Revista Colombiana de Computación
Vol. 26, No.2. July – Dec 2025, pp. 16-30
e-ISSN: 2539-2115, https://doi.org/10.29375/25392115.5184

REVISTA COLOMBIANA
DE COMPUTACIÓN

# Software development using project-based learning

## Desarrollo de software mediante aprendizaje basado en proyectos

Alan David Ramírez Noriega[1], Samantha Paulina Jiménez Calleros[2],
Herman Geovany Ayala Zuñiga[1], Yobani Martínez Ramírez[1]

[1] Universidad Autónoma de Sinaloa, México.
[2] Universidad Autónoma de Baja California, México.

alandramireznoriega@uas.edu.mx, Samantha.jimenez@uabc.edu.mx, hayala@uas.edu.mx, yobani@uas.edu.mx

(Received: 17 September 2024; Accepted: 26 December 2025, Published: 31 December 2025)

**Abstract.** Project-based learning is a work strategy that facilitates the development of real projects within an educational setting, enabling students to develop various skills in preparation for their professional careers. This approach is widely used in technological fields, including software engineering, with the goal of developing quality software systematically. This research presents a comparative study of groups utilizing project-based learning for software development in a team-based format. It involves work teams that replicate the roles found in software development. The experiment was conducted with two groups of Software Engineering students over a semester. The results assess the performance of both groups and highlight the generic and specific competencies of the career that are enhanced by applying project-based learning. This study aims to identify the characteristics that groups undertaking project-based learning should possess for effective team-based software development.

**Keywords:** Project-Based Learning, software development, software engineering.

**Resumen.** El aprendizaje basado en proyectos es una estrategia de trabajo que facilita el desarrollo de proyectos reales en el ámbito educativo, permitiendo a los estudiantes desarrollar diversas competencias como preparación para su carrera profesional. Este enfoque es ampliamente utilizado en campos tecnológicos, entre ellos la ingeniería de software, con el objetivo de desarrollar software de calidad de manera sistemática. Esta investigación presenta un estudio comparativo de grupos que utilizan el aprendizaje basado en proyectos para el desarrollo de software en un formato basado en equipos. Se trata de equipos de trabajo que replican los roles que se encuentran en el desarrollo de software. El experimento se realizó con dos grupos de estudiantes de Ingeniería de Software durante un semestre. Los resultados evalúan el desempeño de ambos grupos y resaltan las competencias genéricas y específicas de la carrera que se potencian con la aplicación del aprendizaje basado en proyectos. Este estudio tiene como objetivo identificar las características que deben poseer los grupos que realizan aprendizaje basado en proyectos para un desarrollo de software en equipo.

**Palabras claves:** Aprendizaje Basado en Proyectos, desarrollo de software, ingeniería de software.

## 1 Introduction

Project-Based Learning – PBL is a methodology where students actively engage, enhancing academic motivation. It is an educational strategy centered on developing a project that addresses a real need, forming the core activity for achieving learning objectives. Typically, this involves creating a final product that is either real or closely simulates reality. To accomplish this, students must acquire or refine skills designated by the instructor. This technique is commonly associated with group or teamwork, enabling students to reach their objectives (Pérez González et al., 2008) collaboratively.

This pedagogical approach supports intra- and interdisciplinary management, promoting collaborative efforts among teachers and educational institutions (Sotomayor et al., 2021). Its primary feature is providing students with an authentic learning context, actively involving them in teaching-learning. Here, students

make critical decisions to complete tasks, with research playing a vital role in their problem-solving approach (Abella García et al., 2020). It has been demonstrated that students engaged in PBL can effectively solve problems, become more engaged in their learning, and develop greater self-sufficiency. Additionally, PBL promotes autonomy and self-confidence, as well as increases motivation. This approach, which departs from traditional teaching methods, presents tasks as dynamic challenges within a standard, rather than as isolated assignments (Abella García et al., 2020). PBL, a valuable tool in various fields, finds relevance in professional domains centered around project generation. In the context of Software Engineering – SE, PBL offers unique benefits, fostering a deeper understanding of the subject matter and enhancing problem-solving skills (Abella García et al., 2020).

Software Engineering involves the application of a systematic, disciplined, and quantifiable approach to software development. This includes the creation of tools, methods, and techniques to support software production, as well as all aspects of project management involving people, processes, and technology tools (Pressman, 2010). In the educational realm, the overarching goal of a Software Engineering course is to equip students with the skills necessary to develop software projects from inception to maintenance. Therefore, it is essential for students to gain both the knowledge and abilities required to undertake a complete software project (Sánchez & Blanco, 2012). In the context of PBL, students, as active participants, must navigate challenges such as dealing with incomplete and imprecise information, self-regulating, and committing to their work. This active involvement in the learning process is crucial as they must define and direct their own learning pathways (Villalobos-Abarca et al., 2018).

PBL is particularly significant in software development as it represents a systematic teaching approach where students acquire essential knowledge and life skills through a comprehensive and structured inquiry process. This method engages students with authentic and complex questions, leading to the creation of high-quality products (Villalobos-Abarca et al., 2018). Through PBL, students are encouraged to take greater responsibility for their knowledge development.

Considering these aspects, this research compares two groups from the Software Engineering program engaged in project-based learning. Both groups work on the same project, allowing for an evaluation based on various variables, including academic performance, specific competencies in the field, teamwork, and technical and methodological knowledge. This will allow to identify characteristics of students who may be more suited to working with PBL.

The structure of this document is as follows: section 2 presents the theoretical framework, detailing concepts related to software engineering, its competencies, and Project-Based Learning – PBL. This is followed by an explanation of the research methodology, where details of the experiment are provided. Next, the results are presented and discussed. The document concludes with the conclusions and references of the study.

## 2 Background

The most crucial aspects of this research are outlined below.

Software engineering is a discipline that applies engineering principles to the efficient design, development, testing, and maintenance of software systems. It encompasses using systematic methods and specialized tools to ensure software products' quality and timely delivery while adapting to user requirements (Sommerville, 2015).

