CLEI Electronic Journal, Volume 28, Issue 5, paper 5, September 2025

Crosscutting Teaching Strategies for Software Development Teams: Insights from Kolb's Inventory

Mauricio Hidalgo
Universidad San Sebastián, Facultad de Ingeniería,
Providencia, Chile, 7510157
mauricio.hidalgob@uss.cl

and

Kattia Rodriguez
Universidad Técnica Nacional, Área de Ciencias Básicas,
Villa Bonita, Costa Rica, 19024050
krodriguezb@utn.ac.cr

and

Laura M. Castro
Universidade da Coruña, Facultade de Informática,
A Coruña, Spain, 15001
lcastro@udc.es

and

Fernando Montoya
Universidad Técnica Federico Santa María, Doctorado en Ingeniería Aplicada,
Valparaíso, Chile, 2390123
fernando.montoya@usm.cl

and

Hernán Astudillo
Universidad Andrés Bello, Instituto de Tecnología para la Innovación en Salud y Bienestar,
Viña del Mar, Chile, 2531015
hernan.astudillo@unab.cl

Abstract

The rapid evolution of the IT industry has attracted many professionals without prior IT experience, creating opportunities and a critical need for effective training and continuous education to maintain a skilled and adaptable workforce. Despite this demand, only about one third of non-IT professionals transitioning into the field remain long-term, often due to insufficient training. This issue becomes particularly challenging when addressing crosscutting aspects of the discipline, such as programming, error detection, and requirements specification. Considering the above, this paper proposes teaching strategies tailored to the most representative learning style within the industry to improve professional development and retention in Software Engineering. To address this, Kolb's Learning Styles Test was administered to 112 software development professionals to identify the predominant learning style. The responses were analyzed to determine patterns and insights relevant to training. The analysis revealed that the Thinking Learning Style is the most representative among industry professionals. Based on this finding, we present customized teaching strategies to address key challenges in Software Engineering Training.

Keywords: Software Engineering Education, Crosscutting Teaching Strategies, Kolb's Learning Styles, Software Workforce Training

1 Introduction

The increasingly critical role of software in key systems presents new challenges for the education of software engineers, although software developers are currently still educated in traditional ways [1]. Moreover, software engineering is a young and promising discipline, still under development and improvement, which is evident in the evolving curriculum and teaching methodologies used in higher education [2]. Furthermore, the main dichotomy that we face in engineering is learning by studying (at school) versus learning by doing (at work) [3], and this issue is not a binary choice. In this way, the learning process for industry professionals is shaped by various factors, such as professional experience, educational background, and training strategies employed by companies. In addition, there is a significant mismatch between industry requirements and recent graduates' competencies [4, 5], prompting organizations to allocate resources for supplementary training for new employees. Despite these efforts, only one third of the employees without an IT background continue to work in the IT sector for the long term [6]. This highlights the need to implement knowledge management systems and foster learning organizations to improve the learning process within industrial environments [7].

Considering this context, our research aims to bridge the gap between educational practices and industry demands by exploring two key research questions:

- **R.Q.1:** What is the predominant learning style among professionals in the software industry?
- **R.Q.2:** What are the teaching strategies for addressing crosscutting aspects in software development training?

Rationale: Identifying the predominant learning style among software industry professionals helps define teaching strategies, providing a baseline for tailoring training approaches, considering that crosscutting aspects refer to competencies and knowledge areas that extend beyond technical skills, influencing professional practice and teamwork for any role of the Software Development Team. Separately, the term "crosscutting teaching strategies" refers to instructional approaches that can be applied across different content areas in software engineering education, regardless of whether they address technical, soft, or managerial topics.

To assist lecturers and provide a quick reference, this paper is structured as follows: Section 2 provides an overview of Kolb's Learning Styles Inventory (KLSI); Section 3 reviews related work on the application of KLSI in Software Engineering Education; Section 4 outlines our research methodology, including the execution process; Section 5 presents the survey findings, identifies the predominant learning styles, and proposes customized educational strategies; Section 6 details the outcomes of an external validation process; Section 7 examines the possible study's limitations and threats to validity; and Section 8 and Section 9 discuss implications for enhancing the education of software practitioners, conclude the study, and propose future research directions.

2 Kolb's Learning Styles Inventory - KLSI

Kolb [8] defined learning as "the process whereby knowledge is created through the transformation of experience: knowledge results from the combination of grasping and transforming experience," and introduced the Experiential Learning Cycle (see Figure 1) as a recursive process that is responsive to the learning environment and the subject matter being learned [8]. Learning styles describe the unique ways individuals spiral through the learning cycle based on their preference for four different learning modes [8] as shown in Figure 1.

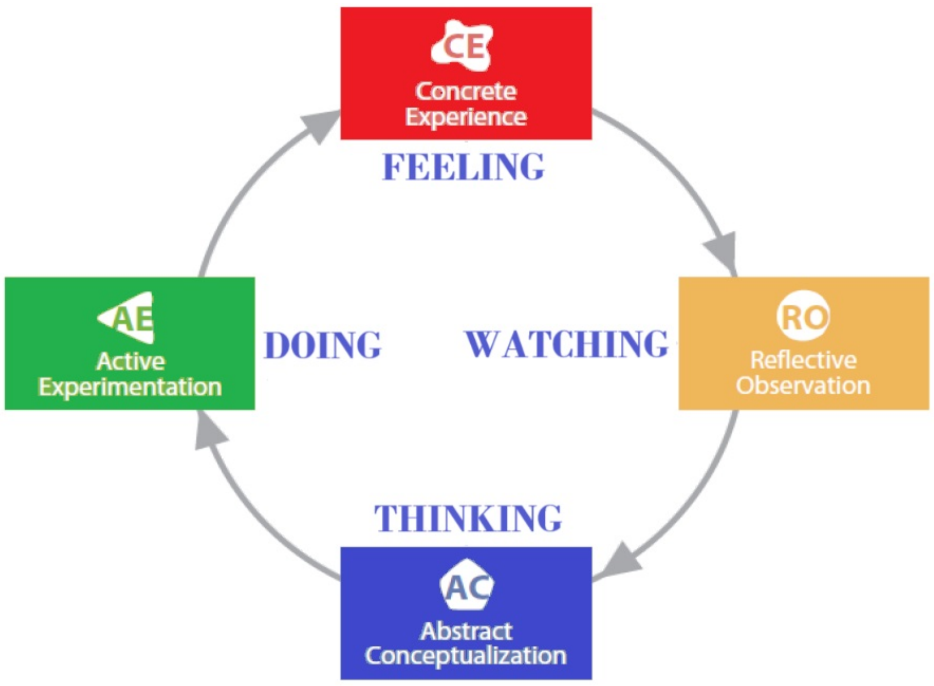

Figure 1: The Experiential Learning Cycle [8]

However, data from empirical and clinical studies over the years has shown that these original four learning style types can be further refined into a nine-style typology [8] — Initiating, Imagining, Reflecting, Analyzing, Thinking, Deciding, Acting, and Balancing — known as the Kolb's Learning Style Inventory (see Figure 2), to define various approaches or modes of engaging with the learning process, each with its distinct characteristics and preferences.

