Designing Parametric Multiple-Choice Tests with Moodle and LaTeX for Engineering Education

1st Rafael Terris-Gallego
Dept. of Telecommunications and Systems Engineering
School of Engineering, Universitat Autònoma de Barcelona
Bellaterra (Barcelona), Spain
rafael.terris@uab.cat

2nd Fran Fabra
Dept. of Telecommunications and Systems Engineering
School of Engineering, Universitat Autònoma de Barcelona
Bellaterra (Barcelona), Spain
franciscojose.fabra@uab.cat

3rd José A. López-Salcedo
Dept. of Telecommunications and Systems Engineering
School of Engineering, Universitat Autònoma de Barcelona
Bellaterra (Barcelona), Spain
jose.salcedo@uab.cat

4th Gonzalo Seco-Granados
Dept. of Telecommunications and Systems Engineering
School of Engineering, Universitat Autònoma de Barcelona
Bellaterra (Barcelona), Spain
gonzalo.seco@uab.cat

***Abstract*—E-learning has become a cornerstone of modern education, offering flexibility, accessibility, and cost-effectiveness. Among the available tools, Modular Object-Oriented Dynamic Learning Environment (Moodle) stands out as a robust Learning Management System (LMS) for managing online content, particularly through Multiple-Choice Questionnaires (MCQs), which provide scalable and objective assessments. However, managing large question banks or engineering-specific content in Moodle can be challenging.**

This paper demo how combining Moodle with LaTeX—using the `moodle` package— addresses these challenges. This approach enables efficient creation, management, and deployment of questions by leveraging LaTeX's formatting capabilities alongside Python or Lua scripting. Successfully implemented in Electrical Engineering courses, it benefits both students and educators by streamlining evaluation and improving learning flexibility.

***Index Terms*—E-learning, Moodle, LMS, LaTeX, Python, Lua, evaluation, test, multiple-choice, questionnaire, MCQ, parametric**

I. Introduction

E-learning has become increasingly important due to its flexibility, accessibility, and cost efficiency. It allows learners to access diverse, up-to-date content tailored to their needs and has been widely adopted to enhance teaching practices and reach broader audiences. By leveraging digital platforms and tools, e-learning enables self-paced learning and fosters collaboration through virtual classrooms.

Moodle is a widely used LMS for facilitating and enriching online education. It offers a robust platform for educators to efficiently create, deliver, and manage e-learning content and activities. It also provides valuable tools for evaluation—tasks that are often resource-intensive for educators.

Multiple-Choice Questionnaires are among the most effective assessment methods in e-learning, widely used in engineering education to efficiently evaluate students' understanding of complex concepts. They offer several advantages, such as broad topic coverage, automated grading, and the possibility of providing immediate feedback. Well-designed multiple-choice assessments also serve a formative purpose by helping students identify misconceptions early and focus on areas needing improvement, while providing instructors with insights to adjust their teaching. These pedagogical benefits are especially valuable in engineering courses, where conceptual understanding and timely feedback are essential.

However, creating and managing these questionnaires remains a time-consuming and complex. Implementing MCQs effectively requires maintaining a comprehensive question bank—a collection of categorized questions that can be reused and managed efficiently—and ensuring efficient management during their preparation and deployment. Moodle provides functionality to create and manage MCQs, including options for adding feedback and assigning scores. However, managing a large question bank can be cumbersome, especially when it comes to editing and updating questions. Furthermore, Moodle's interface can be limiting when it comes to creating complex questions, particularly those that require advanced mathematical notation or formatting.

Many of these challenges can be mitigated by integrating complementary tools such as LaTeX. Combined, they provide a reliable and efficient system for question management. LaTeX is a powerful typesetting system widely used in academia and engineering to create complex documents, particularly those involving advanced mathematical notation. The community has developed several LaTeX packages to support exam creation in general and MCQ in particular (see, for example, [9] and [10]). However, these packages rely solely on LaTeX and do not offer utilities for integration with Moodle.

In this paper, we present how these tools can be combined based on the `moodle` package. This approach draws on several years of experience implementing them in Electrical

This work has been partly supported by the AGAUR-ICREA Academia Program and by Spanish Agency of Research (AEI) under grant PID2023-152820OB-I00 funded by MICIU/AEI/10.13039/501100011033.

979-8-3315-0290-4/25/$31.00 ©2025 IEEE

Engineering courses. Results show measurable benefits for students, who can assess and evaluate their learning flexibly. Additionally, this system has proven extremely useful for educators, enabling rapid student evaluation not only through computer-based access but also using Moodle's offline test tool. Moreover, this method enables the generation of questions using the parametric capabilities of Python and Lua, allowing for an efficient expansion and management of the question bank. Beyond its technical efficiency, this approach supports educational alignment by promoting adaptive assessments that enhance conceptual understanding and reduce rote learning through variation and adaptivity.

This paper is organized as follows. Section II describes how to create MCQs directly in Moodle, outlining its advantages and limitations. Section III presents the integration of LaTeX with Moodle using the `moodle` package. Section IV details the generation of parametric questions with Python/Lua and LaTeX, emphasizing the resulting flexibility and scalability. Section VI summarizes the main contributions, and Section VII suggests directions for further work.

II. Multiple-Choice Questionnaires with Moodle

A multiple-choice question is a type of assessment that presents a question or statement followed by several possible answers. Moodle offers several options to handle them, including selecting one or more correct answer, providing answer-specific or general feedback, and assigning scores to each answer.