A software development methodology is a structured set of practices, processes, procedures, and rules that guide the development and management of software projects throughout their life cycle. These methodologies provide a framework for planning, structuring, and controlling the development process to improve the efficiency and quality of the resulting software (Pressman, 2010). There are various methodologies, each showcasing different phases and roles (Ali Munassar & Govardhan, 2010). However, the classical methodology is known as the waterfall methodology, one of the earliest to emerge, with phases common to other approaches. Its phases are (Ali Munassar & Govardhan, 2010):

1. System requirements – analysis: Establishes the components needed to build the system, including hardware requirements, software tools, and other necessary components.
2. Software requirements – analysis: Sets the expectations for software functionality and identifies which system requirements the software will affect. Requirements analysis includes determining the necessary interaction with other applications and databases, performance requirements, user interface requirements, etc.

3. Architectural design: Determines the software framework of a system to meet specific requirements. This design defines the major components and their interaction but not the structure of each component. Designers may determine the interfaces and external tools used in the project.
4. Detailed design: Checks the software components defined in the architectural design stage and specifies how each component is implemented.
5. Coding: Implements the detailed design specification using a programming language.
6. Testing: Verifies if the software meets the specified requirements and identifies any bugs in the code.
7. Maintenance: Addresses problems and enhancement requests after the software's release.

Software engineering combines technical aspects of computer science with soft skills such as communication, negotiation, collaboration, and teamwork. In this way, the software engineer can generate software solutions that meet the needs of stakeholders and increase the efficiency of impacted business processes (Gómez Álvarez et al., 2015). These soft skills have been incorporated into the software engineering teaching process (Gómez Álvarez et al., 2015) through case studies for simulating natural software development environments and executing university-industry software projects.

## 2.1 PBL and software engineering

The essential skill students must acquire in the digital era is learning how to learn. As a result, learning has shifted from being an individual construction of knowledge to becoming a social process. However, even for the most experienced teachers, keeping students engaged and motivated in educational institutions is a considerable challenge (Marti et al., 2010). In this context, several didactic strategies can be applied, considering the characteristics of the students. According to de Miguel Díaz (2006), didactic methods can be classified into: **1)** an individualization didactic approach, with methods such as programmed teaching, self-directed learning, and academic tutoring, among others; **2)** a socialization didactic approach, with models such as the case method, seminar, and peer tutoring; and **3)** the globalized approach, with methods such as Project-Based Learning – PBL and problem-solving. The project-based learning method **3)** is the subject of analysis in this research.

Unlike traditional lectures, PBL is believed to foster deeper learning, where students retain less of what they are taught. Assessment in project-based learning is also influenced by its proximity to real-world situations. Since the artifacts constructed during the project are representations of learning, it is vital to provide authentic and constructive feedback toward the goals of the problem (da Cunha, 2005).

PBL has been widely adopted in educational engineering. It is defined as an instructional method of constructivist teaching that uses real problems as a motivating element for learning. Thus, some studies in computer science education have shown that PBL facilitates students' engagement and learning ability, contributing to developing skills such as teamwork, holistic vision, critical thinking, and problem-solving. Previous research has shown that PBL is particularly well suited for teaching software engineering. This approach allows students to apply knowledge from the beginning of the course, contributing to deep learning.

Moreover, seeing a software solution that does not deviate from industrial practice and uses state-of-the-art technologies represents a significant additional incentive (Adorjan & Solari, 2021).

## 2.2 Software engineer competencies

A competency refers to a combination of knowledge, skills, abilities, and attitudes that integrate into an individual's capabilities, traits, motivations, values, and personal experiences (Cerato & Gallino, 2013).

Competencies can be categorized as:

1. Cross-functional or generic competencies: These are essential for individuals to be productive when they enter the workforce and are not explicitly related to the professional area.
2. Specific competencies: These are competencies for a professional area. In the case of this research, they are the competencies a software engineer should have.

The cross-functional or generic competencies considered in this research are:

1. Oral and written communication: The ability to transmit knowledge, express ideas, and arguments clearly, rigorously, and convincingly, both orally and in writing, using graphic resources and necessary means appropriately, adapting to the characteristics of the situation and the audience.
2. Analysis and synthesis of information: The ability to recognize and describe a reality's constitutive elements and organize significant information according to pre-established criteria suitable for a purpose.
3. Problem formulation and solving: The skill of analyzing the elements of a problem to devise strategies that allow for a reasoned solution that is contrasted and in line with certain pre-established criteria.
4. Solution modeling: The capability to analyze the fundamentals and properties of existing models and to translate and interpret model elements in terms of the real world.
5. Autonomous learning: The initiative and lifelong interest in learning.
6. Teamwork: Effective participation in diverse teams and active collaboration in achieving common goals.
7. Decision making: The ability to identify patterns anticipating possible explanations and solutions to industrial, technological, and operational problems for appropriate decision-making.
8. Effective use of ICT tools: The capacity to keep updated with the use of technology in their area, impacting continuous improvement.
9. Responsibility in performance: Understanding the professional, ethical, legal, security, and social aspects and the responsibility inherent in each.
10. Insight into the Impact of Solutions: The skill to analyze IT solutions' local and global impact on individuals, organizations, and society.

The specific competencies of a software engineer, as defined by ANIEI (2024) and CONAIC (2023), are expected to be acquired during a degree in software engineering.

Specific competencies of a software engineer:

11. Conducts software requirements engineering: Recognizes the context, needs, and stakeholders in a system, employing techniques to identify, obtain, analyze, prioritize, document, verify, and validate requirements within the context of software development life cycles and processes.
12. Designs software: Designs the behavior, architecture, and interface of software solutions based on requirements, utilizing strategies, methods, techniques, and modeling languages specific to software design.
13. Builds software: Develops software for different types of applications, using programming methodologies and paradigms within the context of software development life cycles and processes, ensuring the required quality attributes.
14. Performs software testing: Plans, allocates, and executes various types, techniques, processes, and controls within testing scenarios, adhering to the required quality attributes.
15. Conducts software maintenance: Applies types, processes, and maintenance techniques following the required quality attributes.
16. Manages software projects: Utilizes methods, strategies, processes, tools, and techniques for effective software project management.
17. Estimates software project parameters: This section applies metrics for software estimation (size, cost, effort, personnel, time, productivity, quality, and documentation) in accordance with system life cycle models.
18. Ensures software quality: Uses techniques, tools, and strategies to plan, assure, and control the quality of a software product.
19. Establishes security mechanisms: Creates or proposes methods and strategies to evaluate security and select criteria to prevent vulnerabilities in software security.
20. Employs life cycles: Utilizes elements and criteria in applying life cycle models according to the context of software development processes.
21. Verifies quality of software solutions: Uses various testing models to ensure the quality of the software product.
22. Uses tools for software creation: Utilizes industrial methods and CASE tools for different phases in the software process.

