

EVERSYS: Efficient exam versioning tool for linear circuits and other problem-based subjects

María Asunción Vicente¹  | César Fernández Peris¹  | Miguel O. Martínez²

¹Health Psychology Department, Miguel Hernandez University, Elche, Spain

²Computer Engineering Department, Miguel Hernandez University, Elche, Spain

Correspondence

César Fernández Peris, Health Psychology Department, Miguel Hernandez University, Av. Universidad s/n, 03202, Elche (Alicante), Spain.
Email: c.fernandez@umh.es

Funding information

Miguel Hernandez University,
Grant/Award Number: PIEU-A/2022/04

Abstract

Exam versioning is a widely used approach to prevent cheating during exams, but it can increase the workload for lecturers in terms of exam preparation and correction. Automated exam versioning can help reduce this workload and minimize errors in correction. This article presents the EVERSYS platform, which is focused on linear circuits but can be applied to other engineering fields. An exam problem is described as a circuit schema, a set of component values, and a set of questions to be answered. The EVERSYS platform allows lecturers to easily create exam versions with high variability, including different schemas, component values, and question sets. The platform is based on the MATLAB/Octave programming language and HTML document descriptions. Creating a multiversion exam is straightforward, involving the adaptation of a set of predefined MATLAB scripts and modification of the HTML format if necessary. A set of example exams is provided as well as the complete platform source code. The platform was tested during a 2021–2022 compulsory course on linear circuits at a Spanish University. The results indicated that the platform significantly reduced the time required for exam preparation and correction, and there was no statistically significant difference in student performance across different exam versions. Students reported high levels of satisfaction with the platform, and almost all agreed that it effectively prevents cheating during exams.

KEYWORDS

automation, cheating, exam versioning, linear circuits, MATLAB

1 | INTRODUCTION

The online teaching and evaluation methodologies imposed by coronavirus disease 2019 (COVID-19) have emphasized the problem of cheating during exams. However, cheating is not exclusively used in online learning. In large groups, it is difficult to prevent students from passing information to each other, directly copying from their classmates, or using hidden electronic devices (mobile phones, smartwatches, mini-earphones) to copy remotely.

Numerous studies have been conducted to address this issue. For example, Romaniuk and Łukasiewicz-Wieleba [20] found that up to 20% of students admitted to using unauthorized assistance during exams. Other studies indicated that cheating is one of the main challenges in online learning [25], and researchers have proposed changes in the way students are evaluated, both to avoid cheating and to improve the learning process. For example, Schultz and Callahan [23] focused on novel evaluation methodologies for chemistry

subjects; and Photopoulos et al. [17] conducted surveys to analyze the students' opinions on different evaluation methodologies.

Across all academic disciplines and study levels, the most widely adopted solution to prevent cheating is exam versioning, but this increases the lecturer's workload, for both preparing and correcting the tests, and it presents a high risk of correction errors due to the differences between versions. To solve these problems, many attempts have been made to automate exam versioning. The most common approach is to generate a large database of exam questions or problems and to randomly assign questions to students. Multiple platforms support such functionality, like Moodle [31]; or the platform presented in Rjoub et al. [19]. A similar option is the reordering of questions and answers in multiple-choice exams, as proposed in Fernández et al. [8] and available in commercial products like Tomamix [32] or open-sourced platforms like TestMaker [33]. All these options are valid for cheating avoidance, but their efficiency is limited, particularly during exam preparation, as each version should be prepared independently and, thus, the time required to prepare the exam increases linearly with the number of versions. A different approach for exam versioning is to use the same exam skeleton for all versions but randomly vary the numeric data between versions. Such an approach can be found in the proposals of Rusak and Yan [22], or Fernández and Vicente [7].

All the previous approaches to exam versioning are compared in Table 1. Ideally, an effective versioning tool should maximize variability between versions by incorporating different problem statements, value sets, and question types while also allowing for loop automation to enhance efficiency. However, from the table, it is evident that currently there are no tools that fulfill all these requirements. Tools that rely on randomly selecting problems from a database can offer complete variability between versions. However, they lack the capability for automated problem generation using loops. On the other hand, tools based on question or answer reordering in

multiple-choice exams do not provide substantial variability between versions, like tools based on a single exam skeleton where only values change between versions. Therefore, there remains a need for a versioning tool that combines the advantages of maximum variability and loop automation to optimize the exam versioning process.

Compared with the approaches presented in Rusak & Yan [22] and Fernández & Vicente [7], where numeric data vary among versions, but the exam skeleton is the same in all cases, the present study proposes a more flexible system in which the exam skeleton, numeric data, and even question sets can vary among versions. Although it was developed for the subject of linear circuits, its applicability to other subjects was considered. Instead of using specific software, the new tool is based on common scientific software such as MATLAB [28] or its open-source alternative Octave [6], which are familiar to lecturers across multiple engineering fields.

The remainder of this article is structured as follows. Section 2 discusses the objective of the study, that is, the requirements for the exam-versioning tool. Section 3 discusses the methodology used in this study. Section 4 discusses the structure of the proposed platform, EVERSYS (Exam VERSIONING SYStem), and presents a usage example. The results obtained using this platform are presented in Section 5. Finally, Section 6 discusses the results, and Section 7 presents the conclusions.

2 | OBJECTIVE

The objective of the EVERSYS platform is to semi-automate exam generation for the subject of linear circuits, while satisfying three key requirements:

- The platform must generate multiple exam versions to prevent cheating.
- The process of exam generation should not present an excessive workload.

TABLE 1 Comparison of previous approaches to exam versioning.

Approach	References	Variability between versions			Allows loop automation
		Problem data	Problem statement	Questions asked	
Random selection from database	(19, 31)	Yes	Yes	Yes	No
Reordering of Q&A in multiple-choice tests	(8, 32, 33)	No	No	Yes	Yes
Data changing approaches	(7, 22)	Yes	No	No	Yes
Proposed tool (EVERSYS)		Yes	Yes	Yes	Yes

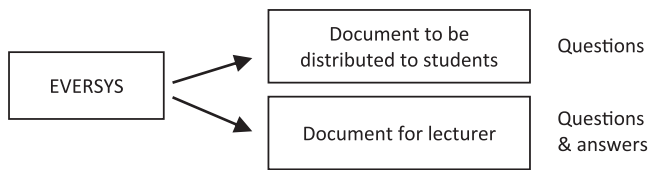


FIGURE 1 Structure of the Exam Versioning System (EVERSYS) platform.

- The platform must minimize errors during exam correction by facilitating the correction task.

To meet these requirements, the platform must produce two distinct documents: one for students (comprising all exam versions without answers) and another for the lecturer (comprising all exam versions with answers). Figure 1 illustrates this approach.

As mentioned, one of the objectives is to minimize the amount of repetitive and tedious work for the lecturer such that they can focus on the most important tasks: (1) designing an exam that properly evaluates the knowledge acquired by the students and (2) correcting the results.

We considered three hypotheses in our work: first, an exam versioning tool could be more efficient than manual versioning, both for exam preparation and for exam correction. Second, that exam versioning, when carried out properly, does not introduce bias in the student marks. Third, exam versioning, and its associated reduction in cheating, can be well accepted by the students.

To check the validity of the hypotheses, exam preparation and correction times were registered; student marks were analyzed; and student satisfaction was compared with that of previous courses. Different threats to validity were identified: first, the comparison of preparation and correction times against a manual versioning approach required the assumption of certain times, which were estimated prudently and based on previous experience. However, these time estimations may not be valid for different lecturers. A second threat to validity involves the comparison of student satisfaction with previous courses, since changes in student satisfaction may depend on other factors apart from the introduction of exam versioning. In Section 5, the incidence of these threats in the validation of the hypothesis is analyzed.

