Learning the Domain by Doing: Evidence from Developer Onboarding in Brazil

Rafaela Otemaier
Graduate Program in Computer Science (PPGIa), Pontifícia Universidade Católica do Paraná (PUCPR)
Curitiba, Brazil
kelly.rafaela@pucpr.br

Sheila Reinehr
Graduate Program in Computer Science (PPGIa), Pontifícia Universidade Católica do Paraná (PUCPR)
Curitiba, Brazil
sheila.reinehr@pucpr.br

Andreia Malucelli
Graduate Program in Computer Science (PPGIa), Pontifícia Universidade Católica do Paraná (PUCPR)
Curitiba, Brazil
andreia.malucelli@pucpr.br

Abstract

Context: Business domain knowledge is essential for software developers to understand not only how to change a system, but also why it behaves as it does. During onboarding, however, developers are often expected to become productive before fully understanding the domain in which the software operates. **Objective:** This study investigates which strategies software developers report using to acquire business domain knowledge during onboarding and how the reported use of these strategies varies across individual and organizational contexts. **Method:** We conducted a survey with 257 software developers in Brazil who joined projects with little or no prior familiarity with the business domain, analyzing the reported frequency of use of different domain learning strategies. **Results:** Reported domain learning is primarily centered on strategies associated with Experiential Learning and Social Learning. Task execution, interaction with colleagues, and reading code or architecture artifacts were the most frequently reported strategies, whereas formal training, internal documentation, customer interaction, and AI-based assistance were reported less consistently. Reported strategy use also varies across contexts: less experienced developers report greater reliance on guided support and AI-based assistance, while organizational factors such as company size and work arrangement are associated with different levels of access to structured resources, observation, and stakeholder interaction. **Conclusion:** Domain onboarding should not be treated as a uniform experience, as developers' reported reliance on learning strategies depends on both individual maturity and the learning opportunities available in their organizational context. These findings show that domain onboarding is a context-sensitive dependency process: developers often act before fully understanding the domain, and their access to business rationale depends on how work contexts expose domain reasoning through tasks, artifacts, peers, and stakeholders.

Keywords

Domain Knowledge, Onboarding, Developer, Software Engineer

1 Introduction

Software systems encode assumptions, rules, constraints, and decisions that emerge from specific business domains. As software development increasingly involves complex systems [41] and distributed teams [13], developers are expected not only to implement technically correct solutions, but also to understand the business logic that makes those solutions meaningful [2, 10]. Domain knowledge is therefore essential for aligning technical decisions with stakeholder needs, reducing ambiguity, and ensuring that software systems fulfill their intended purpose [2]. This requirement becomes particularly demanding during onboarding, when developers must quickly build domain understanding while already being expected to contribute.

When developers join a new project, they often need to act before they fully understand the system, the team, and the business domain in which the software operates. This initial period requires newcomers to simultaneously learn technical artifacts, development practices, organizational routines, and domain-specific concepts, frequently with limited contextual information and strong dependence on tacit knowledge shared by more experienced developers [32, 40]. Onboarding is therefore not merely a transition into a team; it is also a process of reconstructing the context needed to make informed development decisions.

Although onboarding has been widely studied in Software Engineering (SE), most prior work has emphasized team socialization, productivity, task allocation, and technical knowledge transfer [32, 35, 40]. These studies show that newcomers learn through tasks, interaction, mentoring, and access to project artifacts. However, business domain knowledge often remains embedded within these broader onboarding concerns and is rarely examined as a central phenomenon. As a result, less is known about the repertoire of strategies developers rely on when the learning target is business domain knowledge, particularly when they start with little or no prior familiarity with the domain [2].

This gap matters because domain learning is not simply another onboarding activity. Developers may learn how to modify a system without fully understanding why certain rules, exceptions, or architectural decisions exist. In such cases, limited domain understanding can delay time-to-productivity, increase dependence on experienced team members, and contribute to requirements misunderstandings, misaligned design decisions, and costly rework. Understanding how developers acquire business domain knowledge is therefore important for improving onboarding programs and reducing the risk of technically correct solutions that are poorly aligned with business needs.

To address this gap, this study investigates the strategies through which software developers acquire business domain knowledge during onboarding, focusing specifically on individuals who entered a project with little or no prior familiarity with the domain. The study is guided by the following research questions:

- **RQ1.1:** Which strategies do software developers report using to acquire business domain knowledge during onboarding?
- **RQ1.2:** How does reported use of domain knowledge acquisition strategies vary across individual and contextual factors during onboarding?