The previously mentioned **22** competencies are considered for evaluating students' performance as they progress through the software engineering program.

# 3 Methodology

The methodology for this research is adapted from Hernández Sampieri's proposal (2014), which involves using experimental and control groups to measure the effects of the treatment. However, due to the specific nature of the topic, some adjustments were made, as detailed below.

## 3.1 Description of the experiment

In general terms, the experiment involves comparing the performance of two groups using Project-Based Learning in a group format. To provide a point of comparison, both groups develop the same project. The experiment is conducted over a semester in the software engineering program on web application development. Further details are provided below.

## 3.2 Study population

The study population comprises software engineering students from the Faculty of Engineering Mochis at the Autonomous University of Sinaloa. These students are in their sixth semester, generally aged between **20** and **22** years, and are predominantly male. The students in this program have a strong technological background, with skills in computer systems, programming languages, and databases; some also bring prior web development experience.

## 3.3 Sampling

The sample type is non-probabilistic for convenience. This approach allows for selecting accessible cases that agree to be included. It is based on the subjects' convenient accessibility and proximity to the researcher. However, it should be noted that the subjects must meet specific characteristics related to the experiment's objectives (Hernández-Sampieri & Mendoza Torres, 2018). These characteristics are defined in the previously mentioned study population block.

Considering the above, a quasi-experiment was conducted as the groups were pre-assigned. The formation of these groups followed the academic organization of the faculty. The groups were formed at the beginning of the semester. Some students do not advance to the next semester due to poor performance, and others join by changing groups for personal reasons or due to low performance; however, the group's core remains as initially formed.

## 3.4 Procedure

The experiment's procedure is explained as follows:

1. A software development project was created, and a list of requirements was generated. The project, titled "Automation of software estimation with planning poker," required developing a web application and generating software engineering documentation.
2. The theory of roles in software projects was explained so that students could understand the roles and select two: a primary and a secondary role. The purpose of the secondary role was to distribute roles among students and avoid an excess of students in the same role. This selection was made to form teams based on roles. The roles were analyst, designer, programmer, tester, documenter, and project leader.
3. The entire project was developed during the "Web application development" class. This class laid the foundation for explaining topics like HTML, CSS, JavaScript, and a server-side language – PHP. Therefore, all projects were expected to apply these technologies, although it was optional; if any group wanted to use other technologies, they were free to do so.

4. The professor acted as the project's client, the person requesting the project. Initially, each group had to set dates for interviews with the client and start generating the software requirements. In the end, the requirements generated by the group were compared with the original ones created by the professors. This served as a comparison point to measure the analysts' understanding of the problem. However, after this stage, the original project requirements were given to ensure everything was clear.
5. The workgroups had to develop various software engineering artifacts such as requirement generation documentation, use case diagrams, use cases, class diagrams, entity-relationship diagrams, and interface design, among others.
6. As the classes progressed, groups were asked to improve their development. Each group had to explain its progress according to its role. The project leaders acted as group coordinators, and communication and demand to the teams were made through them.
7. At the end of the semester, the group had to present their finished project, at least to an advanced level. The project began with the semester.

### 3.5 Project description

The "Automation of software estimation with planning poker" project is a user story estimation system for Scrum (Martel, 2014). The system must estimate the complexity of user stories using virtual planning cards. Additionally, the system should include project management, user stories, members, and a history of estimations.

### 3.6 Statistical analysis

At the end of the project delivery period, a survey was conducted to collect data and analyze relationships between variables. The statistical tool SPSS was used for data processing and analysis. An intrasubject analysis was applied, meaning the same experiment was conducted on all students, and internal aspects of the sample were analyzed to explore whether the obtained results were related among variables.

Two statistical tests were applied to the data analysis to find relationships: the student's ***t***-test and the Mann-Whitney ***U*** test. Both are used to check differences between the means of their results. However, the first is for numerical values, and the second is for ordinal categorical values (Flores-Ruiz et al., 2017; Hernández Sampieri et al., 2014).

### 3.7 Applied survey

The survey initially considered basic aspects of the students, such as names, surnames, age, gender, average grade at the time of the survey, group, and whether they are regular students. It included the field of grade in the project.

The second part included aspects related to the acceptance of the work strategy:

- I have participated in another similar group project this semester or previously.
- The group project seemed very useful for collaboration among group members.
- The group project seemed very useful in determining the function of each role in a software project.
- I would like to develop another similar project in other subjects.
- I found the project's theme very complex.
- The project was too complex for our technical knowledge (use of programming languages, database managers, code, document repositories, etc.).
- The project was too complex for our methodological knowledge (software development methodologies, role activities, documentation development, etc.).
- The project was too complex for the group's organizational capacity (organization of work activities, communication between groups, attendance at meetings).

- Do you think the start and end time of the project was appropriate?
- Do you think another role was needed for the project to develop better? [Yes (which), No]

The last section focuses on the competencies of the software engineer.

These competencies are described in the theoretical framework section and consist of **22** generic and specific competencies. Therefore, students were asked to rate their level of proficiency in these skills.

## 4 Results

According to the results described in Table 1, there are **39** participants in both groups. The groups maintain nearly the same proportion of women and men and similar age ranges. Consequently, the averages are very similar.