The main assumption was that the subjects could be evaluated through problem-solving exams, where changing the problem values is one of the ways to create multiple exam versions. A secondary assumption is that the process of obtaining correct results for all versions can be automated with Matlab/Octave or similar tools. These assumptions hold true not only for Linear Circuit

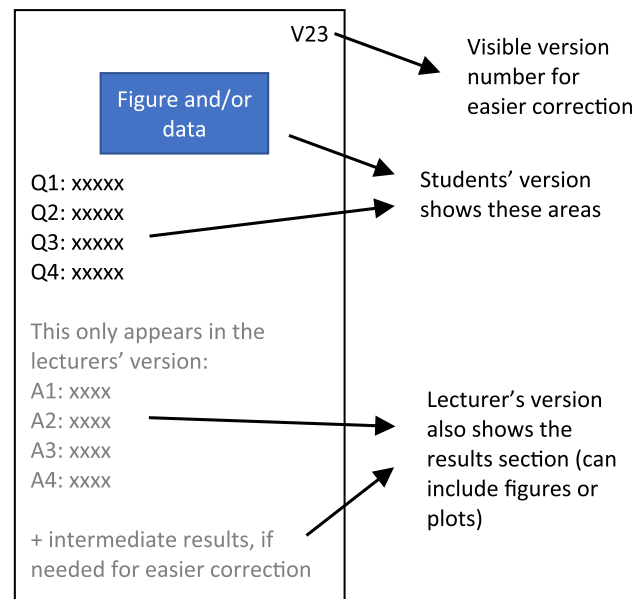


FIGURE 2 Example exam.

subjects but also for numerous other subjects within engineering degrees.

3 | METHODOLOGY

The structure of the exam documents for students and lecturers is shown in Figure 2, where Q1–Q4 represent the questions asked to the students, and A1–A4 represent the correct answers, which are only visible in the lecturer's version.

To develop the EVERSYS platform, two subtasks were identified and handled independently: automatic PDF generation and automatic linear circuit solving.

3.1 | Automatic PDF generation

As the exam is to be printed, PDF was selected as the standard printer-ready format. The goal was to automatically generate a PDF document containing all exam versions from a set of question descriptions, data, and answers.

Different methods can be used to generate a custom PDF document programmatically.

- Creating PDF content directly using a PHP code [18] or other programming languages such as Java [27], or even MATLAB [29]. Advanced programming skills are required in all cases.
- Creating an intermediate Microsoft Word document in XML docx format. There are also libraries for

generating docx documents from Java [26] and PHP [14]. However, the level of complexity is comparable to that of the previous option, and advanced programming skills are required.

- Creating an intermediate HTML document. This is easier than the previous options with regard to programming skills, but specific templates are required to achieve correct pagination when converting to PDF.

The last option (creating an intermediate HTML document) was chosen to develop a user-friendly system with broad applicability. MATLAB/Octave was selected as the platform for automating the generation of HTML documents. The HTML format allows lecturers to easily customize the exams.

3.2 | Automatic circuit problem solving

The exam generator must create both exam questions and their corresponding answers. The process of obtaining answers for each version of the exam must be automated.

In the case of linear circuit theory, the problems to be solved can be expressed as a system of linear equations with real or complex numbers (node or mesh analysis for direct current [DC] or alternating current [AC]) or simple quadratic equations (e.g., finding the resistor for a certain power consumption).

To facilitate exam versioning, no symbolic answers should be required from students. An example of a question to avoid is as follows:

“using this circuit, derive a formula that shows the power consumption in the resistor as a function of the amplitude of the voltage source and the inductance value.”

Instead, these questions must be reformulated as follows:

“in this circuit, with a voltage source that provides 240 Vef at 50 Hz and an inductance of 10 mH, calculate the power consumption of the resistor in watts.”

Each exam version consists of a circuit, a set of values for the circuit components, and a set of questions to be answered with numeric results.

As no symbolic results are asked, it is unnecessary to use symbolic math to generate answers, which simplifies the automation of problem solving. There are two main options for this:

- Use SPICE tools [1, 4] or similar circuit-solving tools to obtain the answers; and
- Use MATLAB/Octave or other numeric computing tools to obtain the answers.

The first option (SPICE) is theoretically easier, as circuit equations are not required, but it has an important drawback: the final answer is obtained, but certain intermediate results are inaccessible. These intermediate results can be useful for accelerating exam correction, for example, detecting the source of a certain error in some of the students' answers. The second option (MATLAB) has the advantage of flexibility even though it requires programming skills. Attempts have been made to create SPICE-like tools with the capability to obtain intermediate results [16], but their flexibility and applicability are limited in comparison with a generic programmable tool for numeric computing such as MATLAB or Octave.

4 | DESCRIPTION OF EVERSIS PLATFORM

4.1 | Proposed structure for exam problems

The structure of a linear-circuit problem generated by EVERSIS consists of three elements:

1. A circuit schema;
2. A set of values for the circuit components; and
3. A set of questions to be answered by the students.

Figure 3 shows a simple example of such a linear-circuit problem, identifying its three components (schema, values, and questions). Clearly, a problem in a real exam should be more complex, but the structure would be the same.

Ideally, each exam version should be as different as possible from the other versions: different schema, different sets of values, and different questions. Our proposal allows the lecturer to create different schemas, sets of values for each schema, and sets of questions for each value set. Thus, the number of different exam versions can be large, as shown in Figure 4. The exact number of versions can be calculated as $\sum_{\text{schemas}} \sum_{\text{valuesets}} \text{Questionsets}$.

Although the EVERSIS platform offers three levels of variability, only the first level (schema loop) requires additional effort from the lecturer, as each schema needs a distinct problem-solving approach. The remaining variability is attained through loops that perform the same computations with different value sets (value-set loop) and only include a specific subset of questions in the exam (question-set loop).

In practice, it is unnecessary to create a unique exam version for each student. A total of 12 different versions

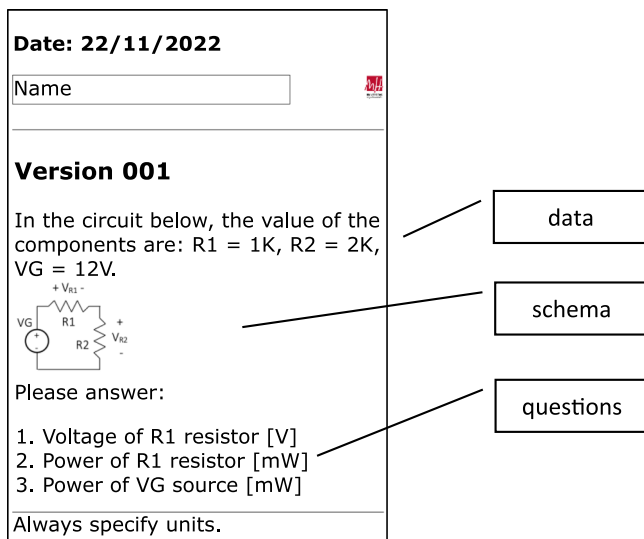


FIGURE 3 Example of a simple linear-circuit problem consisting of three main elements: schema, values, and questions.

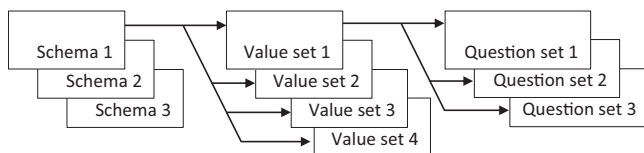


FIGURE 4 Possible sources of variability in a linear-circuit problem.

make it difficult for students to cheat (even if the total number of students exceeds 200), provided that the students do not know who has the same exam version. Our strategy to reach this goal is simple: once the total number of versions is reached (e.g., 12 different exams), the loop is repeated, and version 1 is generated again. However, version 1 is labeled as version 13 to prevent students from knowing who has the same exam version.