Kolb's Learning Style Inventory (KLSI) builds on the original four learning modes, expanding them into nine distinct styles:

- **Initiating:** This style is marked by the ability to take action in response to experiences and situations, integrating both active experimentation and concrete experience.
- **Experiencing:** Characterized by a deep engagement with experiences to extract meaning, this style combines concrete experience while balancing both active experimentation and reflective observation.
- **Imagining:** Defined by the ability to envision possibilities through observation and reflection on experiences, this style blends concrete experience with reflective observation.
- **Reflecting:** Distinguished by the ability to connect experiences with ideas through extended reflection, this style merges reflective observation with both concrete experience and abstract conceptualization.
- **Analyzing:** This style involves structuring and systematizing ideas through reflection, combining reflective observation with abstract conceptualization.
- **Thinking:** Defined by the ability for focused, abstract, and logical reasoning, this style uses abstract conceptualization while balancing both active experimentation and reflective observation.
- **Deciding:** This style involves using theoretical models and frameworks to make decisions and determine actions, blending abstract conceptualization with active experimentation.

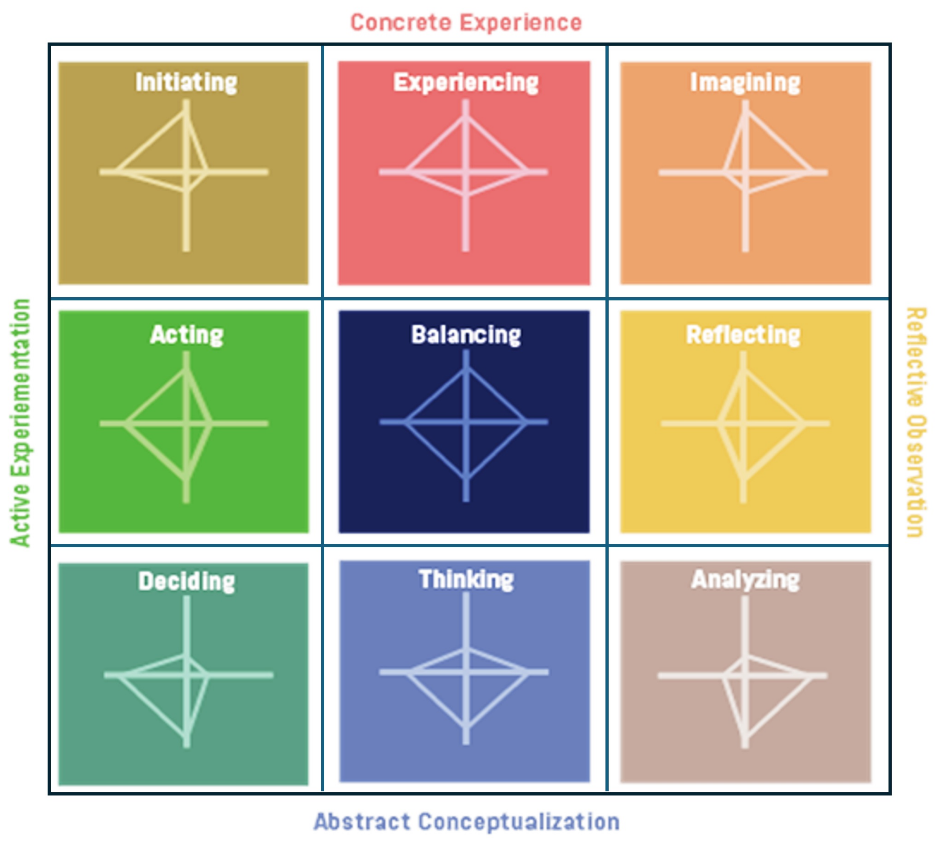

Figure 2: The Nine KLSI Learning Styles [8]

- **Acting:** Characterized by a strong drive for goal-oriented actions that integrate both people and tasks, this style emphasizes active experimentation, with a balance of concrete experience and abstract conceptualization.
- **Balancing:** Defined by adaptability, this style involves weighing the benefits of action versus reflection, as well as experience versus thinking, balancing all four learning modes: concrete experience, abstract conceptualization, active experimentation, and reflective observation.

To better illustrate how different teaching approaches align with the Kolb Learning Style Inventory (KLSI), we present a mapping in Table 1 to highlight how various educational strategies support different learning modes, providing a more explicit link between theoretical models and practical applications.

Table 1: Mapping of teaching approaches to KLSI learning modes

Teaching Approach	**Associated KLSI Dimension(s)**
Role-playing in requirements elicitation	Concrete Experience, Active Experimentation
Problem-based learning	Reflective Observation, Active Experimentation
Project-based teamwork	Concrete Experience, Abstract Conceptualization
Case studies in software design	Abstract Conceptualization, Reflective Observation
Simulations and modeling exercises	Active Experimentation, Abstract Conceptualization

Given the skills necessary and the conceptual framework provided by KLSI, Software Engineering Education (SEE) can benefit from identifying the predominant learning style of software development practitioners. By aligning instructional methods with their preferred learning styles, this approach can foster more effective skill development.

3 Related Work

The research regarding the application of Kolb's Learning Style Inventory (KLSI) in educational contexts, particularly in understanding individual learning preferences and styles, takes into account key insights at the outset of role identification:

1. Kolb's Learning Style Inventory demonstrates reliability across different cultures and languages, as evidenced by the successful adaptation and validation of the Hebrew version, which exhibited similar reliability and internal consistency to the original English version [9].

2. The KLSI effectively distinguishes among students' major study areas, supporting its utilization in identifying individual learning styles and potentially guiding role identification within educational contexts [9].

3. Utilizing the KLSI in educational settings, including software engineering education, offers insights into the learning styles of students, faculty, and field instructors, with potential implications for optimizing learning experiences and role assignments [10].

The Kolb Learning Style Inventory (KLSI) has played a pivotal role in understanding and categorizing students' learning styles, spanning various educational contexts, including software engineering. It has garnered widespread acceptance and empirical support, undergoing iterative refinements to address limitations and enhance its utility. For instance, a study by [10] adapted the KLSI into a continuous measure, proving more concise and user-friendly through multi-sample studies with graduate students in engineering and computer science. Moreover, investigations into the relationship between KLSI-assessed learning styles and student interaction in asynchronous computer-mediated conferencing (CMC) confirmed theoretical predictions of the Kolb model [11], highlighting the influence of learning styles on communication behavior in online learning environments.

Efforts to bolster the reliability of Kolb's Revised Learning Style Inventory have yielded modifications significantly improving its stability, as evidenced by higher test-retest reliabilities and kappa coefficients [12], further affirming its value for studying learning styles. In software engineering education, accommodating diverse learning styles among students has been a challenge, addressed through pilot initiatives adapting teaching methods to varied learning habits [13]. Moreover, broader reviews of learning style instruments, including the KLSI, emphasize the importance of diverse teaching approaches to cater to student diversity [14]. An exploratory study utilizing the KLSI highlighted the significance of understanding how students learn, suggesting its utility in aligning teaching strategies with student preferences in both field and classroom experiences [15].

In an emerging research field, efforts have been made to identify, group, and define technical and soft skills for software engineers, facilitating the development of comprehensive frameworks to enhance undergraduate education in Software Engineering [16]. Recent pilot efforts have even utilized the KLSI to characterize the learning styles of Software Architects, proving instrumental in refining teaching and learning strategies tailored to this specific role [17, 18].