**Table 1.** Descriptive statistics of ages by group

| | Group | | |
|---|---|---|---|
| | 1 | 2 | General |
| **Total** | 20 | 19 | 39 |
| **Women** | 6 | 5 | 11 |
| **Men** | 14 | 14 | 28 |
| **Minimum** | 20 | 20 | 20 |
| **Maximum** | 22 | 24 | 24 |
| **Mean** | 20.6 | 21.2 | 20.9 |
| **Std. Dev.** | 0.598 | 1.357 | 1.071 |

The reliability of the survey was measured using Cronbach's *Alpha*, obtaining a score of **0.811**. Although this value is imperfect, it is considered a good evaluation, falling within the **0.8**-**0.9** range.

Figure 1 and Figure 2 shows a higher number of irregular students in group **2** compared to group **1**. An irregular student has failed one or more subjects. The figures show that group **2** has almost **50%** irregular students.

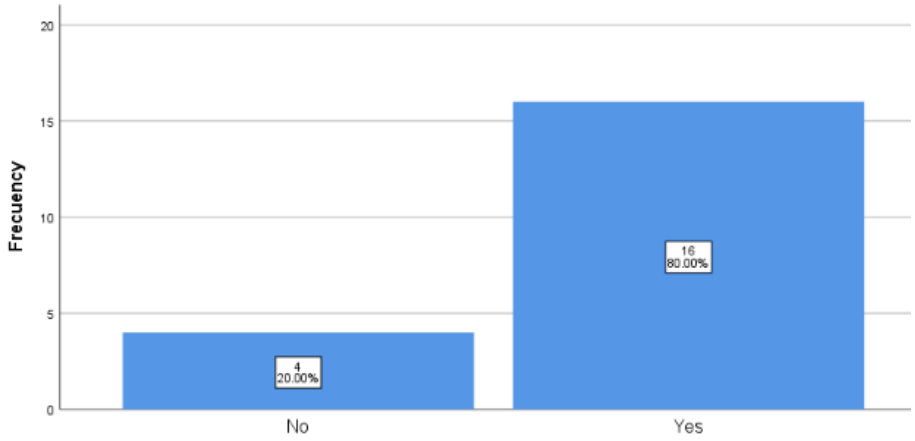


**Figure 1.** Irregular (no, 20%) and Regular (yes, 80%) Students, group 1

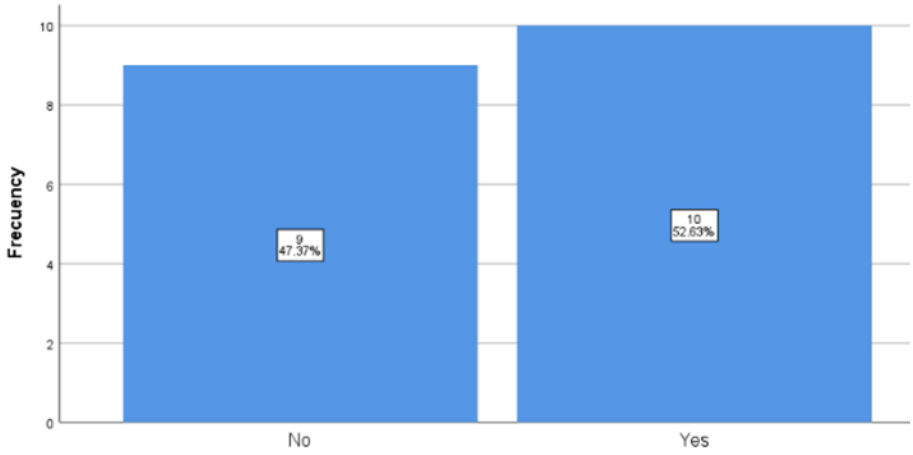


**Figure 2.** Irregular (No, 47.37%) and Regular (Yes, 53.63%) Students, group 2

Table 2 and Table 3 show various aspects of project acceptance by the group, as well as the evaluation of the technical, methodological, and organizational complexity of the didactic strategy. The aspects are:

A. The group project was very useful for collaboration among group members.
B. The group project was very useful to understand the function of each role in a software project.
C. I would like to develop another similar project in other subjects.
D. The theme of the project seemed very complex.
E. The project was too complex for our technical knowledge (use of programming languages, database managers, code, and document repositories, etc.)
F. The project was too complex for our methodological knowledge (software development methodologies, role activities, documentation development, etc.)
G. The project was too complex for the group's organizational capacity (organization of work activities, communication between groups, attendance at meetings)

The tables are divided into columns where each statement is evaluated on a **5**-point Likert scale. The final column (Result **1-5**) shows an average, considering the Likert frequency, which allows for a general comparison of the group results. The results indicate that group **1** better accepts the group project in almost all aspects.

**Table 2.** Acceptance and complexity of the group project (group **1**)

| Aspect | Totally disagree | Disagree | Neutral | Agree | Totally agree | Result (1-5) |
|---|---|---|---|---|---|---|
| **A** | 0 | 1 | 5 | 7 | 7 | **4.00** |
| **B** | 1 | 1 | 3 | 5 | 10 | **4.10** |
| **C** | 3 | 3 | 7 | 2 | 5 | **3.15** |
| **D** | 1 | 5 | 5 | 7 | 2 | **3.20** |
| **E** | 2 | 6 | 6 | 5 | 1 | **2.85** |
| **F** | 3 | 6 | 8 | 2 | 1 | **2.60** |
| **G** | 2 | 7 | 3 | 8 | 0 | **2.85** |

**Table 3.** Acceptance and complexity of the group project (group **2**)

| Aspect | Totally disagree | Disagree | Neutral | Agree | Totally agree | Result (1-5) |
|---|---|---|---|---|---|---|
| **A** | 3 | 4 | 8 | 3 | 1 | **2.74** |
| **B** | 1 | 1 | 5 | 8 | 4 | **3.68** |
| **C** | 8 | 3 | 4 | 3 | 1 | **2.26** |
| **D** | 0 | 0 | 5 | 8 | 6 | **4.05** |
| **E** | 1 | 6 | 5 | 4 | 3 | **3.11** |
| **F** | 1 | 3 | 3 | 6 | 6 | **3.68** |
| **G** | 0 | 1 | 6 | 1 | 11 | **4.16** |

Overall, the groups' opinions on the issues in Table 2 and Table 3 are significantly contrasting. Group **1** showed high acceptance of the didactic strategy, while group **2** perceived much complexity in the various

attributes evaluated. Therefore, a Mann-Whitney *U* test for differences in means with ordinal categorical data was applied to determine a statistically significant difference, considering a **95%** confidence level. The results are shown in Table 4.