4.2 | Software architecture of EVERSYS

The EVERSYS platform comprises a collection of MATLAB/Octave scripts and a series of HTML templates that are associated with a CSS style file. The primary objective was to develop a user-friendly platform that only requires lecturers to adapt the predefined MATLAB scripts to their exams and customize the appearance (if necessary) by modifying the HTML templates or CSS style files. As most lecturers in the field of linear circuit theory are familiar with the MATLAB/Octave environment, the platform is built on these common scripting languages. Additionally, HTML and CSS are widely

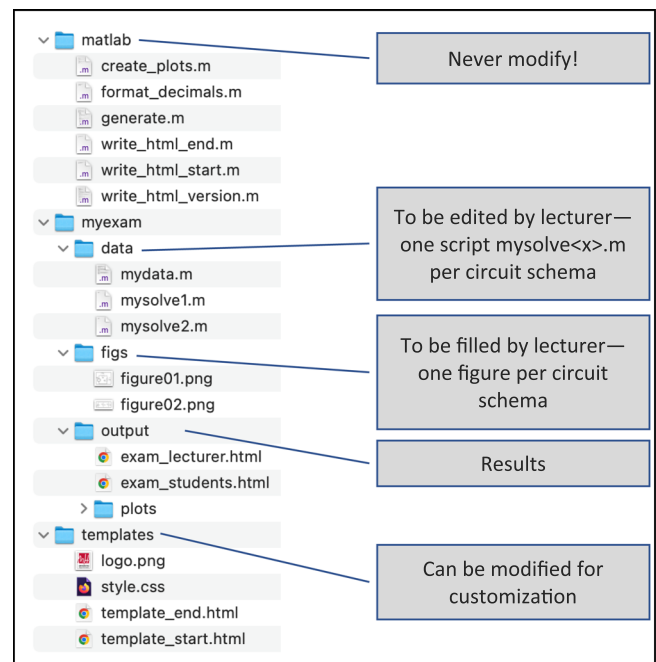


FIGURE 5 Folder structure of the Exam VERSIONING SYSTEM (EVERSYS) platform.

known, at least at a fundamental level, by most lecturers. Hence, most lecturers can make simple customizations, such as text formatting, with ease.

The complete structure of the system is illustrated in Figure 5, with three primary folders, each containing specific code and materials:

- **matlab folder:** contains noneditable MATLAB code, particularly the `generate.m` script, which should be executed to generate the exams.
- **myexam folder:** provides a detailed description of the exam and is further divided into three subcategories:
 - **data:** contains a MATLAB-style description of exam versions, component data, solving code, and question sets. The `mydata.m` file specifies the number of versions to print, datasets for each circuit schema, and question sets for each schema. In addition, the lecturer must provide a `mysolve<x>.m` file for each schema. All files contain initial example code that can be adapted to a specific exam problem.
 - **figs:** includes figures that depict the circuit schemas, in png format. They can be created with any editing tool if they are converted to png format in the last stage. There should be one figure for each circuit schema.
 - **output:** contains the generated documents: student and lecturer versions of the exam as printer-ready HTML files. These files can be printed on paper or saved as PDF documents.

- templates folder: contains files responsible for the exam appearance. They can be left unmodified or modified to customize the exam style and look.

The next section outlines the steps required to generate a basic exam using the EVERSIS platform. In addition, the complete source code has been made available as Supporting Information Material, along with other exam examples and detailed usage instructions.

4.3 | Application example

As an application example, we present a simple exam, where versioning is based on the following structure:

- Two different schemas: voltage and current dividers.
- Three data options for each schema.
- Two question sets for each schema.

Consequently, there are $2 \text{ schemas} \times 3 \text{ data options} \times 2 \text{ question sets} = 12$ different versions.

Specifically, some students must solve a voltage divider problem (Schema 1 in Figure 6), and others must solve a current divider problem (Schema 2 in Figure 6).

For the group of students working on the voltage divider problem, the question sets differ depending on the specific exam version:

- Some students are asked to provide the voltage and power of resistor R1, as well as the power of the voltage source.
- Other students are asked to provide the voltage and power of resistor R2, as well as the power of the voltage source.

Similarly, for the group of students working on the current divider problem, the question sets differ depending on the specific exam version:

- Some students are asked to provide the current and power of resistor R1, as well as the power of the current source.
- Other students are asked to provide the current and power of resistor R2, as well as the power of the current source.

Even in instances where the schema and question sets are identical across two exam versions, there are three unique value sets (three different sets of values for the resistors and voltage or current sources). As a result, this yields a total of 12 distinct exam versions.

Consequently, the proposed EVERSIS platform offers a simple solution to create an exam with considerable variability, by simply adapting three predefined MATLAB/Octave scripts provided in the “myexam” folder. A detailed description of these scripts is available as Supporting Information Material (document usage_instructions.docx).

Moreover, version ordering allows the efficient distribution of exams to students while maximizing the difficulty of cheating. Figure 7 shows the output of the tool, with increased font sizes and reduced margins for better visualization. Only the lecturer version (with added results) and the first eight exam versions are shown. The full documents (student versions 1–100 and lecturer versions 1–100), with standard font sizes and margins, can be downloaded as Supporting Information Material.

As shown in Figure 7, contiguous students (those with consecutive exam versions) have different schemas (i.e., the most different versions, nothing in common), so copying from the neighboring student is useless. Non-contiguous students share schemas but have different question sets, which also makes copying useless. Finally, students who are distant from each other (e.g., versions 1 and 5) may share schemas and question sets, but they have different datasets, meaning that their results will not match. By chance, two students may have identical exams but different version numbers, so they are unlikely

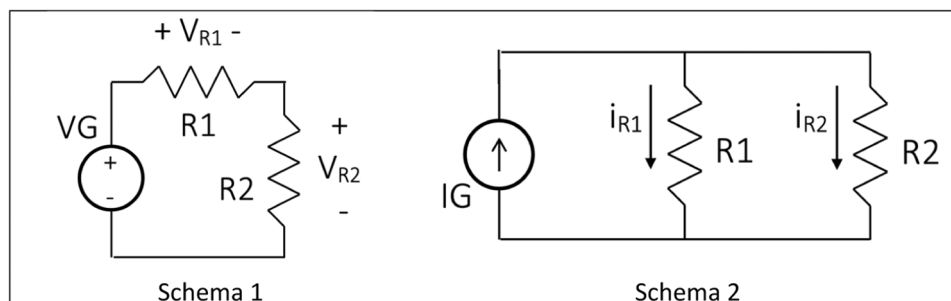


FIGURE 6 Schemas for the example exam (Schema 1: voltage divider; Schema 2: current divider).