To create an MCQ, Moodle offers the ability to create a repository of questions known as the question bank. This allows educators to organize questions into categories (see Fig. 1), making it easier to manage large sets of questions. It also facilitates the reuse of questions across different assessments.

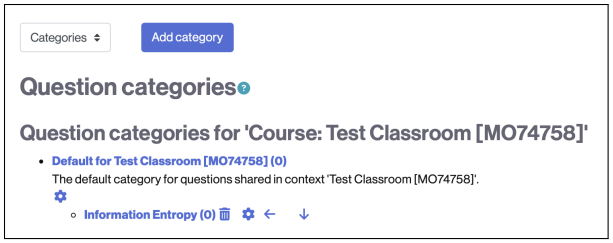

Figure 1. Question categories in Moodle.

Moodle allows the creation of different question types, including standard multiple-choice questions as well as calculated multi-choice questions [7], as shown in Fig. 2. The latter enables the use of parameters that can be randomly generated, allowing the creation of numerous variations of the same question. This is especially useful in engineering courses where students solve problems with varying inputs.

However, we will focus on regular multiple-choice questions, as calculated questions are not currently supported by the `moodle` package used to integrate LaTeX with Moodle.

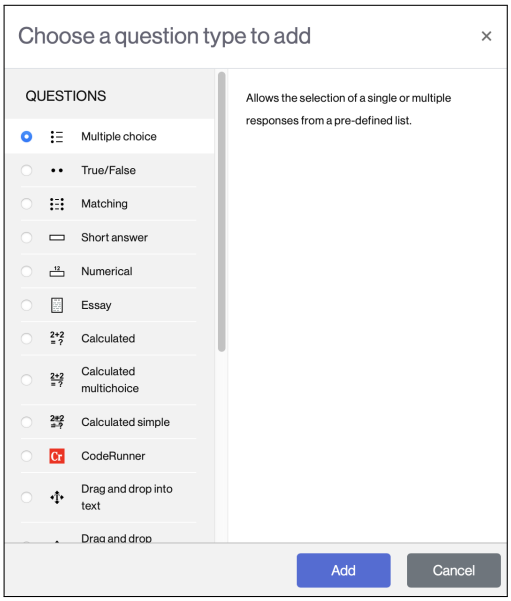

Figure 2. Question types in Moodle.

It is worth noting that TeX notation[1] can be used directly in Moodle, e.g. in question statements, as shown in Fig.3. This feature is useful for rendering mathematical notation directly within the question bank. However, Moodle uses a different LaTeX engine than standard compilers or editors, so LaTeX capabilities are limited. In particular, since Moodle 2.7, Moodle has used the MathJax JavaScript library to render TeX commands in browsers at display time [8].

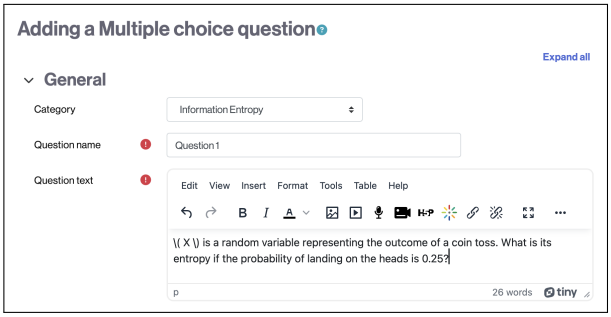

Figure 3. Title and description for mutiple-choice question in Moodle, using TeX notation.

Including the solution to the question is highly recommended, as it allows participants to review and compare their answers afterward. This can be achieved either by adding specific feedback to each answer option, or by providing general feedback for the entire question, as shown in Fig. 4.

The correct answer for a multiple-choice question is designated by assigning a score of 100%, as shown in Fig.5. For incorrect answers, several options are available. The simplest approach is to assign a score of 0%, which does not penalize

[1]For TeX notation in Moodle, it is recommended to enclose expressions with the operators "\(" and "\)" and "\[" and "\]" to ensure proper rendering, instead of the often used $...$ and $$...$$, respectively.

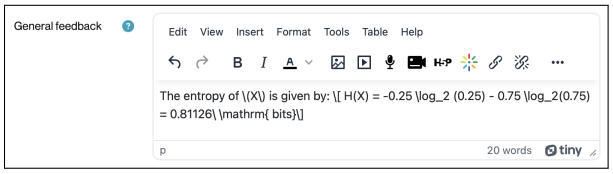

Figure 4. General feedback for mutiple-choice question in Moodle.

incorrect selections. However, this can lead to inflated scores due to random guessing. To address this, Moodle allows the assignment of negative scores to incorrect answers, which are subtracted from the total score. For example, with four possible answers and a negative score of one-third (see Fig.5), randomly guessing all questions would yield an average score of zero. This encourages thoughtful responses over guessing. It is also possible to assign a positive score—less than 100%—to answers that are partially correct.

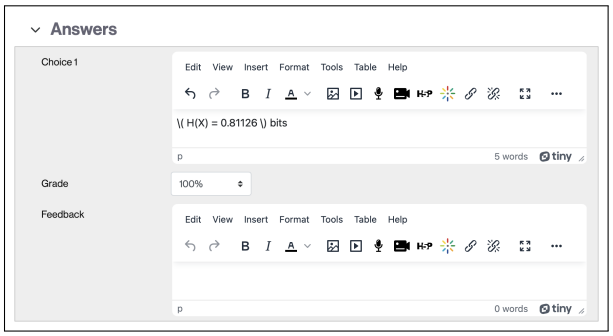

Figure 5. Correct option for multiple-choice question in Moodle.