In summary, the KLSI serves as a valuable tool in identifying and accommodating diverse learning styles in software engineering education. Its continual evolution and application across educational settings underscore its relevance in enhancing the learning experience for students in this field, complementing efforts to establish benchmarks and assess progress in Software Engineering Education.

4 Research Method

This section presents an overview of the survey used as a research method and discusses aspects related to its administration, inclusion and exclusion criteria, assumptions, and characterization of the survey respondents.

4.1 Kolb's Learning Style Test Survey

To address our research questions and guide our study, we relied on the insights presented in [1, 19], particularly in defining evaluation criteria and data analysis methods. In particular, we adopted the Guidelines for Case Survey Research [19] by applying inclusion/exclusion criteria to reduce bias, and on data analysis methods by employing descriptive statistics, t-tests, and visual representations such as radial graphs taking into account the Challenges in Survey Research [19]. Additionally, we focus our roadmap considering to provide effective means for software engineering students to keep their skills current [1] by crosscutting teaching strategies. Considering the nature of our investigation, we opted to utilize a survey, specifically the Kolb's Learning Styles Test, for data collection. The survey was made available in both English and Spanish to ensure that participants could respond in their preferred language, minimizing potential language bias, and the test consists of 12 sentence completion questions (see Table 2), where participants are asked to rank the provided suffixes on a scale of 1 to 4, with 4 indicating the highest preference. This format allows us to evaluate the extent to which participants lean towards the four learning modes and enables us to categorize each participant into a learning style [8].

Table 2: Kolb's test applied

PREFIX	SUFFIX
Q1. "When I learn..."	I prefer to rely on my sensations and feelings
	I prefer to look and pay attention
	I prefer to think about ideas
	I prefer to do things
Q2. "I learn best when..."	I trust my hunches and feelings
	I listen and observe carefully
	I trust my logical thoughts
	I work hard to get things done
Q3. "When I am learning..."	I have strong feelings and reactions
	I am reserved and calm
	I seek to reason about things that are happening
	I feel responsible for things
Q4. "I learn through..."	Feelings
	Observations
	Reasonings
	Actions
Q5. "When I learn..."	I am open to new experiences
	I take into account all related aspects
	I prefer to analyze things by breaking them down into their component parts
	I prefer to do things directly
Q6. "When I'm learning..."	I am an intuitive person
	I am an observant person
	I am a logical person
	I am an active person
Q7. "I learn best through..."	Relationships with my peers
	The observation
	Rational Theories
	The practice of the topics covered
Q8. "When I learn..."	I feel involved in the topics covered
	I take my time before acting
	I prefer theories and ideas
	I prefer to see the results through my own work
Q9. "I learn best when..."	I rely on my intuitions and feelings
	I rely on personal observations
	I take into account my own ideas about the subject matter
	I personally try out the task
Q10. "When I am learning..."	I am open-minded
	I am a reserved person
	I am a rational person
	I am a responsible person

Continued on next page

Table 2: Kolb's test applied (Continued)

Q11. "When I learn…"	I get involved
	I prefer to observe
	I prefer to evaluate things
	I prefer to take an active attitude
Q12. "I learn best when…"	I am receptive and open-minded
	I am careful
	I analyze ideas
	I am practical

Considering the survey results, the respondent can be characterized based on these modes, and a radial graph can illustrate their learning style. Within this framework, individuals positioned closer to the outer corners of the graph exhibit a stronger inclination toward the respective learning mode, and this categorization corresponds to one of the nine Learning Styles identified in the KLSI.

4.2 Inclusion/Exclusion Criteria

Our research established the following inclusion/exclusion criteria for the participants:

4.2.1 Inclusion Criteria

To qualify for participation in the study, respondents must meet the following criteria:

- They must be software development practitioners.
- They must not have previously completed the Kolb's Learning Styles Test.

4.2.2 Exclusion Criteria

To uphold the study's integrity and mitigate potential biases, individuals will be excluded from participation if they meet any of the following conditions:

- Lack of proficiency in written English or Spanish.
- Work directly with any of the researchers involved.

Therefore, our research is based on the following assumptions:

- **Assumption 1:** The sample's heterogeneity, regarding professionals' roles and experience, will provide comprehensive insight into learning styles, facilitating the identification of representative trends.
- **Assumption 2:** The Kolb test, designed for adults, applies to professionals without necessitating adjustments or modifications.

4.3 Test Execution Process

For our research, the execution of the survey considered the following aspects:

4.3.1 Test Administration

The test was administered voluntarily, and responses were collected anonymously using a link to access the test. Invitations were extended via LinkedIn to individuals currently employed in Software Development, inviting them to participate.

4.3.2 ***Survey Sample Characterization***

The test was administered over 60 days, from the second week of January to the first week of March. Ultimately, a sample of 112 participants [1] from Chile, Costa Rica, Argentina, Colombia, and Spain who met the criteria was obtained. The details are as follows:

- The sample included 15 Analysts, 50 Developers, 27 Project Managers, 2 Quality Assurance professionals, and 18 Software Architects, as shown in Figure 3.

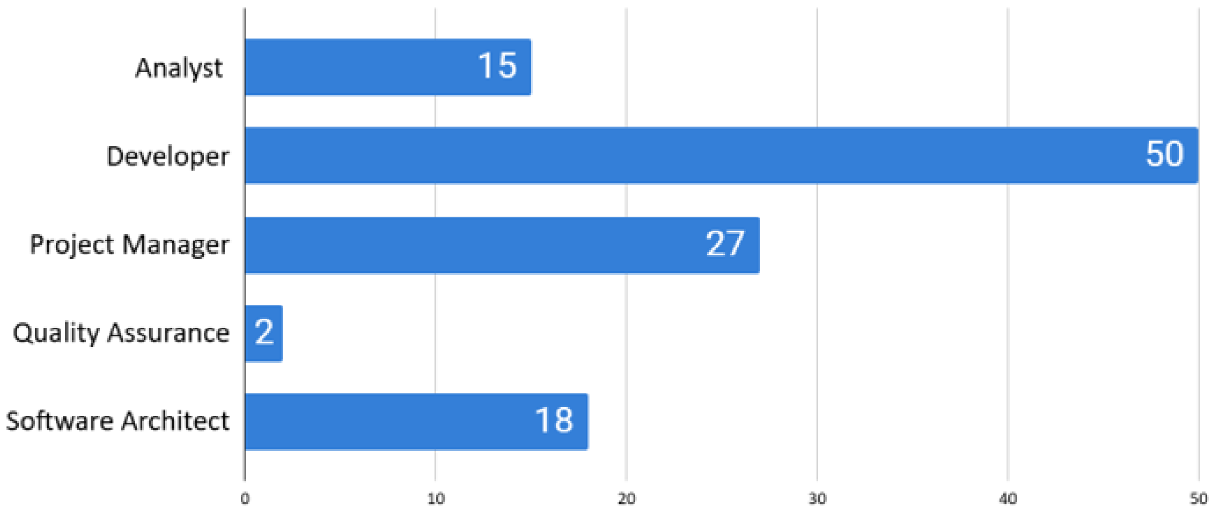

Figure 3: Practitioners per Role

- Consists of 70 Senior professionals, 24 Mid-Senior professionals, and 18 Junior professionals. Their percentage representation is shown in Figure 4.