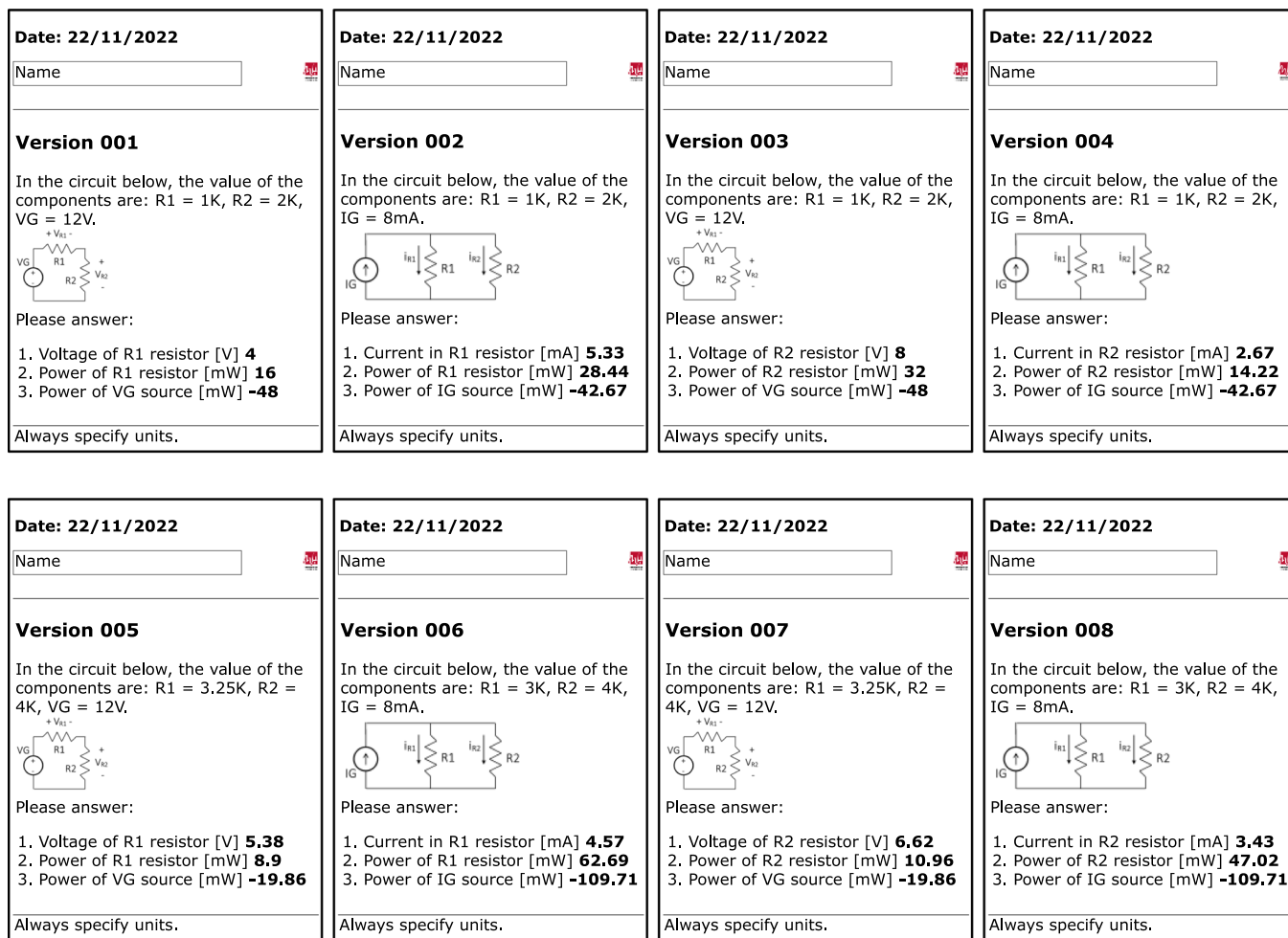


FIGURE 7 First eight versions of the generated example exam (lecturer version, with added results).

to notice that they have the same exam (e.g., in our example, versions 1 and 13 share schemas, question sets, and datasets).

A schematic view of the distribution of the first 21 exam versions in a standard classroom is presented in Figure 8. As previously mentioned, the exam versions start repeating from the 13th version, which is identical to the 1st version. Therefore, students with identical exams sit far away from each other, and they do not know that they have the same exam because the version numbers are different (e.g., version #1 vs version #13). Consequently, the 12 versions used in this example are sufficient for a large classroom, and even fewer versions would make cheating difficult.

4.4 | Additional exam features

The EVERSIS platform offers lecturers the flexibility to modify the standard structure of an exam, which consists of data, schema, and questions. As a first

example, Figure 9 shows how lecturers can add intermediate results in their exam versions, for easier correction. In this example, the exam question requires students to compute previously the Thevenin equivalent of the circuit. The lecturer can check whether the student obtained this intermediate result correctly, even though it is not a specific exam question. Adding intermediate results can be easily accomplished with the EVERSIS platform. Further details on the required MATLAB/Octave code are available as Supporting Information Material (document usage_instructions.docx). Besides, some example exams using this feature can also be downloaded (e.g., Course 2021-2022 Example Exams, Exam P1).

As a second example, Figure 10 shows how to add plots as question results. In this example, students are requested to present their results graphically as a function of time, and the lecturer is provided with the plot to aid with correction. Adding plots is also a straightforward process with the EVERSIS platform. Further details on the required MATLAB/Octave code are available as supplementary

... sequence continues up to the number of exam attendees ...

VER: 15 SCH: 1 QSET: 2 DSET: 1	VER: 16 SCH: 2 QSET: 2 DSET: 1	VER: 17 SCH: 1 QSET: 1 DSET: 2	VER: 18 SCH: 2 QSET: 1 DSET: 2	VER: 19 SCH: 1 QSET: 2 DSET: 2	VER: 20 SCH: 2 QSET: 2 DSET: 2	VER: 21 SCH: 1 QSET: 1 DSET: 3
VER: 08 SCH: 2 QSET: 2 DSET: 2	VER: 09 SCH: 1 QSET: 1 DSET: 3	VER: 10 SCH: 2 QSET: 1 DSET: 3	VER: 11 SCH: 1 QSET: 2 DSET: 3	VER: 12 SCH: 2 QSET: 2 DSET: 3	VER: 13 SCH: 1 QSET: 1 DSET: 1	VER: 14 SCH: 2 QSET: 1 DSET: 1
VER: 01 SCH: 1 QSET: 1 DSET: 1	VER: 02 SCH: 2 QSET: 1 DSET: 1	VER: 03 SCH: 1 QSET: 2 DSET: 1	VER: 04 SCH: 2 QSET: 2 DSET: 1	VER: 05 SCH: 1 QSET: 1 DSET: 2	VER: 06 SCH: 2 QSET: 1 DSET: 2	VER: 07 SCH: 1 QSET: 2 DSET: 2

FIGURE 8 Distribution of the example exam in a standard classroom; closer students have the most different exam versions, with different schemas or different question sets (VER = version, SCH = schema, QSET = question set, DSET = data set).

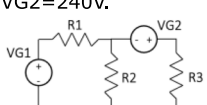
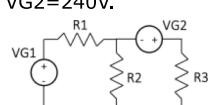
Name <input type="text"/>	Name <input type="text"/>
<p>Version 001</p> <p>Data: $R_1=4K$, $R_2=2K$, $V_{G1}=120V$, $V_{G2}=240V$.</p>  <p>Please answer:</p> <p>1. Resistor R_3 that dissipates 5W (max. of 2 possible values) [K]</p> <p>Always specify units.</p>	<p>Version 001</p> <p>Data: $R_1=4K$, $R_2=2K$, $V_{G1}=120V$, $V_{G2}=240V$.</p>  <p>Please answer:</p> <p>1. Resistor R_3 that dissipates 5W (max. of 2 possible values) [K] 12.88</p> <p>Intermediate results:</p> <ul style="list-style-type: none"> • V_{TH} [V] 280 • R_{TH} [K] 1.33 <p>Always specify units.</p>
<i>Version for students</i>	<i>Version for lecturer</i>

FIGURE 9 Example of exam with intermediate results in the lecturer version.

material (document usage_instructions.docx). Besides, some example exams using this feature can also be downloaded (e.g. Course 2021–2022 Example Exams, Exam P6).

Figure 11 shows an additional example where the goal is to exemplify how to handle exam questions requiring more than one figure. To keep the platform as

easy to use as possible, the structure for all exam questions remains fixed: main text, one figure, and questions. When multiple figures are necessary, they must be combined and referenced within the exam text. This specific example is also provided as Supporting Information Material for further reference.