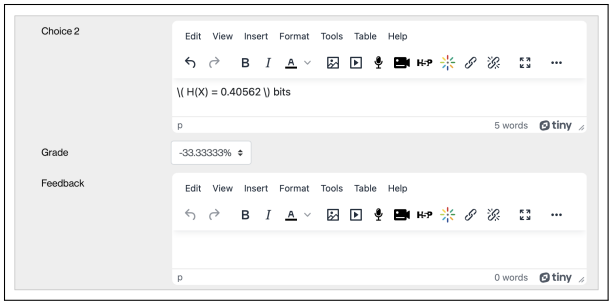

Figure 6. Incorrect option for multiple-choice question in Moodle.

When introducing the questions in the question bank it is also very useful to use tags to help identify the questions, as shown in Fig. 7. This is particularly important when the question bank contains a large number of questions, as it allows for easy filtering and searching. It also allow to group questions by topic, year of creation, or any other criteria relevant to the course. Combining the use of tags and categories enables a powerful and flexible way to manage the question bank. This is particularly useful when creating questionnaries, as it allows for easy selection of questions based on specific criteria.

Finally, Moodle allows previewing the questions created, as shown in Fig. 8. This enables comprehensive verification of the

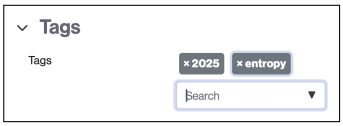

Figure 7. Selecting tags for the new question in Moodle.

question before adding it to the question bank, especially with regard to TeX notation, given the limitations mentioned above. The preview also includes the option to test the question, which is particularly useful for checking its behavior and the feedback provided.

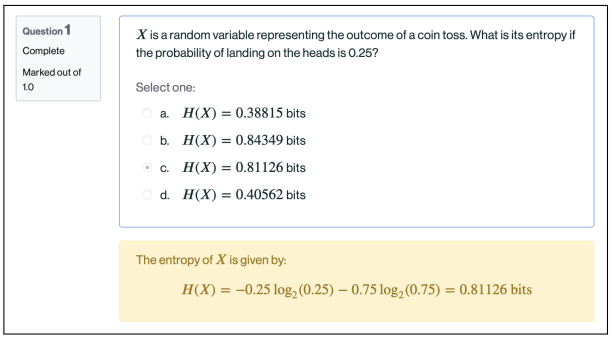

Figure 8. Preview of the created question.

While it is possible to build a complete question bank by repeating the process described above, certain limitations of Moodle's interface quickly become evident. While managing a small set of questions may be manageable, handling dozens or hundreds of them becomes increasingly tedious, particularly when entering answer options. Additionally, updating information across multiple questions requires editing each one individually. Fortunately, these limitations can be effectively overcome by using LaTeX in combination with the `moodle` package.

III. Integrating LaTeX with Moodle

A. Why LaTeX?

LaTeX is a flexible typesetting system, well-known for handling complex mathematical notation. It is particularly well-suited for engineering courses requiring complex expressions. It is also widely used in academic and research settings, making it familiar to many educators and students. The popular online platform Overleaf has become a standard for creating LaTeX documents, offering a wide range of features and robust collaboration tools. In its typical configuration, the left panel provides the editing interface, while the right panel displays the rendered output, as shown in Fig. 9.

Despite the complexity of LaTeX and the learning curve associated with it, very basic notions are sufficient to create a question bank. LaTeX is a markup language, meaning it uses tags to define the structure and formatting of the document, similar to HTML used for web pages.

LaTeX commands are easily recognizable because they begin with a backslash (\). Mandatory arguments are enclosed