**Table 4.** Mean difference between categorical data (group **1** and **2**)

| Item | Asymptotic significance (two-tailed) |
|---|---|
| The group project was very useful for collaboration among group members | .001 |
| I would like to develop another similar project in other subjects | .053 |
| The theme of the project seemed very complex | .015 |
| The project was too complex for our technical knowledge | .551 |
| The project was too complex for our methodological knowledge | .007 |
| The project was too complex for the group's organizational capacity | .001 |
| The group project was very useful to understand the function of each role in a software project | .137 |

Figure 3 shows a boxplot representing the distributions of the final grades of the group project. The grades of group **1** have a median of **9** and an average of **8.65**, while group **2** has a distribution with a median of **6** and an average of **6.21**. The data shows a higher average for group **1** than for group 2.

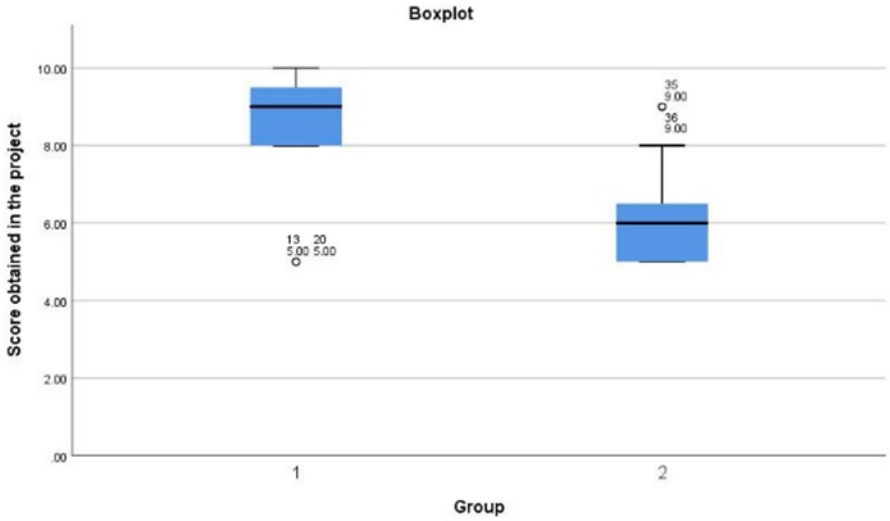


**Figure 3.** Grades obtained in the project by group

Figure 3 provides a visual representation, such as a bar chart or line graph, illustrating the grades obtained by each group in the project. The chart displays the distribution of grades for group **1** and group **2**, highlighting the differences in performance between the two groups.

Figure 4 shows the evaluation results of the **22** software engineer competencies considered in the project. The competencies are represented on the ***x***-axis, while the evaluation value is on the ***y***-axis. The ***y***-axis value corresponds to the Likert scale, but an average was obtained (considering the frequency and Likert value) to generalize the question for the group members. Therefore, the closer the competency evaluation is to the value **5** (totally agree), the better. The results indicate that group **1** (represented by the blue line) performs better in most evaluated competencies than group **2** (represented by the red line). Competencies **1** to **10** are considered generic, while **11** to **22** are specific.

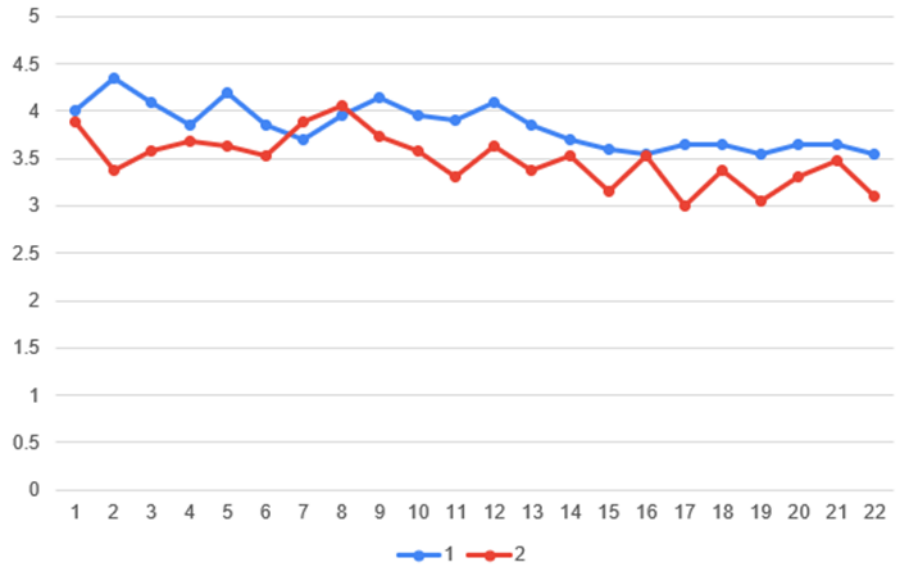


**Figure 4.** General comparison of competencies between group **1** and **2**

Figure 4 likely compares the competencies of group **1** and group **2** in a visual format, like a line graph or bar chart, with the competencies on the ***x***-axis and their evaluation scores on the ***y***-axis. The blue line represents group **1**, and the red line represents group **2**, showing the differences in competencies between the two groups.

Comparisons between variables were made to establish whether there was a significant difference between the performances of both groups. A Student's *t*-test was applied. Group 1 had an average project grade of **8.65**, and group **2** had an average of **6.21**. The alternative hypothesis states a significant difference exists between the averages of groups **1** and **2**. With **95%** confidence, if the ***p***-value is less than **0.05**, the alternative hypothesis is accepted; otherwise, it is rejected. The results indicated a ***p***-value of **0.000**; therefore, the alternative hypothesis of a significant difference between groups **1** and **2** averages is accepted.