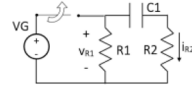
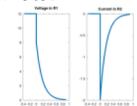
<p>Name <input type="text"/></p> <p>Version 001</p> <p>Switch initially closed, opens at $t=0$. $R1=4K$, $R2=2K$, $C1=30\mu F$, $V_G=12V$.</p>  <p>Please answer:</p> <ol style="list-style-type: none"> Time constant [s] Draw voltage in R1 and current in R2: $v_{R1}[V]$, $i_{R2}[mA]$ <p>Always specify units.</p> <p style="text-align: center;"><i>Version for students</i></p>	<p>Name <input type="text"/></p> <p>Version 001</p> <p>Switch initially closed, opens at $t=0$. $R1=4K$, $R2=2K$, $C1=30\mu F$, $V_G=12V$.</p>  <p>Please answer:</p> <ol style="list-style-type: none"> Time constant [s] 0.18 Draw voltage in R1 and current in R2: $v_{R1}[V]$, $i_{R2}[mA]$ <p>Plots:</p>  <p>Always specify units.</p> <p style="text-align: center;"><i>Version for lecturer</i></p>
---	--

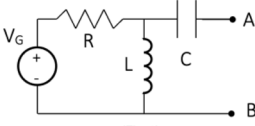
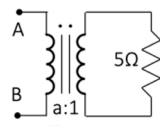
FIGURE 10 Example of exam with plots in the lecturer version.

Date: 22/11/2022

Name

Version 001

In the circuit below (figure 1), the value of the components are: $R = 12\Omega$, $L = 40mH$, $C = 100\mu F$, $V_G = 120\cos(100t)$ V

Please answer:

- Thevenin voltage V_{TH} between A and B. [V] **12+36j = 37.95/71.57°**
- Thevenin impedance Z_{TH} between A and B [ohm] **1.2-96.4j = 96.41/-89.29°**
- Maximum power of an impedance connected to A and B [W] **150**
- Maximum power of a resistor connected to A and B [W] **3.69**
- If the ideal transformer of figure 2 is connected to the terminals AB of the circuit of figure 1, which transformation ratio a maximizes the 5Ω resistor power consumption? **4.39**

Always specify units.

FIGURE 11 Example of exam with multiple figures in the lecturer version.

4.5 | Other customization options

The EVERSIS platform is designed to make exam versioning easy for most lecturers by providing predefined MATLAB scripts. However, the platform also offers additional customization options when required. This is one of the advantages of using HTML as an intermediate format before generating the final PDF printable document. There are two main options for additional customization.

- (1) The textual data describing the exam is always provided by the lecturer as MATLAB strings, which can have HTML formatting if required.
- (2) The system includes various HTML documents acting as templates, and a CSS style file, as described in Section 4. Lecturers can modify these templates and style files to meet their formatting needs, provided that the main elements are preserved.

Basic examples of these customization options are available as Supporting Information Material (document usage_example.docx).

5 | RESULTS

The EVERSIS platform was tested during a 2021–2022 compulsory course on linear circuit theory for the Mechanical Engineering degree at a Spanish University.

TABLE 2 Created exams.

Exam	Schemas	Question sets	Datasets	Total versions	Total students
1. DC analysis	2	1	10	20	82
2. AC power factor	2	1	10	20	74
3. AC analysis	2	1	10	20	74
4. AC magnetic coupling	2	1	11	22	67
5. Three-phase circuits	2	1	11	22	67
6. DC transient behavior	2	1	11	22	18

Abbreviations: AC, alternating current; DC, direct current.

TABLE 3 Exam preparation and correction times.

Exam	Preparation time (h)	Correction time (h)	Total students	Correction time per student (min)
1	7.5	8.5	80	6.37
2	5.0	6.5	74	5.27
3	6.0	8.0	73	6.57
4	5.5	5.0	67	4.48
5	4.5	8.5	63	8.09
6	6.0	3.5	16	13.12
Average	5.75	6.67	62.17	7.32

Six exams were conducted during the course. The main data for these exams are presented in Table 2. In all cases, there were two different schemas and only one question set, with the number of datasets ranging from 10 to 11. All the exams can be downloaded as Supporting Information Material.

As shown in the table, the number of different exam versions was considerably smaller than the number of students (except for exam 6, which was not mandatory). Furthermore, not all the variability allowed by the platform was used, as all the exams had only one question set. However, no instances of cheating were observed during the correction of the exams. Specifically, no similarities were found among students sharing the same exam versions. This is consistent with the expected behavior of the platform.

To assess the efficiency of the EVERSYS platform, the preparation and correction times for all six exams were measured and are presented in Table 3. The average preparation time was 5.75 h. The correction times were also reasonable, with an average of 6.67 h for correcting an entire exam (i.e., answers from all students) and an average of 7.32 min for correcting a single exam (i.e., answers from one student). The time allocated to exam

sorting and spreadsheet work was included in all the correction times.

To compare these preparation and correction times with other options (i.e., EVERSYS vs. manual versioning vs. no versioning at all), the previous times have been subdivided in their main components.

Table 4 is focused on exam *preparation* times. Regardless of the methodology used, certain common aspects always demand a similar time (assuming the exam is created from scratch and not copied from a repository). First, a draft must be created in paper: that is the “manual problem design” time specified in the table. Second, the problem must be solved with values for all components (“problem solving with initial values” in the table). Third, the circuit schema must be drawn, usually with digital tools. These three tasks must be repeated per each schema, or individual problem.

The remaining tasks depend on the methodology used. Whether using EVERSYS or manual versioning, several lists of values for each circuit component must be created (“value set creation” in the table). If EVERSYS is used, there must be a time allocated to prepare the Matlab + HTML code; and a final coherence check, where the generated PDF is revised to detect possible

TABLE 4 Exam preparation time: detail and comparison with other options

Exam	1	2	3	4	5	6	AVG
<i>Common to all approaches</i>							
Manual problem design (per schema)	90	50	75	45	35	70	60.8
Problem-solving with initial values (per schema)	30	30	30	30	30	30	30
Drawing of each schema	20	10	15	15	10	15	14.2
<i>A. EVERSYS approach</i>							
Value set creation (per schema)	15	15	15	15	15	15	15
Matlab+html preparation (per schema)	45	30	30	45	30	30	35
Coherence check (per schema)	25	15	15	15	15	20	17.5
<i>B: Manual versioning approach</i>							
Value set creation (per schema)	15	15	15	15	15	15	15
Word processing and versioning, 10/11 versions (per schema)	120	120	120	120	120	120	120
<i>C: No versions approach</i>							
Word processing, one version	15	15	15	15	15	15	15
TOTAL A (EVERSYS, two schemas)	450	300	360	330	270	360	345
TOTAL B (manual versioning, two schemas)	550	450	510	450	420	500	480
TOTAL C (no versions, one schema)	155	105	135	105	90	130	120

Note: All times are given in minutes.

inconsistencies, such as problems without feasible solutions due to certain component values. On the other hand, with manual versioning, considerable time is invested in manually creating the final exams using word processing software, involving copying, pasting, and generating similar exam versions with varying component values and questions for students. Finally, if no versioning is used, only a single word-processing step is required.

The values presented in the table are based on the exams conducted during the 2021–2022 academic year, comprising six exams, each featuring two schemas, with the number of value sets per schema ranging from 10 to 11. Under these circumstances, the average preparation time using the EVERSYS platform is 345 min per exam, while the estimated time using manual versioning would have been 480 min per exam, and creating an exam with no versions would have taken just 120 min. Although versioning requires more preparation time, based on our estimations, the EVERSYS tool helps reduce this additional time from 360 min (480 minus 120) to 225 min (345 minus 120).

Regarding exam correction time, the steps involved in correcting the exams, regardless of the option chosen, mainly include: (1) an initial step of exam

ordering and a final step of spreadsheet work to publish the results; (2) a quick check to compare each student's results to the correct results; (3) a meticulous review of each student's explanations and processes to ensure their coherence with the results and to identify the source of mistakes when the results are incorrect (depending on the source of a mistake, there may be different gradings).

