ISBN: 978-628-96613-3-0. ISSN: 2414-6390. Digital Object Identifier: https://dx.doi.org/10.18687/LACCEI2026.1.1.899

Preliminary Design Of An Interactive Platform For The Teaching Of Digital Logic In University Education

Witman Alvarado-Diaz[1]; Brian Meneses-Claudio[2]
[1,2]Universidad Cientifica Del Sur, Peru, *walvaradod@cientifica.edu.pe*

Abstract*—Currently, the integration of digital platforms in higher education represents a key strategy to strengthen the understanding of complex subjects, particularly in the areas of engineering. Digital logic, being a fundamental subject in the training of future engineers, requires pedagogical tools that facilitate both conceptual understanding and the development of practical skills. Faced with this need, the design of an interactive and intuitive web platform, developed in Python, is proposed, which aims to support both teachers and students in the teaching and learning process of digital logic courses. The platform incorporates structured modules of theoretical content, interactive practices, visual simulations of digital circuits and academic monitoring panels customized according to the user profile (student, teacher, or administrator). In addition, it integrates an embedded visual simulator, an automated assessment system, and an academic management dashboard that allows teachers to manage courses, assign activities, and track student progress in real time. The design of this tool seeks to bridge the gap between theoretical knowledge and practice through accessible and adaptive resources that adjust to various levels of mastery, from beginners to advanced students progressively. Likewise, the requirements, the architecture of the system and the curricular mapping of the digital logic course are defined, establishing the bases for its future implementation and experimental validation.

Keywords*—Digital logic; interactive platforms; educational technology; Python; engineering education; circuit simulation; online teaching.

I. INTRODUCTION

Digital logic is an essential component in programs in electronic engineering, computer science, and related areas, as it allows an understanding of the fundamentals of hardware design and digital systems. [1] However, traditional approaches, based on lectures and physical labs, often limit the flexibility and accessibility of learning. [2]

Numerous studies have shown the effectiveness of virtual laboratories and web platforms to improve engineering teaching, offering interactive environments, greater access to resources and immediate feedback [3] [4]. Tools such as Logisim and CircuitVerse have proven to be valuable in educational contexts, although they have limitations in curricular integration or scalability.

The use of digital platforms and interactive simulators in the teaching of digital logic has established itself as an effective strategy to improve learning in engineering and applied science students.

Recent studies highlight the effectiveness of virtual laboratories and simulators, reporting a significant positive effect on academic performance with a Hedges g of 0.686 ($g \approx 0.686$) which means that students who use simulators or virtual platforms have a better performance equivalent to a moderate-high effect, compared to those who learn only with traditional methods; In addition, a significant increase in students' motivation to learn from ($G \approx 3{,}571$) and an increase in student engagement ($G \approx 2{,}888$) is also shown, which shows that interactive tools favor learning.[3]

In the article [5], they show us that the application of logic gate simulators has shown notable improvements in learning outcomes, raising average scores from 76.37 to 85.82 points out of 100, along with an increase in student participation from 65.52% to 89.66%.

Additional comparisons confirm that students using simulators significantly outperform those who follow conventional methods, with averages of 0.59 versus 0.48 on final assessments. [6]

Finally, it should be considered that in student environments there is a heterogeneity in the digital capabilities of students, as shown in [7] a survey of more than two thousand university students where it was shown that only 37% reached high levels of digital learning achievement, while 9% showed low levels. These findings highlight the need for web platforms that are interactive, intuitive, and adaptable to various levels of student experience, from beginner to advanced.

II. DEVELOPMENT METHODOLOGY

Nowadays there are several methodologies for software development starting from the oldest such as the waterfall model [8], which is based on a sequential implementation by faces (requirements analysis, design, implementation, testing, deployment, and maintenance). Each phase must be completed before moving on to the next; This model is appropriate when you have well-defined requirements from the start.

Among other methodologies that we could apply are:

- Spiral Model. This model combines elements of the waterfall model, and gives them an incremental approach, this model involves planning, risk analysis, development, and evaluation. Enables early validation using prototypes [9].