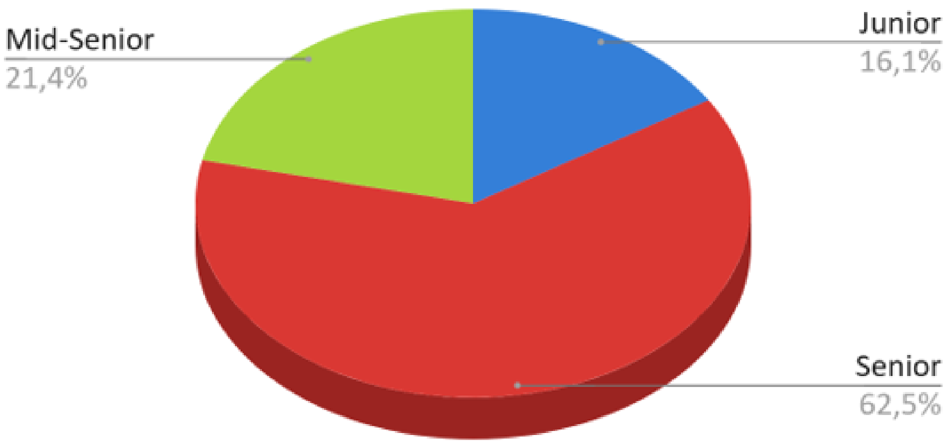

Figure 4: Practitioners' Experience

- It included 46 practitioners from Chile, 23 from Argentina, 16 from Spain, 15 from Costa Rica, and 12 from Colombia, as shown in Figure 5.

[1]The conference paper [18] did not include QA-based analysis, whereas this extended study incorporates it to explore transversal aspects in more depth.

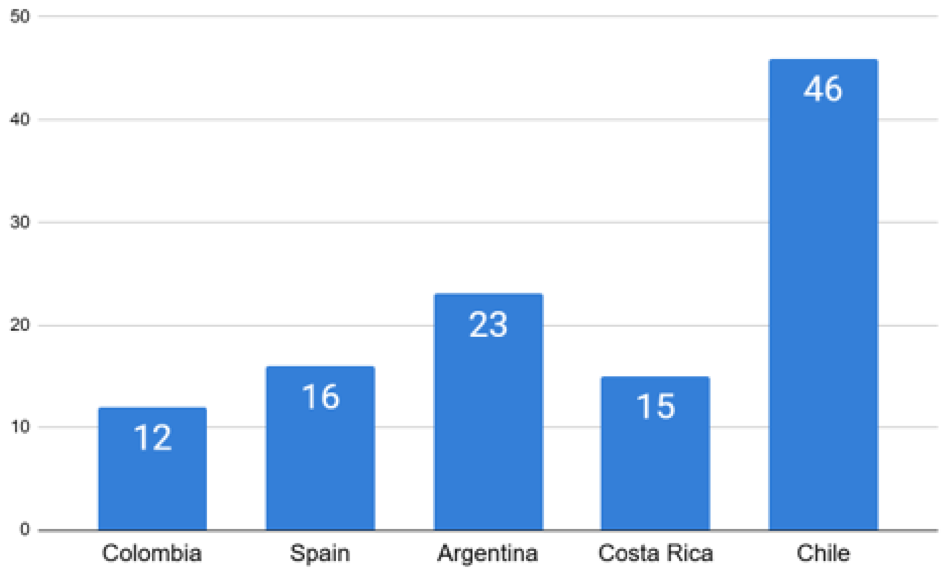

Figure 5: Practitioners per country

5 Research results

After conducting the test, the scores for each learning mode of all practitioners were calculated (see Appendix 1). Based on these scores, the mean, median, and standard deviation for the entire sample were determined, as presented in Table 3.

Table 3: Survey results statistics

LEARNING MODE	**CE**	**RO**	**AC**	**AE**
MEAN	24.86	29.84	33.30	32.00
MEDIAN	24.50	30.00	33.50	32.00
MEAN-MEDIAN DIFFERENCE	0.36	0.16	0.20	0.00
STD DEVIATION	5.46	5.66	5.33	6.06

To determine the validity of the mean as a representative measure of each learning mode for the sample, we employ the t-student test [20] and define our hypotheses H_0 and H_1 as follows:

- $H_0 \rightarrow$ The mean of each learning mode is not representative of the Sample.
- $H_1 \rightarrow$ The mean of each learning mode is representative of the SPMs Sample [2].

Searching the Student's t-distribution table (two-tailed) with a significance level of 0.05 (corresponding to 95% confidence) and 111 degrees of freedom ($112 - 1 = 111$), the critical value for the sample is ± 0.96. To determine the t-statistic value for each learning mode, we will use the formula:

$$t = \frac{\bar{x} - \mu}{s \div \sqrt{n}}$$

where

- $\bar{x}$ represents the sample mean,
- μ represents the hypothetical population mean (median),
- s represents the sample standard deviation,
- n represents the sample size.

Table 4: t-statistic value for each Learning Mode

Learning Mode	**t-statistic value**
Concrete Experience (CE)	0.69
Reflective Observation (RO)	-0.30
Abstract Conceptualization (AC)	-0.39
Active Experimentation (AE)	0.00

Utilizing the scores from Table 3 and taking the median as our hypothetical population mean, the results of the t-statistic value for each mode can be observed in Table 4.

Finally, since the statistical value for each learning mode falls within the acceptance interval determined by the critical value of the Student's t-distribution, ± 0.96, we reject our H_0 hypothesis and accept H_1. This indicates that the sample mean is representative of each learning mode in the sample and serves to define the most representative learning style.

5.1 Software Development Practitioners Learning Style

To address our **R.Q.1**, based on the mode values presented in Table 3, we observe that the AC mode exhibits the strongest inclination, indicating a preference for theoretical analysis. However, there is a balance between the AE and RO modes, suggesting that practitioners also show a significant preference for the practical application of learning and for considering multiple perspectives before taking action. The CE mode, while still present, shows a less pronounced preference for learning through hands-on experience compared to the other modes. Overall, although the modes are not perfectly balanced, all four are notably represented, leading to the classification of software development practitioners under the Thinking Learning Style, as illustrated in Figure 6.