These are the only steps required for both the EVERSYS platform and the no-versioning approach, as the correct results are known from the exam preparation stage in both cases. However, in the manual versioning approach, an additional step is required, which involves solving all exam versions to obtain the correct results.

Table 5 presents the results, considering an estimated time of 0.5 min for checking the correctness of the final results, 120 min for exam sorting and spreadsheet work, and 20 min for solving each exam version (whether manually or using SPICE). The average values show that the EVERSYS platform does not require extra time for exam correction compared with the nonversioning approach (since the correct results for all versions are known in advance). On the other hand, the manual versioning approach requires twice the amount of time for correction.

TABLE 5 Exam correction time: Detail and comparison with other options.

Exam	1	2	3	4	5	6	AVG
Students	80	74	73	67	63	16	62.17
Schemas	2	2	2	2	2	2	2
Data sets	10	10	10	11	11	11	10.5
<i>Common to all approaches</i>							
Exam sorting and final spreadsheet work	120	120	120	120	120	120	120
Check for correct results (per student)	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Revision of process followed and explanations (per student)	4.4	3.1	4.4	2.2	5.7	5.1	4.15
<i>B. Manual versioning approach</i>							
Problem solving (per schema and value set)	20	20	20	20	20	20	20
TOTAL A (EVERSYS)	512	386	478	301	511	210	399.7
TOTAL B (manual versioning, two schemas, 10/11 data sets)	912	786	878	741	951	650	819.7
TOTAL C (no versions)	512	386	478	301	511	210	399.7

Note: All times are given in minutes.

TABLE 6 Comparison of preparation and correction times versus a manual versioning approach (all times are given in minutes).

	Manual versioning mean (std)	EVERSYS mean (std)	<i>t</i> test <i>p</i> Value
Preparation time	480 (48.2)	345 (62.2)	.0018
Correction time	819.7 (114.3)	399.7 (124.1)	.0001
Total	1299.7 (124.5)	744.7 (144.3)	<.0001

Note: Bold values show statistically significant differences.

Abbreviation: std, standard deviation.

To check the validity of the first hypothesis of the work, the increase in efficiency versus a manual versioning approach, both preparation and correction times are compared in Table 6. The reduction in time is clear, particularly for correction times, which, on average are less than half of those of the manual versioning approach. Besides, a Student's *t* test shows that the differences are statistically significant in all cases.

According to the results, the first hypothesis should be accepted. However, as mentioned previously, there are threats to validity. In particular, some of the times used in the comparison have been estimated (e.g., exam sorting and spreadsheet work, circuit drawing, etc.). We have tried to reduce the effect of these estimations by using the same values in both options (manual versioning and automated versioning); and by using reasonable estimations according to our experience. Nevertheless, the results may not be extended to different subjects than Linear Circuit Analysis.

The marks obtained by the students were also analyzed. Particularly, as the exams had more than one schema, there may have been differences in difficulty between the versions. Table 7 presents the average marks obtained by the students in all exams, classified by the schemas of their versions. A Student's *t* test was performed, and the results showed that there were no statistically significant differences in the marks obtained for the different versions.

According to these results, the second hypothesis of the work (exam versioning, when carried out properly, does not introduce bias the student marks) should be accepted. Nonetheless, the key for avoiding biased marks is a careful selection of exam questions of similar difficulty, so EVERSYS, like other versioning tools, should be properly used.

The satisfaction of the students was evaluated in two ways. First, a standard quality survey was conducted, whose results are presented in Table 8. To check the effect of the introduction of the exam versioning

TABLE 7 Average marks.

Exam	Marks for Schema 1: average (std), N	Marks for Schema 2: average (std), N	<i>t</i> test <i>p</i> Value
1	4.76 (3.12), 42	4.42 (2.37), 38	.587
2	7.24 (3.17), 38	6.72 (3.19), 36	.489
3	5.73 (3.40), 37	7.00 (3.20), 36	.105
4	5.68 (2.47), 34	5.48 (3.07), 33	.779
5	7.90 (4.39), 31	6.84 (2.66), 32	.250
6	7.50 (1.38), 7	6.67 (1.56), 9	.285

Abbreviation: std, standard deviation.

TABLE 8 Results of the quality survey for the subjects.

Item	Range	Course 21/22mean (std), N	Course 19/20mean (std), N	<i>t</i> test <i>p</i> Value
P1. The lecturer provides clear information about the subject at the beginning of the course.	[0, 5]	4.8 (0.4), 50	4.15 (0.83), 26	<.0001
P2. The lecturer explains in a clear and organized way.	[0, 5]	4.78 (0.42), 50	4.35 (0.89), 26	.0054
P3. The lecturer motivates the students and arouses interest in the subject.	[0, 5]	4.5 (0.61), 50	3.52 (1.05), 25	<.0001
P4. The methodology used in the subject helps me to learn the content provided in the program.	[0, 5]	4.6 (0.61), 50	4.16 (0.94), 25	.0169
P5. The practices help to better understand the theoretical contents.	[0, 5]	4.2 (0.83), 50	3.67 (1.27), 24	.0348
P6. The resources used by the lecturer benefit my learning.	[0, 5]	4.52 (0.65), 50	4 (1.04), 25	.0097
P7. The lecturer has resolved the doubts that have been raised in class.	[0, 5]	4.8 (0.49), 50	4.62 (0.75), 26	.2116
P8. The students have been adequately cared for when they have attended tutoring hours.	[0, 10]	4.39 (0.83), 46	4.56 (0.53), 9	.5584
P9. Global satisfaction with the lecturer.	[0, 10]	9.32 (1.06), 50	7.77 (2.39), 26	.0002
P10. Global satisfaction with the subject.	[0, 10]	8.96 (1.31), 49	7.27 (2.41), 26	.0002

Note: Bold values show statistically significant differences.

Abbreviation: std, standard deviation.

approach in the student satisfaction, the results of course 21–22 were compared with those of course 19–20, before the introduction of the EVERSYS tool. The questions related to the evaluation of the exam-versioning methodology were P4 and P6 (methodology and resources used), as well as P9 and P10 (global satisfaction with the lecturer and subject). In all these cases (P4, P6, P9, and P10), a Student's *t* test shows that the marks obtained were significantly higher than those of the 19–20 course. According to these results, the third hypothesis, stating that students can accept exam versioning, should be accepted. There are also threats to validity in this case, since the improvement in student satisfaction may be caused by other factors. However, courses 19–20 and

21–22 were given by the same lecturer and using a similar methodology, except for the exams, so it seems reasonable to accept the hypothesis.

Furthermore, a survey was conducted to assess the effectiveness of the versioning methodology from the students' perspective. A total of 38 responses were collected. The first question inquired about the perceived impact of versioning on cheating, and 36 students (95%) agreed that versioning made cheating more difficult. The second question addressed whether students attempted to cheat in any of the exams, and 37 students (97%) indicated that they did not attempt to cheat. Table 9 presents the detailed results of the survey.

TABLE 9 Survey on the versioning methodology.

Question	Yes	No
1. Versioning made cheating more difficult	36 (95%)	2 (5%)
2. I tried to cheat at least once	1 (97%)	37 (3%)

6 | DISCUSSION

This study focused on versioning as a means of preventing cheating during exams, but there are other methods available. Strict control during the exam, with or without electronic aids, is one such approach [2, 3, 9]. Allocating sufficient but not excessive time for exams has also been shown to reduce cheating [24]. For exams with multiple problems, scheduling the problems in different time slots for different students can also help prevent cheating [13]. However, these approaches are not mutually exclusive; hence, they can be combined for greater effectiveness.

A completely different approach is the fully automatic generation of exam questions through generative artificial intelligence techniques, where the only input required from the lecturer is a text file describing the subject (e.g., a book chapter). This approach is available in different commercial products, like, for example, QuizBot [34] or QuizGecko [35]. Although, at present, such approaches are focused on theoretical subjects and they are not reliable for problem-solving subjects, the rapid evolution of artificial intelligence may allow the development of more powerful tools in the near future.