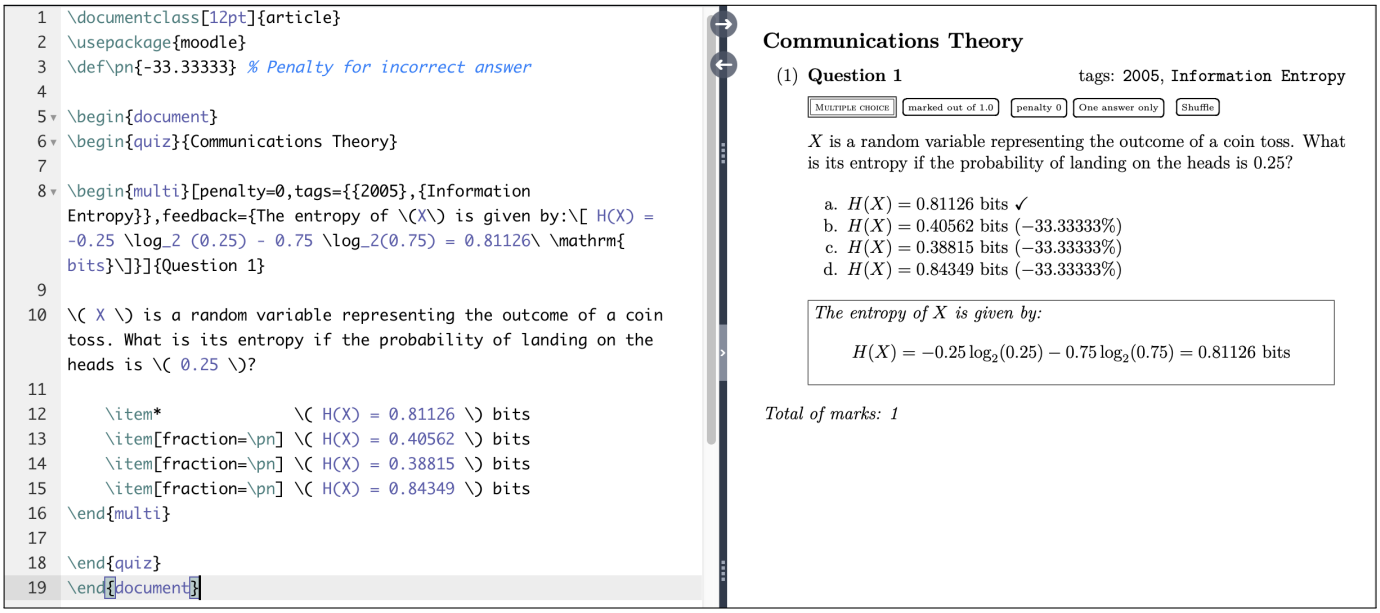

Figure 9. Creation and preview of a LaTeX document in Overleaf.

in curly braces ({}), while optional arguments are placed in square brackets ([]). Commands that begin with `\begin{}` create environments, which must be closed with the corresponding `\end{}` command. Interested readers can refer to [1] or [2] for comprehensive LaTeX documentation, and to [3] for resources specific to Overleaf.

It is also possible to load packages—additional features not included by default in LaTeX. These must be specified in the preamble of the document, which is the section before the `\begin{document}` command. In our case, it is essential to include the `moodle` package, which enables the use of questions created in LaTeX within the Moodle question bank.

B. The Moodle package

The `moodle` package [5] is a LaTeX package that allows the creation of questions in LaTeX format, which can then be imported into Moodle. Although it supports question types beyond multiple-choice, we will focus exclusively on the latter. Fortunately, the package is available by default on Overleaf.

To create an MCQ using the `moodle` package, we must first define a `quiz` environment, which takes the title of the questionnaire as its argument. An optional argument can be used to specify the question category. Within this environment, each question is defined using a `multi` environment, which requires the question title as a mandatory argument. Optional arguments include the penalty for each attempt (not to be confused with the penalty for an incorrect answer), general feedback, or tags. Finally, the possible answers are defined using the `\item` command. If the item is preceded by an asterisk (*), it marks the correct answer; otherwise, it indicates an incorrect answer, which can optionally be followed by a specified penalty.

The penalty for an incorrect answer is defined as a fraction of the total score, which is 100% by default (`[fraction=100]`). For example, if the penalty is set to -33.33333, selecting that answer will result in a deduction of one third of the total score. It is important to use the exact number of decimal places, as Moodle will not accept any value other than those specified in the official documentation [6]. To simplify this, it is convenient to define a variable for the penalty using the `\def` command in LaTeX, as shown in Listing 1. This avoids repeating the same value in each question and is particularly useful when creating a large number of questions, as it facilitates easy updates and modifications.

```latex
\documentclass[12pt]{article}
\usepackage{moodle}
\def\pn{-33.33333} % Penalty for incorrect answer

\begin{document}
\begin{quiz}{Communications Theory}

\begin{multi}[penalty=0,tags={{2005},{Information
    Entropy}},feedback={The entropy of \(X\) is given
    by:\[ H(X) = -0.25 \log_2 (0.25) - 0.75 \log
    _2(0.75) = 0.81126\ \mathrm{ bits}\]}]{Question 1}

\( X \) is a random variable representing the outcome of
     a coin toss. What is its entropy if the
    probability of landing on the heads is \( 0.25 \)?

  \item*              \( H(X) = 0.81126 \) bits
  \item[fraction=\pn] \( H(X) = 0.40562 \) bits
  \item[fraction=\pn] \( H(X) = 0.38815 \) bits
  \item[fraction=\pn] \( H(X) = 0.84349 \) bits
\end{multi}

\end{quiz}
\end{document}
```

Listing 1. LaTeX document for generating an MCQ using the `moodle` package.

C. Importing from LaTeX to Moodle

The compilation of a LaTeX document that loads the `moodle` package generates, alongside the expected PDF output, an eXtensible Markup Language (XML) file named `output-moodle.xml`. This file can be obtained directly from Overleaf using the "Download other output files" option, as shown in Fig. 10.

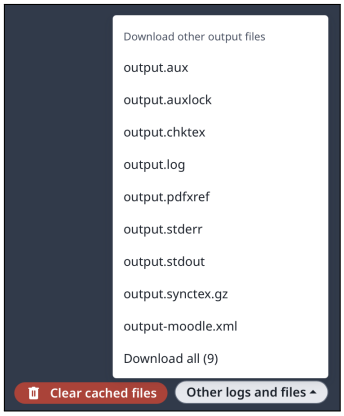

Figure 10. Downloading the XML file for Moodle from Overleaf.

The generated XML file contains all the questions defined in the LaTeX document—including their titles, descriptions, feedback, scores, and answer optio—formatted according to Moodle specifications. This file can therefore be imported directly into the Moodle question bank, as shown in Fig. 11.