- Incremental Model. The implementation with this model is developed with functional increments, until the final product is completed, it is a flexible model that allows partial and useful versions to be delivered before a definitive version. [10]
- Extreme programming (XP) This methodology is agile and focuses on software quality, including iterative development practices, continuous integration, automatic testing, and pair programming.[11]
- Scrum. It is an agile framework, based on short cycles called "Sprints"; Teams work iteratively and collaboratively, adapting quickly to changes. This methodology encourages self-organization and incremental value deliveries. [12]
- DevOps. It is a modern methodology that integrates development (Dev) and operations (Ops), in such a way that it improves collaboration, continuous delivery and automation of deployments. It allows us to reduce times and improve the quality of the final product. [13]
- Lean Software Development. It is a development model inspired by Toyota's production system, which focuses on eliminating waste, optimizing processes, and maximizing value for customers, thus providing light and flexible development cycles.[14]

To carry out the development of the interactive platform for teaching digital logic, the use of the Scrum methodology is proposed, because it is more flexible and adaptable, thanks to its iterative and incremental approach. This model will allow us to integrate early feedback from both students and teachers, thus guaranteeing that the final product we obtain responds to the real needs of the users of the system.

Each Scrum Sprint includes planning, designing, coding, testing, and presenting partial results. In this way, we will be able to make continuous adjustments without affecting the overall planning of the project.

For the development of this project, 5 sprints are proposed with a duration of 3 weeks each, this will facilitate to have enough time to integrate new functions, adjust the user interface and optimize the user experience according to the feedback received; In addition to promoting multidisciplinary collaboration between developers, pedagogues and teachers, which is considered an essential aspect in the development of educational platforms. Ensuring that the minimum viable product (MVP) is validated early and evolves iteratively towards a more robust version.

For the development of the software, the following schedule is proposed:

A. Planning and System Basis (Sprint 1)

This phase covers from week 1 to 3, here we define the requirements, architecture and backend base; To this end, the following activities are proposed: Survey of functional and non-functional requirements, define the architecture and database, make an initial design of the user interface, configure the development environment, create the basic authentication and profiles module. The deliverables for this section are requirements document; initial architecture implemented; prototype of a graphic interface; login system and functional roles.

B. Core Module Development (Sprint 2)

The main objective of this phase is the development of the core functionalities, among which we have: implementation of modules (courses, lessons, and exercises), integration of a virtual simulator, frontend and backend connection, creation of the main APIs (Application Programming Interface). The deliverables for this section are platforms with interactive modules; integrated simulator; Functional API.

C. Assessments and Follow-up (Sprint 3)

The main goal of this stage is to add evaluation and analytical tools. Among the main activities we have development of automatic evaluations, progress dashboard, reports for teachers and usability review. Deliverables for this stage include active evaluation module: statistics dashboard, exportable reports.

D. Optimization and Pilot Testing (Sprint 4)

In this stage, the main objective is to implement the tests, as well as to improve the performance and preparation of the pilot launch. The main activities to be developed in this stage are unit and integral tests, optimization of the user interface, final adjustments in the functionalities, and deployment of pilot version. The deliveries in this section are stable MVP platform, test report, and pilot version deployed.

E. Implementation and Feedback (Sprint 5)

This last stage covers from week 13 to 15, with the objective of launching the platform in a controlled way and applying improvements, for this the following activities are proposed: Tests with real users, collection of feedback, correction of errors, technical documentation. Among the deliverables of this stage are final validated platform, feedback report, complete documentation.

III. SYSTEM REQUIREMENTS

We must keep in mind that the system requirements must be clear, traceable, and verifiable; Based on this, the following are established: functional requirements, non-functional requirements.

A. Functional requirements

1. **Authentication and authorization:** it is composed of: Registration, login, and session management; roles: student, teacher, admin. You must have role-based authorization for resources and actions.
2. **User profile management:** composed of: Create and edit student profile (name, email, career, semester, photo, bio, password). The teaching section includes

affiliation and subjects; The admin section includes permission management.