# 5 Discussion

## 5.1 Rate of irregular students

The key point is that the lower performance average of group **2** is due to the irregularity of the students in the group. Nearly **50%** of the students in group **2** have failed subjects, which implies that their low performance affects their studies overall. Failing a subject puts a student in an irregular position and impacts their behavior and communication with classmates, family, and friends. The anxiety or distress they experience can negatively affect their long-term health if not adequately addressed (Martínez Rico & Luna Reséndiz, 2006). Therefore, this group project development strategy does not favor groups with a high failure rate. PBL requires student autonomy in choosing methods, rhythms, and work styles, and if the group does not show sufficient cognitive development in learning skills, PBL can be counterproductive.

## 5.2 Acceptance and complexity of the group project

Group **2** showed little willingness to work as a team, which affected the results. The test showed a ***p***-value of **0.001**, indicating a significant difference in the collaboration applied by the group members (see Table 4). In the current work environment, teamwork has surpassed individual work (Asún Dieste et al., 2019). Thus, instilling collaboration skills is very important in university education, as individuals who cannot collaborate effectively in work activities are easily replaceable by others who can. For a didactic strategy

like PBL, where the strength is based on collaborative work and dividing efforts among team members, poor coordination can lead to poor results in task achievement.

### 5.3 I found the subject matter of the assignment to be very complex.

Group **2** considered that the subject matter of the project was very complex. However, the planning poker estimation method was indeed unknown; the method is quite simple, basing the complexity of the estimates on letters with ascending numbering according to the complexity of the activity (Mahnič & Hovelja, 2012). Thus, the project's complexity is a justifying factor for the group's results. The difference between the opinions of the two groups is significant because a ***p***-value equal to **0.015** was obtained. After all, group **1** does not consider the subject complex (see Table 4).

### 5.4 The project was too complex for our methodological knowledge.

The methodological aspect of the project is considered complex for group **2**, while the group demonstrates better management in this aspect (see Table 2 and Table 3). The difference between both averages (groups **1** and **2**) is significant, with a ***p***-value of **0.007** (see Table 4). The use of methodologies is essential in developing quality software; this represents coherent methods and is related by common principles, which aim to create or update software according to customer requirements (Rivas et al., 2015). Therefore, if this skill is lacking, software construction becomes complicated since there are no guides or guidelines for development. Although the subject of web application development does not consider in depth the use of any particular methodology, previous knowledge of other subjects serves as a basis for new projects. Developing software in an empirical and artisanal way leads to projects that do not meet customer requirements, deliveries out of time, exceeded budgets and conflicts among team members (Zumba Gamboa & León Arreaga, 2018).

### 5.5 The project was too complex for the group's organizational capacity.

In addition to the previous point, the forms of group organization strongly influence academic performance; however, group **2** needs to indicate having an adequate group organizational capacity, which led to poor performance (see Table 2 and Table 3). Group **2** considers its organizational capacity to be below average. In contrast to group **1**, the difference in averages in their opinions is significant, with a ***p***-value equal to **0.001** (see Table 4). Teamwork represents an aspect of vital importance in the development of projects since the goal is divided into various phases with assigned responsibilities; therefore, lacking this competence indicates a disadvantage in current times; it should be noted that the university represents the preparation and training of students for the work environment, so collaboration among team members should be a priority.

### 5.6 Other aspects

The aforementioned aspects are those for which there was a significant divergence in the group members' responses. Conversely, other aspects, though less frequently considered, were deemed equally important.

For example, both groups considered having the regular technical knowledge to develop the project. However, one group performed better, so the variable technical knowledge is less necessary than the others. Also, both groups consider that the didactic strategy helps determine the function of each role in the project. Finally, group **2** indicates that the work dynamics were not to their liking, i.e., PBL in group format needed to be better received, while group **2** did not show a total liking. However, they do perceive the work scheme as positive. Project-Based Learning for software development is an educational approach that focuses on students acquiring practical skills and knowledge by applying theoretical concepts in the realization of real software development projects. Rather than simply receiving theoretical information from an instructor,

students work in teams or individually to address concrete challenges and create functional solutions (Villalobos-Abarca et al., 2018).

### 5.7 Software engineer competencies

Considering that group **1** performed better than group **2**, the further apart the competency evaluations of both groups are (see Figure 4), the more significant the competencies will be in explaining the behavior of the groups. Applying this criterion, the competencies with the most significant difference between groups are:

1. Analysis and synthesis of information: Recognize and describe a reality's constituent elements and organize significant information according to pre-established criteria suitable for a purpose (competence **2**).
2. Estimate software project parameters: Apply metrics for software estimation (size, cost, effort, personnel, time, productivity, quality, and documentation) according to systems life cycle models (competence **17**).
3. Perform software requirements engineering: Recognize the context, needs, and stakeholders in a system using techniques to identify, elicit, analyze, prioritize, document, verify, and validate requirements in the context of software development life cycles and processes (competence **11**).
4. Autonomous learning: Learning by initiative and self-interest throughout life (competence **5**).

The competencies analysis and synthesis of information and software requirements engineering allow understanding the problem to be solved through a computer system; not mastering these competencies implies starting a project without understanding it, causing problems in its development. As indicated by Arias Chaves (2005), requirements engineering implies the creation of precise specifications that detail, in a clear, unambiguous, coherent, and concise manner, the demands of users or customers. The objective is to minimize the problems associated with poor requirements management during system development. The results obtained in Goñi *et al*. (2014) indicate that one of the positive aspects of PBL is to foster analytical skills. This is consistent with the results of the present research.

Also, estimating software project parameters refers to the effort to develop the product, including costs; however, for educational purposes, the economic resource is not a problem since the purpose is didactic. Nevertheless, the delivery times are essential since they are required to meet the estimated grades in estimated times; this constraint plays with the students as if it were a deadline for an actual project. Therefore, in case of not finishing the project in the stipulated time, the group will have a low grade, as happened with group **2**, where some roles did their work, others did not do their activity, and the rest left the assignment half done. Due to the excellent organization of time and work, group **1** performed better than group **2**, affirming that competence is essential for completing projects.