Automatic assessment has also been studied to increase the efficiency of exam correction. For example, in the study presented in Zampiroli et al. [30], the automatic assessment of programming exercises was analyzed, and it was concluded that automation improves the learning process because it increases the number and frequency of tests and allows feedback to be instantly provided to students. In the field of linear circuits and electronics, there have been proposals for the automatic evaluation of circuit designs [5, 15].

In other studies, the goal was not to generate or correct exams automatically but to provide other tools for lecturers. For example, the methodology presented in Jardim [11] focuses on language learning and natural language processing; lecturers are provided with a tool that can classify questions automatically depending on their difficulty level. Similarly, in another study [21], the optimization of the time devoted to preparing an exam was analyzed in the framework of e-learning for medical sciences, from both the students' and lecturer's perspectives. Other studies have focused on fraud detection after it has

happened [10, 12]. However, preventing fraud (instead of detecting it) appears to be a better option.

Our results indicated that the versioning strategy used in this study significantly limited cheating. However, it is unclear whether similar results would have been obtained with fewer versions, which would correspond to a smaller workload for the lecturer. Further experiments are needed to determine the optimal number of versions that strikes a balance between extra work and cheating avoidance.

Regarding exam preparation and correction times, the comparative values presented in Tables 4–6 are derived from estimations of the time required for specific tasks. It is important to acknowledge that different estimations may lead to slight variations in the results. Nevertheless, the main concept remains consistent: certain tasks require repetitive execution in manual versioning, whereas the EVERSYS tool enables these tasks to be performed just once.

Although the EVERSYS platform was designed for exams, it can also be adapted to create multiple versions of student assignments, which are often completed at home and are therefore more susceptible to cheating (i.e., copying from classmates or external sources). The simplest way of creating an assignment with the EVERSYS tool is straightforward: it just requires a specific heading and, optionally, more complex problems whose solving takes longer than the time available in an exam (an example is included as Supporting Information Material). However, there are aspects that can be improved in the tool. At present, the output is a single HTML or PDF document containing all the exam or assignment versions. This output is well suited for printing and distributing paper copies of the exam in the classroom, but not for electronically sending each version to individual students. To address this, it would be beneficial to have multiple PDF documents as the output, enabling seamless distribution of specific versions to each student. Such an improvement is currently under development as future work.

All the examples provided in the paper correspond to exams consisting of only one problem. However, it is common for exams, especially final exams covering the entire subject, to comprise multiple problems. Currently, the tool is designed to handle one problem at a time, which means generating exams with multiple problems

necessitates individually generating each problem. These can then be managed jointly or separately, depending on the preferences of the lecturer. Future work will consider new EVERSYS functionalities that can allow the management of such long exams (multiple problem exams) without the need to generate individually each problem. For a better understanding, the Supporting Information Material includes an example of a final exam (three problems) generated with the current EVERSYS tool.

An additional functionality that we plan to add to the EVERSYS tool is automatic result comparison, to detect possible copies between students, particularly when they are coincident but both wrong. The only requirement for automation is that students should upload their results, even in paper-based exams, as proposed in Fernández & Vicente [7].

Previous sections state that automatic versioning can minimize correction errors. In particular, with the EVERSYS tool, the lecturer only needs to solve one exam version per schema to obtain all the correct results. Without the tool, the lecturer should solve all exam versions independently to find the correct results of each version. Assuming that the probability of making a mistake while solving a problem is the same in both cases (same lecturer), it is much more likely to make a mistake when the number of exams to solve is higher. However, it has been impossible to gather enough data as to obtain adequate statistics about error minimization. Further studies may be conducted to gather information on the number of student claims or appeals after correction, which could provide an indicator of the probability of correction errors with and without the tool.

While not a specific functionality of the EVERSYS platform, the exams generated by the tool can be reused over time, even without modifications or creating new versions with slight changes. Future studies can explore the results obtained when repeating the same questions or schemes over time, potentially revealing an expected increase on average marks due to students preparing for the exams by solving previous test versions. This valuable insight could shed light on the impact of exam reuse and the benefits of leveraging past exam materials for students' preparation.

As a final thought, and from the students' perspective, it appears that they do not want their answers to be copied, even though they may feel pressured to let others see their work. This may be one reason for the high level of student satisfaction reported in the surveys. In future research, students could be asked whether they prefer a completely cheat-proof system.

7 | CONCLUSIONS

The first hypothesis explored in the study suggested that the EVERSYS platform could reduce both exam preparation and exam correction times, as compared with manual versioning. According to the results obtained, the reduction is statistically significant, with an average decrease of 28% in preparation times (from 480 to 345 min) and an average decrease of 51% in correction times (from 819.7 to 399.7 min). This reduction in workload enables lecturers to increase the number of tests given during a course, potentially improving student evaluation.

The second hypothesis stated that, when properly used, the EVERSYS versioning would not introduce bias in student marks. The analysis of the marks obtained by the students shows that, in none of the six exams carried out during the semester, there were statistically significant differences in student marks based on the exam version. Consequently, this hypothesis was also validated.

Finally, the third hypothesis suggested that students would accept tools like EVERSYS, focused on the reduction of cheating during exams. Analysis of student surveys, compared with previous years, indicated statistically significant improvements in aspects related to methodology, supporting the validity of the hypothesis.

Given that the EVERSYS platform is built on widely adopted languages like MATLAB/Octave and HTML, it can be easily adopted (in its current state) by most lecturers teaching linear circuit theory and similar subjects.

Future research will focus on expanding the use of this tool to other engineering subjects and further enhancing its usability, to make it accessible to a wider range of lecturers, including those without knowledge of MATLAB/Octave.

ACKNOWLEDGMENTS

This work has been funded by Miguel Hernandez University through project PIEU-A/2022/04.

DATA AVAILABILITY STATEMENT

Data is openly available upon request to the corresponding author.