3. **Course Management:** CRUD (create, read, update, delete) courses; each course has a title, description, instructor(s), calendar, list of lessons, and enrolled students.
4. **Lesson and content management:** CRUD of the lessons; the lessons contain: title, objectives, content, multimedia resources, formulas (LaTeX), and simulation area.
5. **Embedded simulator:** Area that integrates an interface to create and execute logic circuits in the browser; with the ability to upload examples and export designs.
6. **Workbench and Autocorrect:** Create Problems: Truth Tables, Boolean Simplification, K-maps, FSM Design, Basic Verilog; Basic Auto-Correct and Execution.
7. **Scheduled Assessments and Assignments:** Teachers will be able to create assignments with a deadline, rubric, and set of exercises; system collects submissions and calculates grades.
8. **Tracking and reporting:** Dashboard for student, teacher, and admin with metrics: progress, scores, time on task, attempts, simulator usage, student lists with grades ordered from highest to lowest.
9. **Forum, Comments, and feedback:** Comments by lesson and discussion forums by course, with notifications by activity.
10. **Administration and auditing:** Admin panel for user management, activity logs, backups, and global configuration. Auditing: Logging of key events.

B. Non-functional requirements

1. **Availability and performance:** The minimum expected availability should be 99% for production environment. The minimum response time must be less than 300; The simulator when running on the client should have no perceptible latency.
2. **Security:** Secure authentication, mandatory HTTPS, encrypted passwords, role-based access control, SQL injection protection, malicious script injection protection, encryption of all sensitive data.
3. **Privacy and compliance:** Storage and handling of data in accordance with institutional policies; options to anonymize data for research. Informed consent for data use.
4. **Usability and accessibility:** Responsive interface (desktop and tablet); Multilanguage (Spanish/English).
5. **Maintainability and modularity:** Code with unit tests, Modular architecture (separate APIs, frontend, simulator, analysis services).

IV. PROPOSED ARCHITECTURE

The proposed architecture for the web platform is presented in 4 levels that are presentation, application, data, and infrastructure level.

1. **Presentation Level:** It is the layer visible to the user, it includes graphic interfaces, dashboards, visual simulators, and results visualization modules. Its goal is to offer intuitive, fast, and engaging experience.
2. **Application Level:** This layer represents the functional core of the system, in other words, it is the "brain" that coordinates how data and requests are processed.
3. **Data Level:** It is responsible for storing and managing all the information in the system; Its mission is to ensure that information is secure, consistent, and accessible.
4. **Infrastructure Level:** It is the technical basis that supports the deployment, maintenance, and scalability of the system; its purpose is to ensure stability, availability, and ease of updating the system.

In the following graph we can see the proposed architecture, textually; It is worth mentioning that the proposed tools are not restrictive and could vary, according to the requirements or new difficulties that arise during the development of the project.

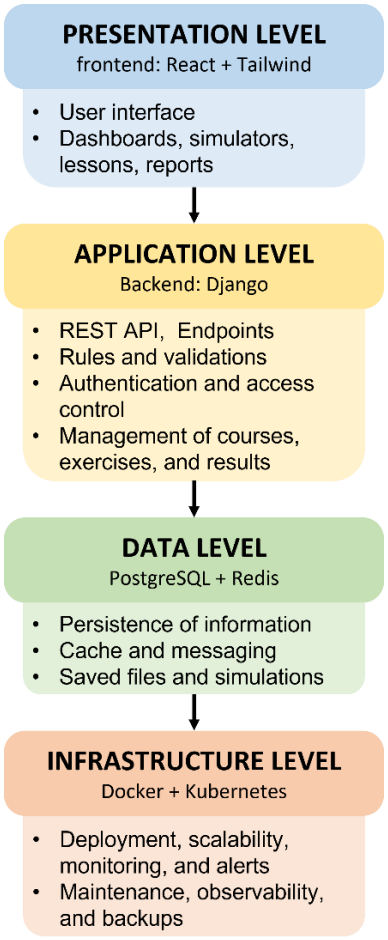

Fig. 1. Proposed architecture for the development of the project

V. PEDAGOGICAL DESIGN

The teaching of complex subjects such as digital logic, rather than the presentation of content, requires students to actively participate, build their own knowledge and be able to apply what they have learned in practical contexts. Methodologies must combine learning theories such as constructivism, active learning, and new orientations of digital pedagogy, aimed at designing relevant experiences, developing interactive content, implementing, and evaluating it in an iterative way, .[15][16][17]