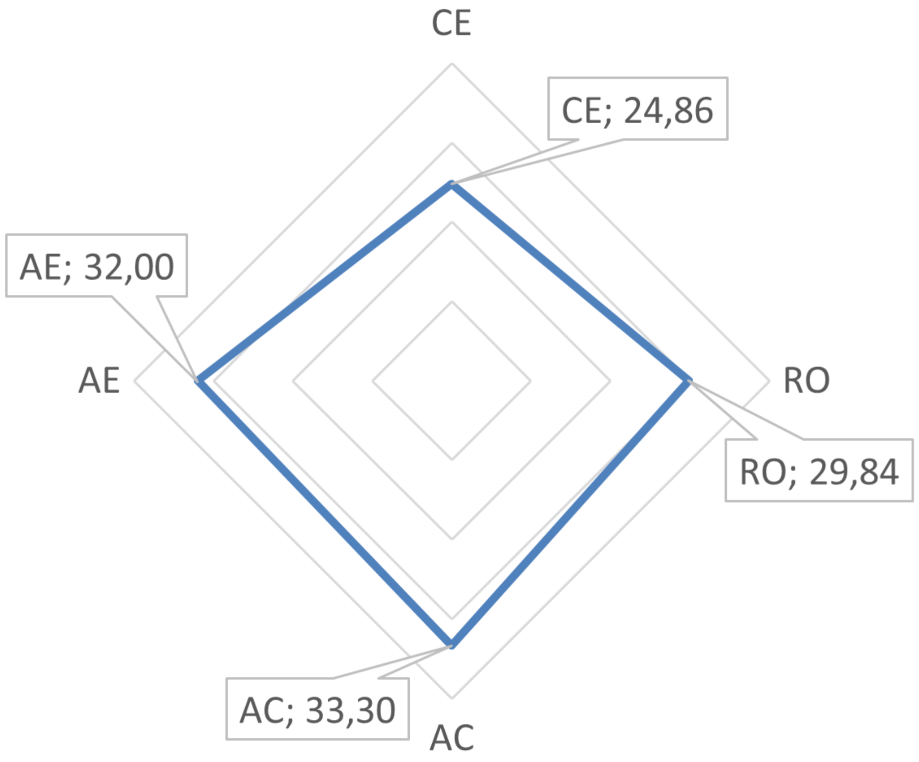

Figure 6: Thinking: Predominant learning style radial graph

The Thinking Learning Style emphasizes Abstract Conceptualization (AC) while maintaining a balance between Active Experimentation (AE) and Reflective Observation (RO). This style is marked by a strong ability for disciplined engagement in abstract reasoning, mathematics, and logic [8]. Individuals with the Thinking style typically enjoy working with numbers and engaging in tasks that require abstract reasoning

[2]By "representative of the sample," we refer to the concept that the sample mean accurately reflects the central tendency of the sample's learning modes.

and analytical skills. They tend to prefer working with quantitative data over qualitative information [8]. Additionally, they excel at planning and goal-setting, express emotions in a controlled way, and favor precise and concise communication [8]. Moreover, individuals with the Thinking style are particularly strong in logical analysis, rational decision-making, and the ability to analyze quantitative data effectively.

5.2 Software Development Teams Teaching Strategies

To address our **R.Q.2**, it is essential to recognize that the "Thinking Learning Style" is characterized by specific traits [8]:

- Prefers abstraction and logical reasoning, excelling in analytical tasks.
- Enjoys working with numbers and quantitative data.
- Thrives in structured environments with clear objectives.
- Focuses on precision and coherence in their approach.

When teaching software development teams with this learning style, educators should create a structured environment that aligns with these preferences. By emphasizing logical analysis, systematic planning, and decision-making, educators can maximize the strengths of this style. At the same time, it is important to address potential challenges, such as fostering collaboration and encouraging introspection. Recommended activities for such teams include:

1. **Problem-solving exercises** [21, 22] that involve logical analysis and the application of abstract concepts, encouraging students to break down complex problems into manageable components.
2. **Data-driven tasks** [23] such as analyzing code performance or optimizing algorithms, which allow students to work with quantitative data and hone their analytical skills.
3. **Structured group discussions** [24, 25] that promote clear communication and precise explanations, ensuring that all team members are aligned in their goals and approaches.
4. **Simulated real-world scenarios** [2] with defined objectives, where students must apply theoretical knowledge to solve practical problems, enhancing their ability to focus on results while maintaining logical rigor.

Additionally, strategies like **Problem-Based Learning** and **Project-Based Learning** provide the opportunity to bridge the gap between theory and practice, allowing students to engage with complex, open-ended problems. These methodologies encourage students to:

- Apply theoretical knowledge to real-world challenges, promoting critical thinking and practical problem-solving.
- Collaborate within teams, enhancing communication and teamwork skills, even for those with a preference for analytical thinking.
- Develop self-directed learning skills, as students are encouraged to explore and research solutions independently within a structured framework.

To provide a clearer overview of the learning preferences of software development practitioners and the recommended pedagogical strategies, two tables are presented below. Table 5 summarizes the learning preferences of professionals in software development, along with recommended pedagogical strategies to maximize their effectiveness.

Table 6 highlights key educational strategies and the expected outcomes of their implementation in software development teams.

By integrating these teaching strategies, educators can foster a learning environment that caters to the Thinking Learning Style, helping students develop both their technical and interpersonal skills while ensuring that they are well-prepared for the dynamic challenges of software development.

6 External Validation

The objective of the external validation was to assess the applicability and relevance of the proposed educational strategies by engaging with professional training consultancies that specialize in software development. This step aimed to gather expert feedback on the feasibility of the strategies and their alignment with industry needs.

Table 5: Summary of Practitioners' Learning Preferences and Recommended Pedagogical Strategies

Learning Preferences	**Recommended Pedagogical Strategies**
Preference for abstract reasoning and logic	Strategies focused on logical analysis and solving complex problems.
Enjoy working with numbers and quantitative data	Activities based on data analysis and algorithm optimization.
Thrive in structured environments with clear objectives	Use of structured projects with defined goals and clear steps to achieve them.
Focus on precision and coherence in their approach	Promotion of precise communication and detailed documentation in teamwork.

Table 6: Key Educational Strategies and Their Expected Outcomes

Educational Strategy	**Expected Outcomes**
Problem-Based Learning	Encourages critical problem-solving and applying theory to practice.
Project-Based Learning	Enhances collaboration and communication skills, develops leadership abilities.
Structured Group Discussions	Aligns team objectives, improve communication and decision-making.
Quantitative Problem-Solving Tasks	Reinforces data analysis and evidence-based decision-making.
Real-World Scenario Simulations	Applies theoretical knowledge to practical situations and develops adaptability skills.

6.1 Methodology

The validation process involved a focus group session [19] conducted with representatives from three professional training consultancy firms (ref. partial transcription in Appendix 2). These organizations are known for designing and implementing training programs for software development teams in various industries.

Participants: The focus group included six participants:

- Two senior consultants from a specialized corporate training firm.
- Two instructional designers from postgraduate programs focusing on technical skill development.
- Two technical leads from an OTEC[3], experienced in software engineering education.

Procedure: The session was structured into three parts:

- Presentation of Results: A brief overview of the research findings, including the identification of the "Thinking Learning Style" and the proposed strategies.
- Interactive Discussion: Participants provided their perspectives on the feasibility, strengths, and potential limitations of the strategies in real-world training scenarios.
- Feedback Collection: Participants filled out a structured feedback form to rate and comment on specific aspects of the strategies.

Guiding Questions: The guiding questions were designed to elicit in-depth feedback from the participants, encouraging them to reflect on their experiences and evaluate the proposed strategies critically. These were:

- Are the identified learning styles representative of your experience with software development teams?
- How practical are the proposed strategies in corporate training environments?
- What adjustments would you recommend to improve their implementation?

[3]In Chile, OTEC is the abbreviation of *Organismos Técnicos de Capacitación* (Technical Training Organizations)

6.2 Results of the Focus Group

To provide further evidence, we include a brief synthesis of key discussion points from the focus group, along with relevant quotes illustrating participant perspectives.

