant information about the CSR such as the Common Name (CN), the key length, the hash function used, the Subject Alternative Name extension (SAN) if included, etc.

```
{
 title: 'Create a CSR for an end entity',
 desc:`Create a csr in order to obtain a certificate for a web server with the following
CN: <span class= "param"><%= cn %></span>. Remember to set the Subject Alternative Name
(SAN) to the following domain name: <span class= "param"><%= dns %></span>. The key length
should be <span class= "param"><%= n %></span> bits.`,
 submissions: [
   {name:'csr', label:'CSR', type:'csr', module:'certificate'}],
```

[2] Embedded JavaScript (EJS): https://ejs.co/

[3] Certificate Signing Request: https://en.wikipedia.org/wiki/Certificate_signing_request

```
 params: [
    {name: 'n', value: '2048'},
    {name: 'type', value: 'server'},
    {name: 'dns', value: 'www.<%= userId %>.lcc.uma.es'},
    {name: 'r', module: 'random', type:'dec', input:{low:0,high:99}},
    {name: 'cn', value: 'Servidor <%= r %> <%= userId %>'},
 ],
 checks: [
    {module: 'certificate', check:'checkKeyLength', submissions:['csr'],params:['n']},
    {module: 'certificate', check:'checkSan', submissions:['csr'], params:['dns']},
    {module: 'certificate', check:'checkCN', submissions:['csr'], params:['cn']},
    {module: 'certificate', check:'checkType', submissions:['csr'], params:['type']},
    {module: 'certificate', check:'checkSecureHash', submissions:['csr']},
 ]
}
```

Figure 3: Sample Activity Template

The most relevant part to provide feedback to the student and properly assess the submissions is the specification of the checks. The list of checks must be in sync with the description, where the student is told what needs to be done in the activity. Hence, all the parameters included in the description should typically be required in a check in order to be sure that the student followed the instructions correctly.

In Figure 4 we can see how the example activity specified in Figure 3 renders in the browser after the activity instance is created and all parameters are instantiated. All parameters in the description are highlighted in blue to note its presence to the student. In the left-hand side menu, there is a list of activities and whenever activities are solved or failed, status icons will turn from white to either green or red, respectively.

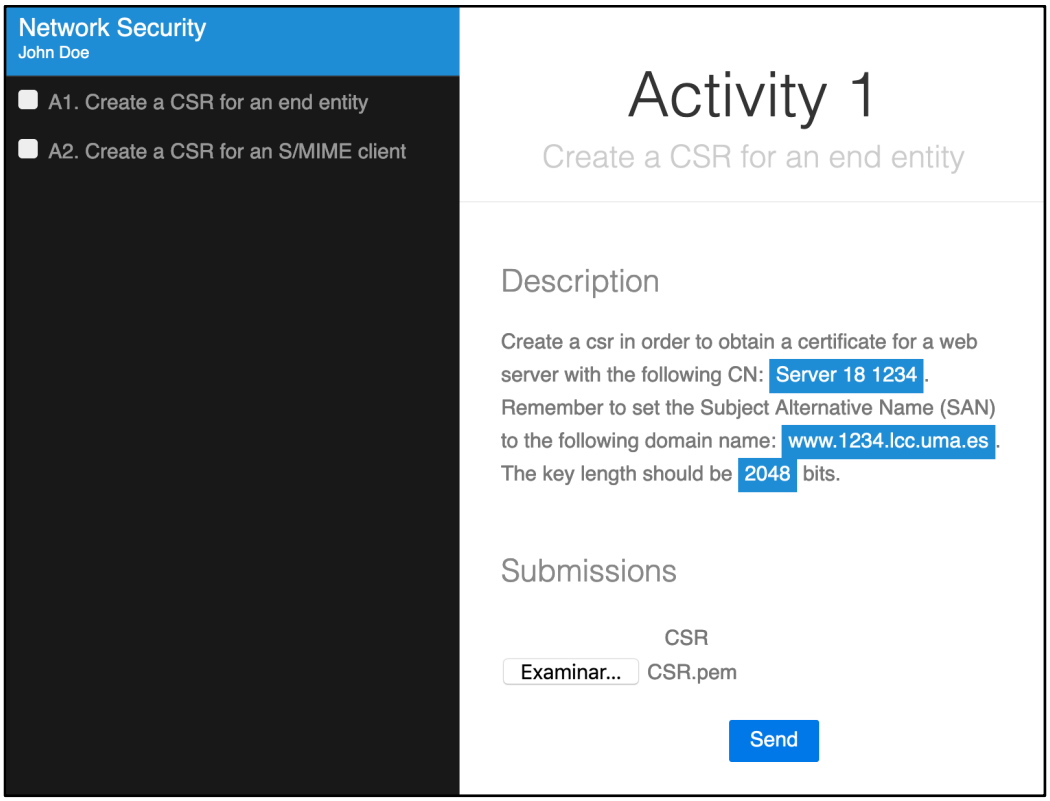

Figure 4: Sample Activity Render

V. SERA INDICATORS

SERA framework has the added advantage of proving the means to analyze both individual student progress as well as group performance. Data gathered from the student submissions allow us to reason about whether the activities where appropriate or if some check condition was too hard to meet. All these analyses are possible because SERA keeps track of all students' submissions.