A. Teaching-learning strategies and activities

- **Initial narrative**: Each module begins with a realistic scenario (for example: "Design the smart traffic light system with logic gates") to motivate.
- **Guided Exploration**: The simulator allows the student to manipulate circuits, change inputs, and visualize outputs.
- **Problem solving**: Challenge-type exercises (K-maps, sequential design) that require thinking, applying, and testing hypotheses.
- **Immediate self-assessment**: After each exercise, the platform shows grades, common errors, and suggestions.
- **Collaborative learning**: Forums/moderated by teacher where students discuss designs, share screenshots of the simulator, solve joint challenges.
- **Metacognition and reflection**: The system invites the student to reflect ("Why did the output change when I reversed these inputs?") and records brief reflection.
- **Adaptive progression**: If a student has difficulties in a topic (e.g., K-maps), the system suggests reinforcement modules or additional resources.
- **Light gamification**: Badges for completing modules, internal ranking to motivate.
- **Teacher follow-up**: Panel where the teacher identifies students with low performance and develops interventions (tutoring, recommendation of resources).

B. Evaluation, metrics and success indicators

Key performance indicators (KPIs) include: Percentage of students completing a module in less than 4 weeks, average score improvement between the first and last preference assessment greater than or equal to 20%, simulator participation rate ≥ 80%, average simulator use time per student ≥ 30 min, number of students detected at risk (score < 50%), User satisfaction (students and teachers) ≥ 4 on a scale of 1-5.

Assessment methods include: Pre-test and post-test to measure learning before and after the module, analysis of logs and progression of scores using control charts, qualitative survey (usability, motivation, perception of the platform), focused interviews with teachers for qualitative feedback; according to the literature, digital environments are expected to have a positive and significant effect on student performance [18].

C. Implementation of pedagogical timeline

- **Week 0 (pre-implementation):** Teacher training, system configuration, and pilot tests.
- **Weeks 1-4:** Module 1 (Fundamentals Boolean Algebra): Theory + Simulator + Exercises.
- **Weeks 5-8:** Module 2 (Combinational Design): theory + simulator + small projects.
- **Weeks 9-12:** Module 3 (Sequential Design): theory + simulator + projects for evaluation.
- **Weeks 13-16:** Reinforcement + final project + analysis of results.

VI. RESULTS

This section presents the results obtained after the design, modeling, and conceptual development of the web platform for the teaching of Digital Logic. It was possible to define an architecture aimed at guaranteeing scalability, modularity, and maintainability, which basically consists of Frontend, Backend,

Data Persistence, Infrastructure and Deployment, designed for the use of Docker containers and complete monitoring.

It was possible to build a model that defines the data structure of the system, obtaining entities such as: Users, Courses, Lessons, Exercises, Results, these entities guarantee complete traceability of the learning process and allow the implementation of progress reports.

As part of the results in the following image, it can see a preliminary dashboard for students, where the general progress, progress per lesson, and access to exercises, in addition to other basic preliminary options are shown. We can also see a menu that allows us to manage courses, create lessons and exercises, as well as visualize reports and performance statistics. This prototype works as a basis for the development of the final frontend and was designed with a minimalist and responsive approach.

Fig. 2. Preliminary system design dashboard

In the following image we can see a diagram that shows us the process of assignment and evaluation of exercises, which constitute one of the central flows of the web system, since it allows the integration of guided practice, self-evaluation, and immediate feedback within the virtual environment.

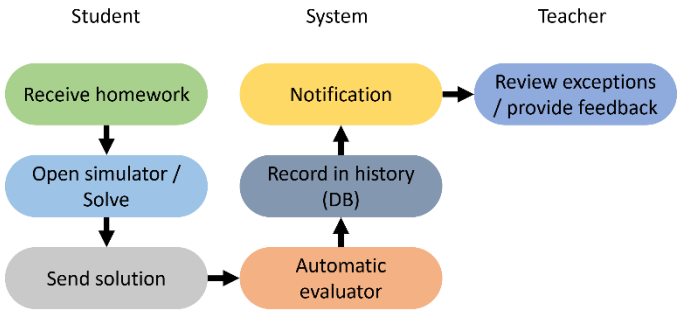

Fig. 3. Process of assignment of exercises, submission, automatic evaluation and feedback.