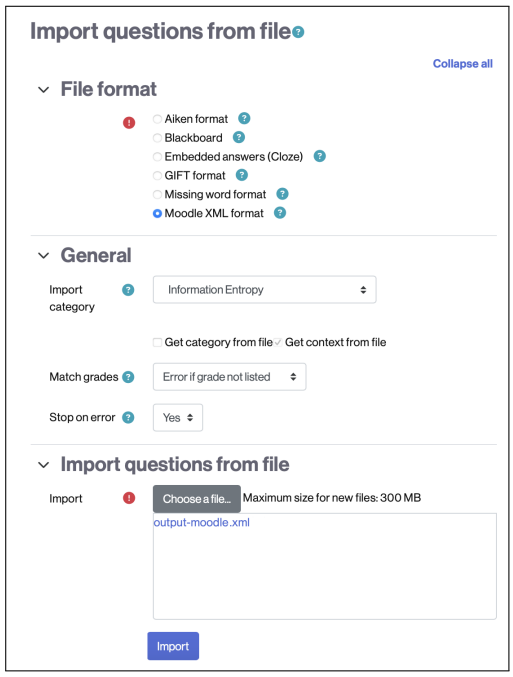

Figure 11. Importing the XML file generated with LaTeX into Moodle.

The import dialog also allows manually specifying the category where the questions will be stored, but Moodle can extract this information directly from the XML file if it is specified in the original LaTeX document using the optional argument of the `quiz` environment, as mentioned previously. If the XML file contains no syntax errors, Moodle will proceed with the import, as shown in Fig. III-C, and the question will be successfully added to the question bank.

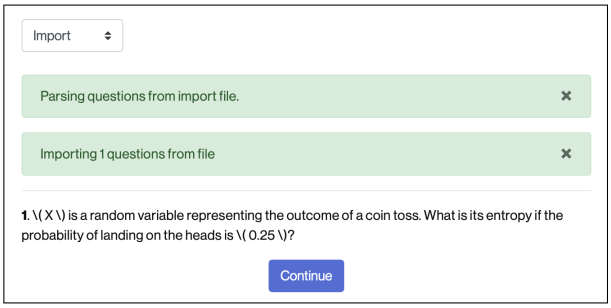

Figure 12. Successful import of the XML file.

This approach has several advantages over entering questions directly in Moodle. Some of the most significant are:

- It simplifies question editing, as a LaTeX document is not constrained by Moodle's interface limitations. A LaTeX file is essentially a plain text document that can be edited with any text editor, even offline. This is particularly useful when collaborating with other instructors on question development.
- It enables the efficient creation of similar questions by modifying only the relevant values or data directly in the LaTeX file. This can be done easily using the find-and-replace function included in text editors[2]. In contrast, Moodle requires editing each question individually through a rigid user interface.
- It allows for much faster question creation by simply copying and pasting the code between the `\begin{multi}` and `\end{multi}` commands in the LaTeX document—an operation far quicker than using Moodle's interface.

D. Creating Multiple-choice Tests from the Question Bank

Once the required number of questions have been imported into Moodle's question bank, they can be used to create a questionnaire. This is done by adding a quiz-type resource in Moodle, as shown in Fig. 13. In addition to the standard quiz, there is also an H5P quiz option—which allows the integration of interactive elements using HTML5—and an offline quiz option—which enables the questionnaire to be printed on paper and the responses scanned later. The latter is particularly useful when computers are unavailable or when physical copies are required due to specific conditions, and it leverages the question bank created in Moodle.

Finally, questions can be selected directly from the question bank, as shown in Fig. 14. To facilitate question selection, filters can be applied by specifying a category and/or relevant tags. This approach provides a high level of control and flexibility when multiple-choice tests are designed.

[2]This process can be further enhanced by using parametric questions,as described in Section IV

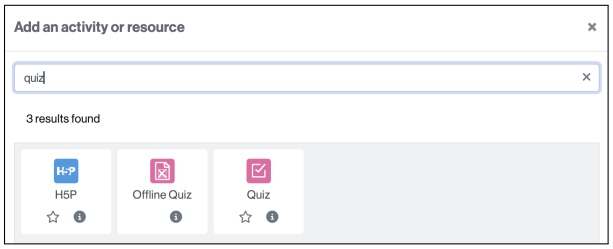

Figure 13. Adding a "quiz" resource in Moodle.

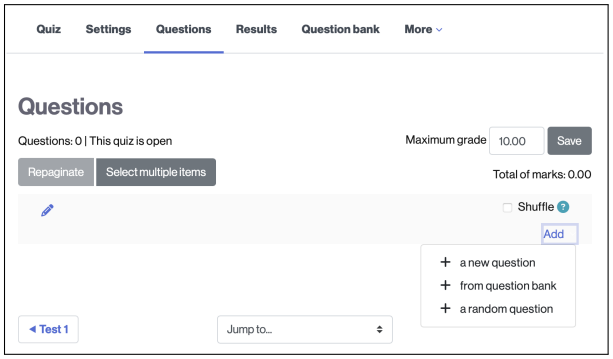

Figure 14. Adding questions to the quiz.

It is important to note that the `moodle` package has certain limitations. As previously mentioned, not all LaTeX commands are supported, and some features may not function as expected. Additionally, the package contains several bugs and issues that should be taken into account (see Section 5 of [5]).

Fortunately, the package is actively maintained, and many of the most significant issues present in earlier versions—such as characters not rendering correctly—have already been resolved. Although not all Moodle options are supported, new features are added regularly. The latest version of the package is available on Overleaf and is recommended for ensuring compatibility with the most recent version of Moodle. Nevertheless, some issues related to the quality of images generated by the `moodle` package are still being reported.