1. **Positive Feedback**
    - The consultants agreed that the "Thinking Learning Style" accurately reflects the cognitive preferences of many software developers, particularly those in roles requiring analytical and problem-solving skills.
    - The strategies were deemed practical and aligned with common industry challenges, such as enhancing algorithm optimization and team communication.
2. **Suggestions for Improvement**
    - Incorporate more collaborative activities to address the challenges of fostering teamwork among analytical learners.
    - Adapt some strategies for remote learning environments, as many training programs are now delivered online.
    - Develop more case-based scenarios tailored to specific roles, such as quality assurance or software architects.

6.3 Implications for the Study

The feedback validated the core premise of the research while highlighting areas for refinement. Incorporating these suggestions will enhance the versatility and effectiveness of the proposed strategies, making them more applicable across diverse training contexts.

7 Limitations and Threats to Validity

The robustness and validity of this study are subject to certain limitations and potential threats, which are outlined below to ensure a transparent understanding of the factors influencing our findings and conclusions.

- **Sample size:** One of the limitations of this study is the small sample size, which may affect the generalizability of the findings. This is acknowledged as a potential threat to validity. While the sample size is limited, the study's findings are derived from clear trends observed in the collected data [19]. The statistical analysis further supports the reliability of these results by demonstrating a well-defined data distribution. The minimal differences between the mean and median suggest a nearly symmetric distribution, offering an accurate representation of the target population. Additionally, the moderate standard deviation provides valuable insight into data dispersion, enhancing the robustness of the findings across both partial and complete samples.
- **Sample diversity:** Although heterogeneous samples can pose challenges in some studies, the diversity of professional roles and experiences included in this research was instrumental in uncovering meaningful trends. This variety enriched the analysis by capturing a broader spectrum of perspectives within the software development field.
- **Role Imbalance:** The over-representation of developers (50 participants) compared to other roles, such as analysts, project managers, and QA professionals, may have influenced the identification of the "Thinking Learning Style" as the predominant style. Nonetheless, previous research by the authors [18] provides a more detailed examination of cognitive differences across these roles, supporting the interpretation of these findings.
- **Response bias mitigation:** While respondent bias is a potential risk in any survey-based study, Kolb's test employed here is specifically designed to minimize such effects. The inclusion of neutral prefixes ensures balanced responses, reducing the likelihood of skewed results and reinforcing the validity of the collected data.

8 Discussion

The findings of this study provide important insights into the cognitive preferences of software development professionals, revealing that the ”Thinking Learning Style” is the most representative within the surveyed sample. This result underscores the tendency of professionals in this field to engage in abstract reasoning, logical analysis, and systematic problem-solving. The dominance of Abstract Conceptualization, coupled with a balance between Active Experimentation and Reflective Observation, highlights the dual need for both theoretical understanding and practical application in software engineering training. This balanced profile suggests that while professionals excel at formulating abstract ideas, they also value hands-on experimentation and reflective practices that enable them to adapt solutions to complex problems.

The statistical analysis confirmed the representativeness of the learning modes, validating the robustness of the survey data. The minimal variance between the mean and median of the modes reinforces the consistency of the sample's cognitive tendencies, ensuring that the identified learning style is not an anomaly but a reliable reflection of the broader population. Additionally, the external validation through the focus group with professional training consultancies further confirmed the relevance of these findings. Experts agreed that the "Thinking Learning Style" accurately reflects the cognitive tendencies they observe in software development teams, particularly in roles requiring high analytical capabilities. Moreover, the focus group provided valuable recommendations for enhancing the strategies, such as incorporating more collaborative activities to foster teamwork and adapting methods for remote learning environments, which are increasingly prevalent in professional training contexts. These insights align with broader discussions in software engineering education about the importance of integrating technical and interpersonal skills development in training programs [8, 9, 19].

In summary, these findings and validations provide a robust foundation for designing educational strategies that align with the cognitive strengths of software practitioners. Tailoring approaches to the "Thinking Learning Style" can enhance learning outcomes by leveraging professionals' strengths in abstract reasoning and systematic problem-solving while addressing the challenges of collaboration and adaptability in dynamic software development environments.

9 Conclusion and Future Work

This study has identified the "Thinking Learning Style" as the predominant cognitive preference among software development professionals. Specifically, this finding emphasizes their strong inclination towards abstract reasoning, logical analysis, and systematic problem-solving. These cognitive strengths serve as a crucial foundation for the design of educational strategies tailored to the specific needs of software professionals. In this regard, the results highlight the importance of aligning teaching methods with the cognitive profiles of learners to enhance their engagement and learning outcomes. Furthermore, the proposed teaching strategies, which were developed based on the findings of this study, were validated through an external focus group with professional training consultancies. The feedback received confirmed that the strategies are not only practical but also highly relevant to the real-world challenges faced by software development teams. This validation process has reinforced the alignment of our approach with industry needs and has provided valuable insights into areas for refinement, particularly the inclusion of collaborative activities and the adaptation of methods for remote learning environments. These enhancements are essential in addressing the evolving nature of software engineering education, where flexibility and adaptability are key.

In addition, the statistical analysis performed on the survey data has confirmed the robustness and reliability of the findings. The minimal variance between the mean and median values, as well as the acceptance of the null hypothesis in t-tests, further supports the credibility of the results. This statistical consistency assures that the identified learning style is not an anomaly but a dependable reflection of the broader population of software professionals. Ultimately, by tailoring training approaches to the cognitive preferences of software professionals, both educators and organizations can foster more effective learning environments. These environments will enhance skill acquisition, retention, and professional development, which are crucial for addressing the increasing demands of the software industry. This study provides a clear path toward bridging the gap between academic education and industry-specific training practices, creating a more integrated and cohesive learning ecosystem.

In conclusion, this research contributes significantly to the ongoing discourse on software engineering education. By demonstrating the importance of understanding and leveraging cognitive preferences, the study underlines how personalized, adaptive strategies can transform training programs. The integration of both theoretical knowledge and practical application, along with the adaptability to dynamic learning environments, ensures that these strategies will remain relevant and impactful in the professional development of software engineers.

In terms of future work, we will focus on applying the proposed educational strategies in real-world software development environments, with direct collaboration from the OTEC that participated in the external validation process. Additionally, we will explore the adaptation of these strategies for remote learning environments to address the increasing demand for flexible and online professional training in the software industry.

Furthermore, although the analysis conducted considers IT professionals in general, it is highly relevant to carry out a comparative study between professionals with and without a formal background in the industry to understand how their educational foundations influence their learning preferences and professional development. Another relevant direction for future research is to consider demographic and professional factors such as age and seniority. These variables may influence learning preferences and perceptions of educational strategies. Their inclusion in a larger, more diverse sample could provide more nuanced insights, enabling a better understanding of how different profiles respond to various teaching methodologies.

Acknowledgments

The authors extend their gratitude to all the practitioners who supported us by responding to the survey for this study, as well as to the experts who participated in the external validation process and provided valuable insights.

References

[1] M. Shaw, "Software engineering education: a roadmap," in *Proceedings of the Conference on The Future of Software Engineering*, ser. ICSE '00. New York, NY, USA: Association for Computing Machinery, 2000, p. 371–380. [Online]. Available: https://doi.org/10.1145/336512.336592

[2] S. Ouhbi and N. Pombo, "Software engineering education: Challenges and perspectives," in *2020 IEEE Global Engineering Education Conference (EDUCON)*, 2020, pp. 202–209.