This flow seeks to improve the learning experience, optimize teaching time and strengthen the monitoring of student progress through evaluation algorithms based on predefined logical rules; The process begins when students receive an assignment which they must solve and send their answers, these will be automatically evaluated by the system, which will finally notify the teacher to provide feedback.

In the following table we show a preliminary catalog of classes, which constitutes a structural representation of the main objects of the system, their attributes and the relationships that allow the interaction between the different modules of the platform. This catalog serves as a reference for technical documentation, as well as precisely defines reusable, scalable, and maintainable components.

TABLE I. TABLE WITH KEY ATTRIBUTES, RESPONSIBILITIES AND RELATIONSHIPS.

Class	Attributes	Responsibilities	Relations
User	id, name, email, role, semester, profile	Log in, view courses, submit assignments, access simulator	Course, Result.
Course	id, title, description, instructor_id, start_date	Contain lessons, enrolling students, publish resources	Lesson, User.
Exercise	id, course_id, title, content, resources	Provide theoretical content, link simulator, and exercises	Exercise, Course.
Exercise	id, lesson_id, type, difficulty, solution_spec	Define activity, validate submissions, generate rubrics	Result, Lesson.
Result	id, exercise_id, user_id, score, attempts, time_spent, feedback	Record performance, store history, and generate reports	User, Exercise.
Simulation	id, user_id, lesson_id, circuit_json, inputs, outputs, runtime_ms	Save simulator state, play back circuits, allow export	User, Lesson
Report	id, course_id, metric, value, timestamp	Metrics aggregation, risk detection, and export	Course

VII. DISCUSSIONS

The development of the web platform for the teaching of Digital Logic integrates theoretical, pedagogical, and technological foundations with the purpose of improving the practical understanding of digital circuits through interactive simulation. Based on the principles of Morris Mano and Ciletti [1] and Roth et al. [2], the proposal seeks that the student constructs knowledge through experimentation, aligning with the National Research Council's vision [4] of active and reflective learning.

The incorporation of virtual laboratories has proven to be effective in the training of future engineers, as Li and Liang [3] and Finkelstein et al. [6] point out, by offering experiences equivalent to or superior to physical environments. In this line, the system's digital simulator allows autonomous practice and

understanding of abstract concepts, while automatic feedback strengthens the self-regulation of learning, as evidenced by the studies of Nur et al. [5] and Zhao et al. [7] on the motivation and power of digital learning.

From a pedagogical point of view, the platform adopts student-centered digital pedagogy approaches, proposed by Huang et al. [16], Thanaraj et al. [17] and Tan et al. [18], fostering autonomy, collaboration, and meaningful learning. Its instructional design guarantees an iterative process of analysis, design, development, implementation, and evaluation.

In the field of software development, agile methodologies such as Scrum and Extreme Programming (XP) were applied [11] [12], along with principles of Lean Development [14] and Continuous Delivery [13], to achieve an adaptable and high-quality product. The system architecture based on React, Django REST, PostgreSQL and Docker respond to the software engineering principles described by Pressman and Maxim [10] promoting continuous improvement and risk reduction.

Together, the project manages to articulate the theory of digital logic, modern pedagogy, and agile engineering methodologies, generating a sustainable, scalable digital learning environment focused on practical experience. This balance between pedagogy and technology consolidates an innovative educational model, aligned with the vision of sustainable digital transformation proposed by Huang et al.[16]

VIII. Conclusions

The development of the interactive web platform for the teaching of Digital Logic demonstrates that the integration of digital pedagogy, modern software engineering and active learning can significantly transform educational processes in engineering.

The modular architecture of the system, based on modern technologies, facilitates the scalability, interoperability, and maintenance of the software, ensuring efficient performance and adaptability to future expansions.