IV. Parametrizing Questions

The previous section has shown how to create a multiple-choice questionnaire using the `moodle` package, which leverages LaTeX editing capabilities. This section explores how to create parametric questions, which allows generating a large number of questions with different values. However, LaTeX by itself is not a programming language, but rather a typesetting system. Therefore, without any other assistance, it lacks native support for variables or functions in the same way that programming languages do.

The LaTeX user community has developed several packages that enable the use of variables and functions within LaTeX documents. The most widely used among them are `pgf` [13] and `fp` [14]. However, these packages can not be combined with the `moodle` package for creating parametric questions in Moodle, as the variables remain unparsed.

To overcome this limitation, a pre-processing approach is required, in which a programming language is used to generate the LaTeX code for the questions. This enables parametric question creation by defining variables and functions in code and generating corresponding LaTeX. In this context, Python is one of the most widely used programming languages, particularly in academia and research.

The use of Python allows for leveraging its extensive libraries and packages, which provide a wide range of functionalities for engineering education, making it well-suited for generating parametric LaTeX content. The workflow for generating parametric MCQs using Python, LaTeX, and Moodle is shown in Fig. 15.

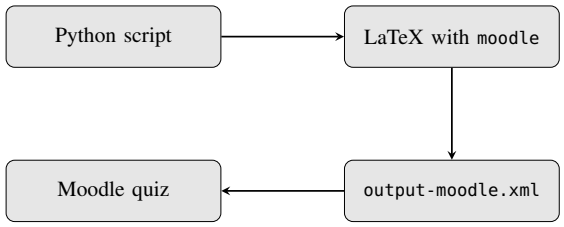

Figure 15. Workflow for generating parametric MCQs using Python scripts, LaTeX, and Moodle.

Several packages allow Python code to be processed within LaTeX documents, such as `pythontex` [11] and `pyluatex` [12]. These packages enable Python code to be embedded in a LaTeX document and executed during compilation to generate the final output. Unforunately, these packages are not currently supported on latest version of Overleaf (v2). This is because of a change to the security model in the last version (as opposed to v1), that prevents running arbitrary code on the server [4].

Therefore, using Python with Overleaf usuarlly requires maintaining separate Python scripts and LaTeX code. As an alternative, Lua—a lightweight programming language that can be embedded directly into LaTeX documents [15]—can be used. This enables the execution of Lua code within the LaTeX source, allowing parametric questions to be generated without the need for external scripts. Nevertheless, Lua's capabilities are more limited compared to those of Python.

Next, three different approaches for generating parametric MCQs using LaTeX are presented. The first two rely on Python scripts, while the third is based on Lua coding. Table I summarizes the main features of each approach, highlighting their strengths and weaknesses.

A. Approach 1: Python script to generate LaTeX document

The first approach consists of using a Python script to generate the LaTeX code, which can then be compiled to obtain the final LaTeX document that produces the required XML file to be imported in Moodle.

In Listing 2, a simple Python script is presented that generates a LaTeX document containing a multiple-choice question. The script uses the `math` library to compute the entropy of the random variable with the given probability `p`.

Table I
COMPARISON OF APPROACHES FOR GENERATING PARAMETRIC MCQS

Approach	1: Python script generating LaTeX	2: Python script directly in Overleaf	3: Lua code in LaTeX
Ease of Use	Requires running Python script separately before LaTeX compilation.	Directly integrates Python script execution in Overleaf.	Fully embedded in LaTeX, no external script required.
Flexibility	Python's full capabilities can be used.	Python's full capabilities can be used.	Limited to Lua's capabilities.
Overleaf Compatibility	Fully compatible but requires manual upload of generated LaTeX file.	Fully compatible.	Fully compatible, no external dependencies.
Complexity	Two-step process (Python + LaTeX).	Single-step process using Overleaf.	Single-step process using LuaLaTeX.
Dependencies	Requires external Python interpreter.	Requires Python interpreter in Overleaf.	Requires LuaLaTeX engine.

The script also generates two random values for the incorrect answers, using the `random` library.

```python
import math, random

p = 0.25
q = 1 - p
H = -p*math.log2(p)-q*math.log2(q)
H_half = H/2
H_r1, H_r2 = random.uniform(0,1), random.uniform(0,1)

with open("latex-question-python-generated.tex", "w") as texfile:
  texfile.write(rf"""\documentclass[12pt]{{article}}
\usepackage{{moodle}}
\def\pn{{-33.33333}} % Penalty for incorrect answer

\begin{{document}}
\begin{{quiz}}{{Communications Theory}}

\begin{{multi}}[penalty=0,tags={{{{2005}},{{Information Entropy}}}},feedback={{The entropy of \(X\) is given by:\[ H(X) = -{p} \log_2 ({p}) - {q} \log_2 ({q}) = {H:.5f}\ \mathrm{{bits}}\]}}]{{Question 1}}
\( X \) is a random variable representing the outcome of a coin toss. What is its entropy if the probability of landing on the heads is \( {p} \)?

  \item*              \( H(X) = {H:.5f} \) bits
  \item[fraction=\pn] \( H(X) = {H_half:.5f} \) bits
  \item[fraction=\pn] \( H(X) = {H_r1:.5f} \) bits
  \item[fraction=\pn] \( H(X) = {H_r2:.5f} \) bits
\end{{multi}}

\end{{quiz}}
\end{{document}}""")
```

Listing 2. Python script `generate_latex_document`.