Autonomous learning is an essential attribute in all areas of knowledge, but it is essential in careers related to information and communication technologies. Autonomous learning represents a process where the student self-regulates his learning pace; the ability to self-regulate favors the learning process by taking control and guiding his own mental processes; in this learning process, the student is expected to be independent and self-manage his practice (Crispín et al., 2011). Software engineering is a branch in constant evolution, many times the industry surpasses the academy in the use of new technologies and methods, so the student must apply his research and autonomous learning skills to be at the forefront of new developments, applying these strategies even after graduating from the university. Therefore, having developed the ability to learn by oneself implies a high probability of overcoming the barriers of ignorance and achieving incomplete, defective, or unaddressed knowledge in class. These results contrast those presented in Goñi *et al*. (2014) where the self-inquiry item is considered one of the most outstanding after applying PBL in a subject in software engineering.

On the other hand, there were a couple of competencies where group **2** indicated better performance than group **1**:

1. Decision making: Identify patterns that anticipate possible explanations and/or solutions to industrial, technological, and operational problems for proper decision-making (competence **7**).
2. Effective use of ICT tools: (including new technologies) Ability to update the use of technology in an area that impacts its continuous improvement (competence **8**).

According to the results, these two attributes were superior to those achieved by group **1**, but they are also the best-performing attributes of group **2**. Group **2** indicates having a correct choice of options, evaluating alternatives, and considering important factors before deciding. In addition, at least the information and communication technology tools known by group **2** are mastered; however, the lack of autonomous learning means this mastery is insufficient for projects involving new tools.

### 5.8 Hypothetical test

Finally, the established hypothesis test indicates a significant difference between the average performance of the groups; considering a better average of group **1** versus group **2**, we can indicate that the characteristics presented by group **1** benefit the didactic strategy employed, i.e., group PBL for software development.

## 6 Conclusion

This research compares performance and academic competencies with software engineering groups using PBL with a group format, i.e., students develop the same project in teams considering the roles of software engineering.

The research obtained the following findings regarding the competencies evaluated:

1. PBL is not recommended when the group has too many irregular students who have yet to show commitment and effort in previous subjects.
2. The analysis phase in software development involves requirements engineering and information synthesis; therefore, to use PBL in a group, students must develop these competencies; otherwise, a good requirements assessment will not be carried out, and the development will not be successful.
3. Autonomous learning is vital in software engineering since it implies investigating by oneself the doubts or lack of knowledge that one may have about a subject. A career in invariable evolution, such as computing, requires constant updating.
4. The organization of work and time are basic aspects of working with PBL; in this way, the projects can be completed in the stipulated time.

As future work, the recommendations found in this research can be applied with other groups to reinforce the results.

## ORCID ID

Alan David Ramírez Noriega https://orcid.org/0000-0002-8634-9988
Samantha Paulina Jiménez Calleros https://orcid.org/0000-0003-0938-7291
Herman Geovany Ayala Zuñiga https://orcid.org/0009-0005-6106-4563
Yobani Martínez Ramírez https://orcid.org/0000-0002-4967-9187

## References

Abella García, V., Ausín Villaverde, V., Delgado Benito, V., & Casado Muñoz, R. (2020). Aprendizaje Basado en Proyectos y estrategias de evaluación formativas: Percepción de los estudiantes universitarios. *Revista Iberoamericana de Evaluación Educativa, 13*(1), 93-110. https://doi.org/10.15366/riee2020.13.1.004

Adorjan, A., & Solari, M. (2021). Software engineering Project-Based Learning in an Up-To-Date technological context. *Proceedings of the IEEE URUCON 2021*, (pp. 486-491). Montevideo, Uruguay. https://doi.org/10.1109/URUCON53396.2021.9647348

Ali Munassar, N. M., & Govardhan, A. (2010, September). A comparison between five models of software engineering. *IJCSI International Journal of Computer Science Issues, 7*(5), 94-101. Retrieved from https://www.ijcsi.org/papers/7-5-94-101.pdf

ANIEI. (2024). *Entidad de Certificación Acreditada (ANIEI)*. Retrieved from ANIEI website: https://www.aniei.org.mx/aniei-entidad-de-certificacion-y-evaluacion-del-conocer/

Arias Chaves, M. (2005). La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. *InterSedes: Revista de las Sedes Regionales, VI*(10), 1-13. Retrieved from https://www.redalyc.org/pdf/666/66612870011.pdf

Asún Dieste, S., Rapún López, M., & Romero Martín, M. R. (2019). Percepciones de estudiantes universitarios sobre una evaluación formativa en el trabajo en equipo. *Revista Iberoamericana de Evaluación Educativa, 12*(1: Evaluación formativa y compartida en Educación), 175 - 192. https://doi.org/10.15366/riee2019.12.1.010

Cerato, A. I., & Gallino, M. (2013). Competencias genéricas en carreras de ingeniería. *Ciencia y Tecnología, 13*, 83-94. https://doi.org/10.18682/cyt.v1i13.58

CONAIC. (2023, February). *¿Quiénes somos?* Retrieved from CONAIC website: https://www.conaic.net/

Crispín, M. L., Caudillo, L., Doria, C., & Esquivel Peña, M. (2011). Aprendizaje Autónomo. In M. Doria Serrano, A. Beatriz Rivera, M. T. De la Garza Camino, S. Carrillo Moreno, L. Guerrero Guadarrama, H. Patiño Domínguez, . . . M. J. Athié Martínez, & M. L. Crispín Bernardo (Ed.), *Aprendizaje autónomo: Orientaciones para la docencia* (First ed., pp. 49 - 65). México D. F.: Universidad Iberoamericana. Retrieved from https://biblioteca.clacso.edu.ar/Mexico/dcsyp-uia/20170517031227/pdf_671.pdf

da Cunha, P. R. (2005). Teaching software engineering using Project-Based Learning. *iCEER-2005, Exploring Innovation in Education and Research, 1-5 March 2005*, (pp. 1-4). Tainan, Taiwan. Retrieved from https://www.researchgate.net/publication/215575463

de Miguel Díaz, F. M. (Ed.). (2006). *Metodologías de enseñanzas y aprendizaje para el desarrollo de competencias: Orientaciones para el profesorado universitario ante el espacio europeo de educación superior.* España: Alianza. Retrieved from https://dialnet.unirioja.es/servlet/libro?codigo=293088