The project confirms that the use of digital environments not only complements but can enhance the conceptual and practical understanding of technical subjects. It also constitutes an innovative contribution to the field of engineering education, by combining theories of meaningful learning, agile methodologies, and user-centered design within the same technological framework.

Finally, the model developed is replicable and scalable for other areas of engineering, allowing its pedagogical and technological structure to be adapted to different disciplines. This work lays the groundwork for future research on the impact of simulation-based learning and the automation of educational assessment in virtual environments.

References

[1] M. Morris Mano and M. D. Ciletti, *Digital Design With An Introduction To The Verilog HDL*, 5th ed. Pearson Education, 2013.

[2] C. H. Roth, L. L. Kinney, and E. B. John, *Fundamentals of Logic Design*, 7th ed. 2021.

[3] J. Li and W. Liang, "Effectiveness of virtual laboratory in engineering education: A meta-analysis," *PLoS One*, vol. 19, no. 12, p. e0316269, Dec. 2024, doi: 10.1371/JOURNAL. PONE.0316269.

[4] N. R. Council, "How People Learn: Brain, Mind, Experience, and School: Expanded Edition," Aug. 1999, doi: 10.17226/9853.

[5] A. Nur, H. Herpratiwi, and R. Firdaus, "Effects of The Application of Logic Gate Simulator Media in Improving The Results and Learning Activities of XI Grade Vocational High School Students," *Jurnal Teknologi Pendidikan : Jurnal Penelitian dan Pengembangan Pembelajaran*, vol. 10, no. 2, p. 287, Apr. 2025, doi: 10.33394/jtp.v10i2.15211.

[6] N. D. Finkelstein *et al.*, "When learning about the real world is better done virtually: A study of substituting computer simulations for laboratory equipment," *Physical Review Special Topics - Physics Education Research*, vol. 1, no. 1, p. 010103, Oct. 2005, doi: 10.1103/PhysRevSTPER.1.010103.

[7] L. Zhao, J. Liu, A. Karimov, and M. Saarela, "Assessing and developing college students' digital learning power: an empirical study based on questionnaire survey in a Chinese university," *International Journal of Educational Technology in Higher Education*, vol. 22, no. 1, pp. 1–23, Dec. 2025, doi: 10.1186/S41239-025-00514-4/FIGURES/1.

[8] W. W. Royce, "Managing The Development Of Large Software Systems," in *Proceedings of IEEE WESCON*, 1970, pp. 1–9.

[9] B. W. Boehm, "A Spiral Model of Software Development and Enhancement," *Computer (Long Beach Calif)*, vol. 21, no. 5, pp. 61–72, 1988, doi: 10.1109/2.59.

[10] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 8th ed. McGraw-Hill, 2014.

[11] K. Beck and C. Andres, *Extreme Programming Explained*, 2nd ed.

[12] K. Schwaber and M. Beedler, *Agile Software Development with Scrum*. Prentice Hall, 2002.

[13] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley, 2010.

[14] M. Poppendieck and T. Poppendieck, *Lean Software Development: An Agile Toolkit*. Addison-Wesley, 2003.

[15] V. V. Senadheera, D. S. Ediriweera, and T. P. Rupasinghe, "Instructional Design Models for Digital Learning in Higher Education — A Scoping Review," *Journal of Learning for Development*, vol. 11, pp. 15–26, 2024.

[16] R. Huang, M. A. Adarkwah, M. Liu, Y. Hu, R. Zhuang, and T. Chang, "Digital Pedagogy for Sustainable Education Transformation: Enhancing Learner-Centred Learning in the Digital Era," *Frontiers of Digital Education 2024 1:4*, vol. 1, no. 4, pp. 279–294, Dec. 2024, doi: 10.1007/S44366-024-0031-X.

[17] A. Thanaraj, P. Durston, A. Chathangoth, and J. Sumpter, "Digital Learning Design Framework & Toolkit," Jul. 2022.

[18] S. C. Tan, J. Voogt, and L. Tan, "Introduction to digital pedagogy: a proposed framework for design and enactment," *Pedagogies*, vol. 19, no. 3, pp. 327–336, Jul. 2024.