Once the script is executed, a LaTeX document named `latex-question-python-generated.tex` is generated. The resulting document is identical to that shown in Listing 1, but the entropy value and the incorrect answers are produced by the Python script, making the question parametrizable. This LaTeX document can then be compiled in Overleaf to produce the XML file required for Moodle.

B. Approach 2: Python script to generate LaTeX question in Overleaf

The first approach implies compiling the python script to obtain the LaTeX document. To avoid this precompilation, a second approach is presented, in which the Python script is directly executed from Overleaf, using the following command:

```
\input{|python generate_latex_code.py}.
```

In this approach, the Python script only generates the code for the questions, rather than the whole LaTeX document, as shown in Listing 3.

```python
import math, random

p = 0.25
q = 1 - p
H = -p*math.log2(p)-q*math.log2(q)
H_half = H/2
H_r1, H_r2 = random.uniform(0,1), random.uniform(0,1)

print(rf"""
\begin{{multi}}[penalty=0,feedback={{The entropy of \(X\) is given by:
\[H(X)=-{p}\log_2({p})-{q}\log_2({q})={H:.5f}\ \mathrm{{bits}}\]}}]{{Question 1}}

\(X\) is a random variable representing the outcome of a coin toss. What is its entropy if the probability of landing on heads is \({p}\)?

  \item* \(H(X)={H:.5f}\) bits
  \item[fraction=\pn] \(H(X)={H_half:.5f}\) bits
  \item[fraction=\pn] \(H(X)={H_r1:.5f}\) bits
  \item[fraction=\pn] \(H(X)={H_r2:.5f}\) bits
\end{{multi}}
""")
```

Listing 3. Python script `generate_latex_question`.

Finally, in Listing 4, the LaTeX document which uses the previous script is shown. When executed in Overleaf, it produces directly the XML file to be imported in Moodle.

```latex
\documentclass[12pt]{article}
\usepackage{moodle}

\def\pn{-33.33333} % Penalty for incorrect answer

\begin{document}
```

```latex

\begin{quiz}{Communications Theory}
  \input{|python generate_latex_question.py}
\end{quiz}

\end{document}
```

Listing 4. LaTeX document based on Python script using Overleaf.

C. Approach 3: Lua code in LaTeX

The third approach consists of using LuaLaTeX, which allows the execution of Lua code directly within the LaTeX document. This approach is particularly useful when using Overleaf, as it does not require any external scripts or precompilation. Listing 5 shows the code to generate the same question as in the previous examples.

```latex
\documentclass[12pt]{article}
\usepackage{moodle}
\usepackage{luacode}

\def\pn{-33.33333} % Penalty for incorrect answer

\begin{document}
\begin{quiz}{Communications Theory}

\begin{luacode*}
function generate_mcq(title, description, tags, feedback, correct, wrong)
  tex.print("\\begin{multi}[penalty=0,tags={" .. tags .. "},feedback={" .. feedback .. "}]{ " .. title .. "}")
  tex.print(description)
  tex.print("\\item* " .. correct)
  for _, opt in ipairs(wrong) do tex.print("\\item[fraction=\\pn] " .. opt) end
  tex.print("\\end{multi}")
end

-- Auxiliary functions
function round(val, n) return math.floor(val * 10^n + 0.5) / 10^n end
function log2(x) return math.log(x) / math.log(2) end
\end{luacode*}

% Question 1
\begin{luacode*}
  math.randomseed(os.time())

  p = 0.25
  q = 1 - p

  local H = -p * log2(p) - q * log2(q)
  local H_half = H / 2
  local H_r1 = math.random()
  local H_r2 = math.random()

  local title = "Question 1"
  local description = "\\( X \\) is a random variable representing the outcome of a coin toss. What is its entropy if the probability of landing on the heads is ".. p .."?"
  local tags = "{2005},{Information Entropy}"
  local feedback = "{The entropy of \\(X\\) is given by :\\[ H(X) = -".. p .."\\log_2 (".. p ..") - ".. q ..")\\log_2(".. q ..") = ".. round(H,5) ..
"\\ \\ \\mathrm{ bits}\\]}"

  generate_mcq(title, description, tags, feedback, round(H,5), {round(H_half,5), round(H_r1,5), round(H_r2,5)})
\end{luacode*}

\end{quiz}
\end{document}
```

Listing 5. LaTeX document based on Lua code.

V. Preliminary Evaluation

At the end of the 2024–2025 academic year, a survey was conducted among students of the Communications Engineering course to assess their satisfaction with the use of MCQs. One of the survey items asked students to rate their agreement with the statement: "I consider that the MCQs are an efficient tool for the course." Responses were given on a 5-point scale, where 1 indicated the lowest level of agreement and 5 the highest. As shown in Fig. 16, 86.4% of students rated the MCQs with a score of 4 or 5, while only 4.6% rated them with a score of 1 or 2, indicating a high level of satisfaction.

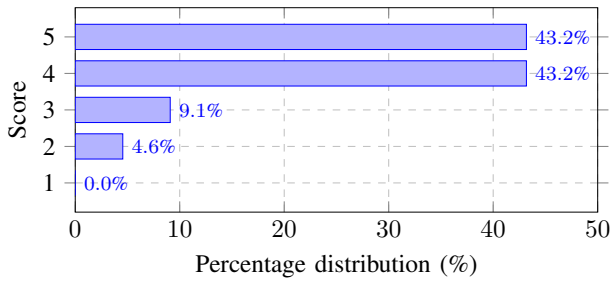

Figure 16. Satisfaction scores of students surveyed regarding the efficiency of the MCQs in the course (1 = lowest, 5 = highest).