[3] M. Jazayeri, "Education of a software engineer," in *Keynote presentation at ASE 2004*, Linz, Austria, 2004.

[4] J. Liebenberg, M. Huisman, and E. Mentz, "Industry's perception of the relevance of software development education," *The Journal for Transdisciplinary Research in Southern Africa*, vol. 11, p. 25, 2015.

[5] C. Portela, A. Vasconcelos, S. Oliveira, and M. Souza, "The use of industry training strategies in a software engineering course: An experience report," in *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*, 2017, pp. 29–36.

[6] B. Prommegger, M. Wendrich, M. Wiesche, and H. Krcmar, "Short-term affair or long-term commitment? an investigation of employees without it background in it jobs," in *Proceedings of the 2020 Computers and People Research Conference*, ser. SIGMIS-CPR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 91–98.

[7] O. Cico, M. Jaccheri, A. Nguyen-Duc, and H. Zhang, "Exploring the intersection between software industry and software engineering education - a systematic mapping of software engineering trends," *Journal of Systems and Software*, vol. 172, p. 110736, 2021.

[8] D. Kolb and A. Kolb, *The Kolb Learning Style Inventory 4.0: Guide to Theory, Psychometrics, Research & Applications.* Experience Based Learning Systems, LLC, 01 2013.

[9] N. Katz, "Individual learning style: Israeli norms and cross-cultural equivalence of kolb's learning style inventory," *Journal of Cross-Cultural Psychology*, vol. 19, no. 3, pp. 361–379, 1988.

[10] B. J. F. Jean M. Kruzich and D. V. Soest, "Assessment of student and faculty learning styles: Research and application," *Journal of Social Work Education*, vol. 22, no. 3, pp. 22–30, 1986.

[11] I. John G. Veres, R. R. Sims, and T. S. Locklear, "Improving the reliability of kolb's revised learning style inventory," *Educational and Psychological Measurement*, vol. 51, no. 1, pp. 143–150, 1991.

[12] C. Manolis, D. J. Burns, R. Assudani, and R. Chinta, "Assessing experiential learning styles: A methodological reconstruction and validation of the kolb learning style inventory," *Learning and Individual Differences*, vol. 23, pp. 44–52, 2013.

[13] P. J. Fahy and M. Ally, “Student learning style and asynchronous computer-mediated conferencing (cmc) interaction,” *American Journal of Distance Education*, vol. 19, no. 1, pp. 5–22, 2005.

[14] T. F. Hawk and A. J. Shah, “Using learning style instruments to enhance student learning,” *Decision Sciences Journal of Innovative Education*, vol. 5, no. 1, pp. 1–19, 2007.

[15] N. Waibel, Y. Sedelmaier, and D. Landes, “Using learning styles to accommodate for heterogeneous groups of learners in software engineering,” in *2020 IEEE Global Engineering Education Conference (EDUCON)*, 2020, pp. 819–826.

[16] G. G. Borges and R. C. Gratão de Souza, “Skills development for software engineers: Systematic literature review,” *Information and Software Technology*, vol. 168, p. 107395, 2024.

[17] M. Hidalgo, H. Astudillo, and L. M. Castro, “How software architects learn: A pilot study of their learning style in kolb’s learning styles inventory,” in *2023 42nd IEEE International Conference of the Chilean Computer Science Society (SCCC)*, 2023, pp. 1–8.

[18] M. Hidalgo, K. Rodriguez, and H. Astudillo, “Mapping kolb’s learning style to roles in software development team,” in *2024 L Latin American Computer Conference (CLEI)*, 2024, pp. 1–9.

[19] M. Felderer and G. H. Travassos, Eds., *Contemporary Empirical Methods in Software Engineering*, 1st ed. Cham, Switzerland: Springer Cham, 2020, part I, Chapters 2-3. eBook Packages: Computer Science, Computer Science (R0).

[20] H. W. Thompson, R. Mera, and C. Prasad, “A description of the appropriate use of student’s t-test,” *Nutritional Neuroscience*, vol. 1, no. 2, pp. 165–172, 1998. [Online]. Available: https://doi.org/10.1080/1028415X.1998.11747226

[21] Y.-T. Lin, “Impacts of a flipped classroom with a smart learning diagnosis system on students’ learning performance, perception, and problem solving ability in a software engineering course,” *Computers in Human Behavior*, vol. 95, pp. 187–196, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0747563218305715

[22] D. Topalli and N. E. Cagiltay, “Improving programming skills in engineering education through problem-based game projects with scratch,” *Computers & Education*, vol. 120, pp. 64–74, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360131518300113

[23] J. Lin, H. Yu, Z. Pan, Z. Shen, and L. Cui, “Towards data-driven software engineering skills assessment,” *International Journal of Crowd Science*, vol. 2, no. 2, pp. 123–135, 2018.

[24] M. Marques, S. F. Ochoa, M. C. Bastarrica, and F. J. Gutierrez, “Enhancing the student learning experience in software engineering project courses,” *IEEE Transactions on Education*, vol. 61, no. 1, pp. 63–73, 2018.

[25] J. P. Landry, J. H. Pardue, M. V. Doran, and R. J. Daigle, “Encouraging students to adopt software engineering methodologies: The influence of structured group labs on beliefs and attitudes,” *Journal of Engineering Education*, vol. 91, no. 1, pp. 103–108, 2002. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2168-9830.2002.tb00678.x

Appendix 1 - Scores for each learning mode of all the sample

Table 7: Practitioners' score for Learning Mode

N°	Role	Experience	Country	CE	RO	AC	AE
1	Analyst	Junior	Colombia	32	21	31	36
2	Analyst	Senior	Spain	21	23	46	30
3	Analyst	Junior	Argentina	25	24	26	45
4	Analyst	Junior	Costa Rica	26	33	40	21
5	Analyst	Mid-Senior	Spain	33	38	27	22
6	Analyst	Junior	Argentina	32	34	27	27
7	Analyst	Junior	Costa Rica	22	40	39	19
8	Analyst	Senior	Colombia	19	35	36	30
9	Analyst	Junior	Chile	25	29	34	32
10	Analyst	Junior	Spain	33	23	27	37
11	Analyst	Senior	Argentina	13	29	40	38
12	Analyst	Mid-Senior	Chile	28	34	23	35
13	Analyst	Junior	Spain	26	33	25	36
14	Analyst	Senior	Chile	16	34	29	41
15	Analyst	Senior	Chile	28	22	35	35
16	Developer	Mid-Senior	Chile	22	26	40	32
17	Developer	Mid-Senior	Chile	25	30	30	35
18	Developer	Mid-Senior	Costa Rica	19	34	34	33
19	Developer	Senior	Argentina	27	32	34	27
20	Developer	Senior	Chile	19	33	33	35
21	Developer	Senior	Chile	21	37	41	21
22	Developer	Mid-Senior	Argentina	27	22	34	37
23	Developer	Junior	Costa Rica	23	39	33	25
24	Developer	Junior	Chile	29	26	29	36
25	Developer	Mid-Senior	Spain	21	37	31	31
26	Developer	Junior	Colombia	28	29	28	35
27	Developer	Mid-Senior	Chile	21	36	32	31
28	Developer	Junior	Spain	31	26	28	35
29	Developer	Senior	Chile	27	33	29	31
30	Developer	Senior	Costa Rica	29	22	34	35
31	Developer	Junior	Colombia	13	40	31	36
32	Developer	Mid-Senior	Chile	21	30	29	40
33	Developer	Senior	Colombia	26	33	35	26
34	Developer	Junior	Chile	26	28	28	38
35	Developer	Mid-Senior	Argentina	21	26	31	42
36	Developer	Mid-Senior	Chile	29	26	39	26
37	Developer	Mid-Senior	Colombia	36	30	27	27
38	Developer	Senior	Spain	19	31	38	32
39	Developer	Mid-Senior	Argentina	32	28	38	22
40	Developer	Junior	Chile	20	32	32	36
41	Developer	Senior	Argentina	25	27	28	40
42	Developer	Senior	Chile	17	35	36	32
43	Developer	Senior	Argentina	19	30	42	29
44	Developer	Senior	Spain	33	20	33	34
45	Developer	Senior	Argentina	25	23	42	30
46	Developer	Senior	Chile	45	23	27	25
47	Developer	Mid-Senior	Costa Rica	30	31	34	25
48	Developer	Senior	Colombia	19	33	32	36
49	Developer	Senior	Chile	31	27	24	38
50	Developer	Senior	Spain	22	28	35	35
51	Developer	Senior	Chile	25	25	31	39
52	Developer	Senior	Argentina	30	36	34	20