We have focused on a limited set of indicators that can help to improve the teaching and learning process.

- **Average completion time**. By looking at the first successful submission by the student, we can compute the average time student took to complete this activity. We can even compare average completion time from different courses or different years to identify any deviations. For example, that fact that an activity that last year took 2 minutes on average to be solved is now taking 5 minutes might point out that not enough attention has been paid this year to the concepts involved. Instead of blaming students' performance, we could elaborate more on the concepts and repeat the activity to see if results are better.
- **Most difficult check for students**. By looking at all student submissions we can find what was the last check completed by the student, or the one that took more time. This could help us improve our feedback for this particular student.
- **Average time for each check**. We can go deeper in the analysis and calculate the time between checks, to understand the time devoted to each particular check and provide an average time for the check for all students. This is particularly helpful when defining new activities. If we find that a particular check is getting really hard for most of the students, the cause might be that the description is not clear enough, so we need to review it.
- **Average number of submissions.** We can count how many submissions were needed, on average, to complete each activity. If the activity requires many submissions that might an indicator that the description was not clear or that the student was not ready for it yet. On the other hand, if we find out that some students never require feedback it might be an indicator that they are ahead of the rest of the class.

We expect that these indicators can help teachers refine their lessons and activities and react promptly to undesirable performance of the class.

For our first trial tests with the tool, we allowed the students to work on some basic activities. The activities were not customized and also not included in SERA but they had support from the instructor for 1 hour. After that, the students were presented with a very similar customized activity in SERA. While most of the students were capable of solving the activity in only 5 minutes using SERA, some students were unable to finish the previous activity because of the lack of timely feedback from the instructor.

Although these initial trial tests seem promising, we have yet to confirm the actual benefits of the SERA platform with more tests.

VI. CONCLUSION

This article presents a framework called SERA devised for the creation of personalized lab exercises with automatic evaluation and instantaneous feedback in the context of computer security courses. The framework provides a language for the specification of exercise templates based on a number of existing modules, which are later instantiated into personalized activities for each student. Moreover, it keeps track of their performance.

The framework has been integrated with Moodle and it is currently being tested in several degrees at the University of Malaga. We are also developing new modules for extending the functionality of the framework to other scenarios. So far, the feedback obtained from the students is positive because they can work more autonomously. Even though we do not yet have significant evidence that the framework improves the performance of the students notably, we are certain that it will help

to reduce the teaching load associated with the evaluation and continuous monitoring. Finally, we are in the process of publicly releasing the code as well as installation and user guides.

ACKNOWLEDGEMENTS

This work has been partially funded by the Project on Innovative Education PIE17-137 from the University of Malaga. R. Rios is supported by the 'Captación del Talento Investigador' fellowship from the University of Malaga. A. Nieto is financed by INCIBE through the grant program for excellency in advanced cybersecurity research.

LIST OF REFERENCES

J. Hattie and H. Timperley (2007) *The Power of Feedback*. Review of Educational Research, 77(1), 81–112. DOI: 10.3102/003465430298487

A.W. Bates (2015) *Teaching in a Digital Age*, Tony Bates Associates Ltd, ISBN: 978-0-9952692-1-7

L. Bot, et al. (2007) *'Learning by doing': a teaching method for active learning in scientific graduate education*, European Journal of Engineering Education 30:1, pp. 105-119, Taylor and Francis.

A. Clark, et al (2018) *Evaluating blended and flipped instruction in numerical methods at multiple engineering schools*, International Journal for the Scholarship of Teaching and Learning, vol. 12, no. 1, Article 11.

J.C. Caiza and J.M. Álamo Ramiro (2013). *Programming assignments automatic grading: review of tools and implementations*. 7th International Technology, Education and Development Conference (INTED2013), Valencia, Spain. pp. 5691-5700

Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. 2018. *A Systematic Literature Review of Automated Feedback Generation for Programming Exercises*. ACM Trans. Comput. Educ. 19, 1, Article 3 (September 2018), 43 pages.

R. Conejo, et al (2004). *SIETTE: A web-based tool for adaptive testing*, International Journal of Artificial Intelligence in Education., vol. 14, no. 1, IOS Press, pp. 29–61

J.C. Rodríguez-del Pino, E. Rubio Royo, and Z. Hernández Figueroa, (2012). *A virtual programming lab for moodle with automatic assessment and anti-plagiarism features*, In Proceedings of the 2012 International Conference on e-Learning, e-Business, Enterprise Information Systems, & e-Government, 2012.