VI. Conclusion

In this paper, a method for efficiently creating parametric Multiple-Choice Questionnaires by combining Moodle, LaTeX, and Python has been presented. First, the process of creating multiple-choice questions directly in Moodle was discussed, with emphasis on its strengths and limitations. Subsequently, the integration of LaTeX with Moodle through the `moodle` package was explored. This integration not only enables efficient question management but also offers clear advantages for engineering courses, where LaTeX's advanced mathematical capabilities are particularly beneficial.

Next, three approaches for generating parametric questions by leveraging the programming capabilities of Python and Lua were presented. The first approach involves the use of a Python script to generate a complete LaTeX document, which can then be compiled to produce the XML file required by Moodle. The second approach allows the Python script to be executed directly within Overleaf, generating only the LaTeX code for the questions without requiring precompilation. Finally, the

third approach employs LuaLaTeX, enabling Lua code to be executed directly within the LaTeX document.

No single approach fits all needs, as each offers distinct advantages and limitations. The first two approaches provide greater flexibility and control over the question generation process, while the third is simpler and more straightforward to implement. Nonetheless, all three methods are far more efficient than using Moodle alone.

While designed with engineering education in mind, these methods are applicable to any discipline that utilizes multiple-choice questions, making them a valuable resource for educators across a wide range of disciplines.

Preliminary evaluations indicate that students perceive MCQs as an effective learning tool, with a significant majority expressing satisfaction with their use in the course. These findings suggest that the methods presented not only enhance the efficiency of question creation but also contribute positively to students' learning experiences.

VII. Future Work

Future work includes enhancing the integration of Python with LaTeX to streamline parametric content generation workflows. The authors are also considering contributing to the development of the `moodle` package to extend its functionality with more advanced features.

Moreover, the use of parametric questions will be expanded beyond multiple-choice formats to include open-ended and essay-type questions, aiming to support more comprehensive evaluations—particularly in engineering courses that emphasize problem-solving.

A formal pedagogical evaluation is currently underway through anonymous student surveys to assess usability, engagement, and perceived learning outcomes. The findings will guide further refinement and help adapt the method to maximize educational effectiveness.

Acknowledgments

The authors would like to thank Anders O.F. Hendrickson and Matthieu Guerquin-Kern, as well as all contributors to the `moodle` package, for their work in developing a tool that has greatly facilitated the creation of multiple-choice questions in LaTeX. The authors also thank the students of the Communications Engineering course at the Universitat Autònoma de Barcelona for their valuable feedback and participation in the survey.

References

[1] LaTeX Project, *LaTeX2e Documentation*. Available: https://www.latex-project.org/help/documentation/, Accessed: 2025-03-01.

[2] Mittelbach, Frank and Goossens, Michel, *The LaTeX Companion*, 2nd edition, Addison-Wesley, 2004. Available: https://www.latex-project.org/help/books/tlc3-digital-chapter-samples.pdf.

[3] Overleaf, *Overleaf Documentation*. Available: https://www.overleaf.com/learn, Accessed: 2025-03-01.

[4] Overleaf, *Overleaf v2 FAQ*. Available: https://www.overleaf.com/learn/how-to/Overleaf_v2_FAQ. Accessed: Accessed: 2025-03-01.

[5] Hendrickson, Anders, *The moodle package: generating Moodle quizzes via LaTeX*, 2016. Available: https://ctan.math.utah.edu/ctan/tex-archive/macros/latex/contrib/moodle/moodle.pdf.

[6] Moodle, *Moodle Documentation*. Available: https://moodle.org/docs/, Accessed: 2025-03-01.

[7] Moodle, *Calculated Questions*. Available: https://docs.moodle.org/405/en/Calculated_multichoice_question_type, , Accessed: 2025-03-01.

[8] Moodle, *Using TeX notation*. Available: https://docs.moodle.org/401/en/Using_TeX_Notation, Accessed: 2025-03-01.

[9] Verna, Didier, *QCM – A LaTeX2e package for making Multiple Choices Questionnaires*, 2004. Available: https://ctan.javinator9889.com/macros/latex/contrib/qcm/qcm.pdf.

[10] Messineo, Grazia and Vassallo, Salvatore, *Package esami*, 2024. Available: https://ctan.fisiquimicamente.com/macros/latex/contrib/esami/doc/esami-doc-en.pdf.

[11] Poore, Geoffrey M., *PythonTeX Package*, 2021. Available: https://ctan.fisiquimicamente.com/macros/latex/contrib/pythontex/pythontex.pdf, Accessed: 2025-03-01.

[12] Enderle, Tobias, *The pyluatex package*, 2024. Available: https://ctan.fisiquimicamente.com/macros/luatex/latex/pyluatex/pyluatex.pdf, Accessed: 2025-03-01.

[13] Tantau, Till, *The PGF package*, 2023. Available: https://ctan.fisiquimicamente.com/graphics/pgf/base/doc/pgfmanual.pdf, Accessed: 2025-03-01.

[14] Mehlich, Michael, *The FP package*, 2019. Available: https://ctan.fisiquimicamente.com/macros/latex/contrib/fp/documentation.pdf, Accessed: 2025-03-01.

[15] Overleaf, *An Introduction to LuaTeX*. Available: https://www.overleaf.com/learn/latex/Articles/An_Introduction_to_LuaTeX_(Part_1)%3A_What_is_itâĂŤand_what_makes_it_so_different%3F, Accessed: 2025-03-01.