ORCID

María Asunción Vicente  <http://orcid.org/0000-0002-8630-7251>

César Fernández Peris  <http://orcid.org/0000-0002-9391-9192>

REFERENCES

1. Analog Devices Inc, LTSPICE electronic circuit simulation software, 2023, available at <https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html>
2. S. Arnò, A. Galassi, M. Tommasi, A. Saggino, and P. Vittorini, *State-of-the art of commercial proctoring systems and their use in academic online exams*, *Int. J. Distance Educ. Technol.* **19** (2021), 55–76. <https://doi.org/10.4018/ijdet.20210401.0a3>
3. Y. Atoum, L. Chen, A. X. Liu, S. D. H. Hsu, and X. Liu, *Automated online exam proctoring*, *IEEE Trans. Multimedia* **19** (2017), no. 7, 1609–1624. <https://doi.org/10.1109/tmm.2017.2656064>
4. Cadence Design Systems, Inc., PSPICE electronic circuit simulation software, 2023, available at <https://www.pspice.com/>
5. P. Dang and H. Arolkar, *Automatic evaluation of analog circuit designs*, *Data Science and Intelligent Applications*. (2021) Lecture Notes on Data Engineering and Communications Technologies (eds) K. Kotecha, V. Piuri, H. Shah, and R. Patel, **52**, Springer, Singapore pp. 553–563. https://doi.org/10.1007/978-981-15-4474-3_60
6. J. W. Eaton, Octave, scientific programming language, 2022, available at <https://octave.org/>
7. C. Fernández and M. A. Vicente, *Tools for simultaneous online and in-person teaching in a linear circuit analysis subject*, *Comput. Appl. Eng. Educ.* **30** (2022), no. 6, 1774–1794. <https://doi.org/10.1002/cae.22555>
8. C. Fernández, M. A. Vicente, R. Puerto, and R. P. Neco, *Latex based tool for managing multiple choice question exams*, *Proceedings of the 3rd International Conference on Education and New Learning Technologies* (2011), Barcelona, Spain, 6115–6122.
9. M. Ghizlane, B. Hicham, and F. H. Reda, *A new model of automatic and continuous online exam monitoring*, *International Conference on Systems of Collaboration Big Data, Internet of Things & Security (SysCoBioTS)* (2019), Casablanca, Morocco. <https://doi.org/10.1109/SysCoBioTS48768.2019.9028027>.
10. S. Haque, S. Zeba, M. A. Haque, K. Kumar, and M. P. A. Basha, *An IoT model for securing examinations from malpractices*, *Mater. Today: Proc.* **81** (2023), no. 2, 371–376. <https://doi.org/10.1016/j.matpr.2021.03.413>
11. R. R. J. Jardim, C. Delgado, and M. F. Silva, *CLIQ! intelligent question classifier for the elaboration of exams*, *Softw. Impacts* **13** (2022), 100345. <https://doi.org/10.1016/j.simpa.2022.100345>
12. B. Küppers, J. Oppen-Rhein, T. Eifert, and U. Schroeder, *Cheating detection: identifying fraud in digital exams*, *European University Information Systems (EUNIS) 2019 Annual Congress* (2019), Trondheim, Norway.
13. M. Li, L. Luo, S. Sikdar, N. I. Nizam, S. Gao, H. Shan, M. Kruger, U. Kruger, H. Mohamed, L. Xia, and G. Wang, *Optimized collusion prevention for online exams during social distancing*, *npj Sci. Learn.* **6** (2021), no. 1, 5. <https://doi.org/10.1038/s41539-020-00083-3>
14. Narcea Producciones Multimedia, S.L., PhpDocx, DOCX document generation from PHP, 2023, available at <https://www.phpdocx.com/>
15. M. Nemeč, P. Procházka, and R. Fasuga, *Automatic evaluation of basic electrical circuits in education*, 2013 IEEE 11th International Conference on Emerging eLearning Technologies and Applications (ICETA) (2013), Stara Lesna, Slovakia, 299–304. <https://doi.org/10.1109/iceta.2013.6674447>
16. H. Pehlivan, C. Atalar, and S. Zavrak, *Development and implementation of an analysis tool for direct current electrical circuits*, *Comput. Appl. Eng. Educ.* **29** (2021), no. 5, 1071–1086. <https://doi.org/10.1002/cae.22361>
17. P. Photopoulos, C. Tsonos, I. Stavarakas, and D. Triantis, *Problem-based multiple response exams for students with and without learning difficulties*, *International Conference on Computer Supported Education*, **1624**. Springer, 2022. https://doi.org/10.1007/978-3-031-14756-2_18
18. O. Plathey, FPDF, PDF document generation from PHP, 2022, available at <http://www.fpdf.org/>
19. A. Rjoub, Y. Eyadat, A. Ghazawi, B. Tall, N. Sharou, and L. Mardeeni, *A multi-form multiple choice editor exam tool based on HTML website and artificial intelligence techniques*, *J Comput Sci* **5** (2009), no. 6, 405–412.
20. M. W. Romaniuk and J. Łukasiewicz-Wieleba, *Challenges of administering university examinations remotely during the COVID-19 pandemic*, *e-mentor* **90** (2021), no. 3, 22–31. <https://doi.org/10.15219/em90.1519>
21. M. Roszak, B. Sawik, J. Stańdo, and E. Baum, *E-learning as a factor optimizing the amount of work time devoted to preparing an exam for medical program students during the COVID-19 epidemic situation*, *Healthcare* **9** (2021), no. 9, 1147. <https://doi.org/10.3390/healthcare9091147>
22. G. Rusak and L. Yan, *Unique exams: designing assessments for integrity and fairness*, *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (2021), Virtual event, USA, 1170–1176. <https://doi.org/10.1145/3408877.3432556>
23. M. Schultz and D. L. Callahan, *Perils and promise of online exams*, *Nat. Rev. Chem.* **6** (2022), no. 5, 299–300. <https://doi.org/10.1038/s41570-022-00385-7>
24. M. Stadler, N. Kolb, and M. Sailer, *The right amount of pressure: implementing time pressure in online exams*, *Distance Educ.* **42** (2021), no. 2, 219–230. <https://doi.org/10.1080/01587919.2021.1911629>
25. C. St-Onge, K. Ouellet, S. Lakhal, T. Dubé, and M. Marceau, *COVID-19 as the tipping point for integrating e-assessment in higher education practices*, *Br. J. Educ. Technol.* **53** (2022), no. 2, 349–366. <https://doi.org/10.1111/bjet.13169>
26. The Apache Software Foundation, POI library, DOCX document generation from Java, 2022, available at <https://poi.apache.org/>
27. The Apache Software Foundation, PdfBox, PDF document generation from Java, 2023, available at <https://pdfbox.apache.org/>
28. The MathWorks, Inc, Matlab, programming and numeric computing platform, 2023, available at <https://es.mathworks.com/products/matlab.html>
29. The MathWorks, Inc, Matlab report generation library, PDF document generation from Matlab, 2023, available at <https://es.mathworks.com/products/matlab-report-generator.html>
30. F. A. Zampirolli, J. M. Borovina Josko, M. L. F. Venero, G. Kobayashi, F. J. Fraga, D. Goya, and H. R. Savegnago, *An*

experience of automated assessment in a large-scale introduction programming course, *Comput. Appl. Eng. Educ.* **29** (2021), no. 5, 1284–1299. <https://doi.org/10.1002/cae.22385>

31. Moodle. Open-source Course Management System, August 1, 2023, available at <https://moodle.org/>
32. TomaMix, Test Scrambling Service, available at <https://tomax.io/our-services/tomamix/>
33. Test-Maker, Versioning Tool for Multiple Choice Exams, available at <https://github.com/iandennismiller/test-maker>
34. QuizBot, AI Question Generator, available at <https://quizbot.ai/>
35. QuizGecko, Online Test Creator, available at <https://quizgecko.com/>

AUTHOR BIOGRAPHIES



María Asunción Vicente received her MS degree in Electronics Engineering from University of Valencia in 1999. In 2007 she obtained her PhD degree (extraordinary doctorate award) from Miguel Hernandez University. Since 2000, she is working as an Associate Professor at Miguel Hernandez University, Telematics Engineering area. Her teaching subjects include linear circuits theory and Android programming. She has carried research on innovation in engineering education, computational neuroscience, and computer vision. Currently, her research is focused on mobile health.



César Fernández Peris is an Associate Professor at Miguel Hernandez University, Telematics Engineering area, since 1997. He obtained his Industrial Engineering degree from the Polytechnical University of Madrid (Spain) and his PhD degree from Miguel Hernandez University in Elche. Currently, he teaches linear circuits theory and mobile development subjects. He has participated in

several Spanish and European research projects. His main research topics are machine learning, computer vision, and telemedicine. He is a cofounder of AppAndAbout, a technological spin-off of Miguel Hernández University.



Miguel O. Martínez received his MS degree in Computer Science from the University of Alicante in 1996. He worked for a multinational French-owned computer company as a data warehouse analyst. In 2003 he entered at Miguel Hernandez University where at present he works as an Associate Professor in the Computer Engineering Department. He obtained his PhD degree (extraordinary doctorate award) from Miguel Hernandez University in 2014. His teaching subjects include Operating Systems and Mobile Programming. His research subjects are related to image and video compression specifically with perceptual coding. Currently, his research is focused on the perceptual enhancement of the HEVC standard.

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: M. A. Vicente, C. Fernández Peris, and M. O. Martínez, *EVERSYS: Efficient exam versioning tool for linear circuits and other problem-based subjects*, *Comput. Appl. Eng. Educ.* (2024), e22715. <https://doi.org/10.1002/cae.22715>