53	Developer	Senior	Costa Rica	21	31	30	38
54	Developer	Senior	Chile	20	32	41	27
55	Developer	Senior	Chile	24	34	33	29
56	Developer	Senior	Argentina	20	31	41	28
57	Developer	Junior	Chile	30	28	31	31
58	Developer	Senior	Argentina	27	29	37	27
59	Developer	Senior	Spain	21	26	37	36
60	Developer	Senior	Chile	28	27	28	37
61	Developer	Senior	Chile	25	32	32	31
62	Developer	Mid-Senior	Costa Rica	18	33	41	28
63	Developer	Senior	Argentina	28	37	30	25
64	Developer	Senior	Costa Rica	17	37	34	32
65	Developer	Junior	Chile	22	31	37	30
66	Project Manager	Senior	Chile	23	40	30	27
67	Project Manager	Junior	Argentina	20	29	35	36
68	Project Manager	Mid-Senior	Chile	25	34	31	30
69	Project Manager	Senior	Chile	21	30	36	33
70	Project Manager	Senior	Chile	26	44	27	23
71	Project Manager	Senior	Chile	20	36	40	24
72	Project Manager	Senior	Chile	24	40	27	29
73	Project Manager	Senior	Argentina	21	34	35	30
74	Project Manager	Senior	Chile	26	25	40	29
75	Project Manager	Senior	Argentina	22	30	36	32
76	Project Manager	Senior	Chile	27	33	33	27
77	Project Manager	Senior	Chile	22	32	40	26
78	Project Manager	Senior	Argentina	25	21	34	40
79	Project Manager	Senior	Costa Rica	22	38	26	34
80	Project Manager	Senior	Chile	19	38	39	24
81	Project Manager	Senior	Argentina	31	19	25	45
82	Project Manager	Senior	Costa Rica	32	26	37	25
83	Project Manager	Senior	Chile	22	21	33	44
84	Project Manager	Mid-Senior	Chile	23	35	28	34
85	Project Manager	Senior	Chile	21	33	37	29
86	Project Manager	Senior	Costa Rica	36	34	27	23
87	Project Manager	Senior	Spain	20	30	39	31
88	Project Manager	Senior	Chile	21	23	33	43
89	Project Manager	Senior	Chile	31	27	29	33
90	Project Manager	Senior	Argentina	34	19	37	30
91	Project Manager	Senior	Costa Rica	24	34	40	22
92	Project Manager	Senior	Chile	26	25	37	32
93	Quality Assurance	Senior	Spain	31	35	16	38
94	Quality Assurance	Senior	Costa Rica	19	31	45	25
95	Software Architect	Senior	Chile	28	25	36	31
96	Software Architect	Senior	Colombia	33	27	23	37
97	Software Architect	Senior	Spain	32	20	27	41
98	Software Architect	Senior	Chile	29	27	28	36
99	Software Architect	Senior	Chile	19	31	38	32
100	Software Architect	Senior	Argentina	20	31	41	28
101	Software Architect	Senior	Spain	32	26	36	26
102	Software Architect	Mid-Senior	Colombia	25	28	25	42
103	Software Architect	Mid-Senior	Costa Rica	21	41	36	22
104	Software Architect	Mid-Senior	Chile	22	23	35	40
105	Software Architect	Mid-Senior	Argentina	23	28	32	37
106	Software Architect	Mid-Senior	Colombia	15	30	38	37
107	Software Architect	Mid-Senior	Spain	24	25	32	39
108	Software Architect	Mid-Senior	Chile	22	28	33	37
109	Software Architect	Senior	Colombia	24	19	39	38

110	Software Architect	Senior	Argentina	24	34	37	25
111	Software Architect	Senior	Spain	26	21	32	41
112	Software Architect	Senior	Colombia	38	18	37	27

Appendix 2 - Focus Group partial transcriptions

9.1 Participants

- Senior Consultant 1 (SC1)
- Senior Consultant 2 (SC2)
- Instructional Designer 1
- Instructional Designer 2 (ID2)
- Technical Leader 1 (TL1)
- Technical Leader 2 (TL2)

9.2 Question 1

Are the identified learning styles representative of your experience with software development teams?

- SC1: “Absolutely. From what we’ve seen, many senior developers and architects tend to be highly analytical. They like everything to have clear data and solutions to be based on pure logic.”
- ID1: “I agree. In graduate programs, we notice that students who already have programming experience excel in activities that require abstract thinking, like when they design algorithms.”
- TL2: “Yes, it’s the most common pattern, but it’s not always the case. In QA or roles with more collaboration, we see more variety. But for typical developers, the profile is very well defined.”

9.3 Question 2

How practical are the proposed strategies in corporate training environments?

- SC2: “We already use problem-based learning, but in a more informal way. What we find great is the idea of integrating it with data analysis - that could be very useful for technical training.”
- ID2: “We love the structured discussion approach. Many engineers shy away from open debates, but when they have clear objectives and guidance, they get more involved.”
- TL1: “The challenge is adapting it for virtual courses. I would add more interactive tools, like simulators where they can program in real time.”

9.4 Question 3

What adjustments would you recommend to improve their implementation?

- SC1: “I would include pair programming dynamics so that ’Thinking’ profiles work as a team. I’ve seen that several programmers learn better when they’re accompanied and see tangible results.”
- ID1: “The materials should be more modular. For example, specific cases for back-end versus front-end, because their needs are different.”
- TL2: “We need to train instructors to know how to handle analytical profiles. I believe the strategies go hand in hand with an instructor who can apply them appropriately.”

9.5 Final Comments

Any final comments before we wrap up?

- SC2: “I agree that the “Thinking” style is well identified, but it would be interesting to expand the study to non-technical roles, like Product Owners, to see the differences.”
- ID2: “The strategies are solid, but they need flexibility. A single team can have a mix of learning styles.”
- TL1: “My recommendation is to prioritize adaptation for hybrid environments. That’s where the future of training is heading.”