Flores-Ruiz, E., Miranda-Novales, M. G., & Villasís-Keever, M. Á. (2017). El protocolo de investigación VI: cómo elegir la prueba estadística adecuada. Estadística inferencial. *Revista Alergia México, 64*(3), 364 - 370. https://doi.org/10.29262/ram.v64i3.304

Gómez Álvarez, M. C., Manrique Losada, B., & Gasca Hurtado, G. P. (2015). Propuesta de evaluación de habilidades blandas en ingeniería de software por medio de proyectos Universidad-Empresa. *Revista Digital Educación en Ingeniería, 10*(19), 131 - 140. https://doi.org/10.26507/rei.v10n19.549

Goñi, A., Ibáñez, J., Iturrioz, J., & Vadillo, J. Á. (2014). Aprendizaje Basado en Proyectos usando metodologías ágiles para una asignatura básica de Ingeniería del Software. In M. R. Albizu, M. Á. Díaz Fondón, , & B. López Pérez (Ed.), *Actas de las XX Jornadas sobre la Enseñanza Universitaria de la Informática: JENUI 2014, del 9 al 11 de julio de 2014*, (pp. 133-140). Oviedo, Spain. Retrieved from https://dialnet.unirioja.es/servlet/articulo?codigo=7368134

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). *Metodología de la investigación* (6th ed.). Spain: McGraw Hill España. Retrieved from https://dialnet.unirioja.es/servlet/libro?codigo=775008

Hernández-Sampieri, R., & Mendoza Torres, C. (2018). *Metodología de la investigación: Las rutas cuantitativa, cualitativa y mixta.* México: Mc Graw Hill Education. Retrieved from http://www.biblioteca.cij.gob.mx/archivos/materiales_de_consulta/drogas_de_abuso/articulos/sampierilasrutas.pdf

Mahnič, V., & Hovelja, T. (2012, September). On using planning poker for estimating user stories. *Journal of Systems and Software, 85*(9), 2086 - 2095. https://doi.org/10.1016/j.jss.2012.04.005

Martel, A. (2014). *Gestión práctica de proyectos con Scrum: Desarrollo de software ágil para el Scrum Master* (Third ed., Vol. 1). C. I. P. Platform.

Marti, J. A., Heydrich, M., Rojas, M., & Hernández, A. (2010, Mayo). Aprendizaje basado en proyectos: una experiencia de innovación docente. *Revista Universidad EAFIT #DescubreyCrea, 46*(158), 11–21. Retrieved from https://publicaciones.eafit.edu.co/index.php/revista-universidad-eafit/article/view/743

Martínez Rico, I. M., & Luna Reséndiz, R. (2006). Qué representa para el alumno de la ENMH ser irregular, cómo lo enfrenta y cuál ha sido su participación en el Programa Institucional de Tutorías. *2do Foro de Investigación Educativa*, (pp. 1-6). Retrieved from https://www.repositoriodigital.ipn.mx/handle/123456789/3111

Pérez González, A., Serrano Mira, J., Peñarrocha Alós, I., & Pérez Soler, E. (2008). Un sistema para la evaluación del aprendizaje basado en proyectos. *XVI Congreso Universitario de Innovación Educativa en las Enseñanzas Técnicas*, (pp. 1-12). Cádiz, España. Retrieved from https://www.researchgate.net/publication/224982193_Un_sistema_para_la_evaluacion_del_aprendizaje_basado_en_proyectos

Pressman, R. S. (2010). *Software engineering: A practitioner´s approach* (7th ed.). New York, USA: McGraw-Hill.

Rivas, C. I., Corona, V. P., Gutiérrez, J. F., & Hernández , L. (2015, December). Metodologías actuales de desarrollo de software. *Revista Tecnología e Innovación, 2*(5), 980 - 986. Retrieved from https://www.ecorfan.org/bolivia/researchjournals/Tecnologia_e_innovacion/vol2num5/Tecnologia_e_Innovacion_Vol2_Num5_6.pdf

Sánchez, P., & Blanco, C. (2012). Implantación de una metodología de aprendizaje basada en proyectos para una asignatura de Ingeniería del Software. In L. Jiménez Linares, M. Genero Bocco, & M. Á. Redondo Duque (Ed.), *JENUI 2012, Actas de las XVIII Jornadas sobre la Enseñanza Universitaria de la Informática 10-13 de julio 2012.* (pp. 41-48). Ciudad Real: Universidad de Castilla-La Mancha. Retrieved from https://rua.ua.es/entities/publication/fd6c2df3-6fc7-4dfb-be34-18697a241eae

Sommerville, I. (2015). *Software engineering* (10th ed.). Pearson Education. Retrieved from https://books.google.com.co/books?id=x_qZCgAAQBAJ

Sotomayor, C., Vaccaro, C., & Téllez, A. (2021, Septiembre). *Aprendizaje Basado en Proyecto: Un enfoque pedagógico para potenciar los procesos de aprendizaje hoy.* Retrieved from FCH Website: https://fch.cl/wp-content/uploads/2021/10/ABP-un-enfoque-pedagogico-para-potenciar-aprendizajes.pdf

Villalobos-Abarca, M. A., Herrera-Acuña, R. A., Ramírez, I. G., & Cruz, X. C. (2018, June). Aprendizaje basado en proyectos reales aplicado a la formación del ingeniero de software. *Formación Universitaria, 11*(3), 97 - 112. https://doi.org/10.4067/S0718-50062018000300097

Zumba Gamboa, J. P., & León Arreaga, C. A. (2018). Evolución de las metodologías y modelos utilizados en el desarrollo de software. *Innova Research Journal, 3*(10), 20 - 33. Retrieved from https://www.redalyc.org/articulo.oa?id=737880712002