We conducted a survey with 257 software developers in Brazil who reported low or no prior knowledge of the business domain when joining a project. The study examines the relative use of different domain learning strategies, including task execution, interaction with colleagues, mentor support, code and architecture reading, documentation, formal training, observation, client interaction, and the use of AI tools. It also analyzes how reported strategy use varies according to individual and contextual factors, such as professional experience, company size, work arrangement, and gender.

This study contributes to SE research by showing that business domain onboarding is not only a matter of exposing newcomers to tasks, people, or artifacts, but of whether these resources make business rationale visible during real work. The findings reveal a context-sensitive dependency pattern: when prior domain familiarity is low, developers rely mainly on real tasks and peers, while access to structured resources, observation, stakeholders, and AI-supported assistance varies with experience, company size, and work arrangement. For practitioners, the results indicate that onboarding should connect task execution to explicit domain reasoning, rather than assuming that productivity implies domain understanding.

The remainder of this paper is organized as follows. Section 2 presents the theoretical background. Section 3 discusses related work. Section 4 describes the study design. Section 5 presents the results. Section 6 discusses the findings and their implications. Section 7 addresses threats to validity. Finally, Section 8 concludes the paper.

2 Background

Onboarding in SE.. Research on onboarding in SE shows that newcomers must adapt to technical, social, and organizational dimensions of a software project [36]. During this process, developers must become familiar with the codebase, development practices, team routines, communication channels, and contextual assumptions that guide project work. Because much of this knowledge is distributed across people, artifacts, and organizational routines, onboarding often depends on both formal support and informal interaction [26, 37]. In this study, onboarding is treated as the broader adaptation process within which business domain knowledge acquisition occurs.

Domain Knowledge in Software Development. Domain knowledge refers to the specialized understanding of the business area, operational rules, constraints, terminology, stakeholders, and goals that give meaning to a software system [2]. In software development, this knowledge helps developers interpret requirements, understand why specific features or architectural decisions exist, and align technical work with organizational objectives. Unlike purely technical knowledge, domain knowledge is often tacit, contextual, and embedded in business processes, stakeholder practices, and previous project decisions [1, 18, 34]. For this reason, acquiring domain knowledge usually requires more than reading documentation; it involves participation in work activities, interaction with knowledgeable actors, and interpretation of project artifacts. Given these characteristics, acquiring domain knowledge requires more than access to explicit information; it involves active engagement in work practices and interaction with knowledgeable actors.

Learning Mechanisms and Strategies in the Work Context. Domain knowledge acquisition during onboarding can be conceptualized as a work-based learning process. Previous research describes several learning mechanisms that can be observed through strategies developers use in daily work [22]. In this study, these mechanisms are organized into four categories:

- **Experiential Learning:** Learning through direct engagement with work activities and technical artifacts. It includes strategies such as task execution, problem-solving, code exploration, system analysis, and engagement with artifacts such as user stories, requirements, and tickets [12, 19, 30].
- **Social Learning:** This mechanism refers to learning mediated by interpersonal interaction. It includes mentorship, informal collaboration with colleagues, participation in team routines, and observation of experienced developers [8, 9, 22].
- **Structured Learning:** This mechanism refers to organized forms of knowledge support provided by the organization. It includes formal training, internal documentation, wikis, and structured onboarding paths [3, 4, 7].
- **Self-directed Learning:** This mechanism refers to proactive information seeking by developers. It includes consulting online communities, external technical materials, and using artificial intelligence tools to retrieve explanations or contextualized information [8, 24, 33].

Individual and Contextual Factors. The use of learning strategies is not uniform across developers or organizations. Individual characteristics, such as professional experience, seniority, generation, and gender, may influence how developers seek support, interpret information, and balance guided versus autonomous learning [22, 23, 29, 38]. Contextual characteristics also shape the availability and use of strategies. For instance, company size may influence the presence of formal training and documentation, while remote, hybrid, and on-site work arrangements affect access to colleagues, stakeholders, and opportunities for informal observation [7, 9, 20, 27]. These factors do not determine behavior, but influence the likelihood of adopting specific learning strategies.

Conceptual Framing of the Study. Taken together, these concepts frame domain knowledge acquisition during onboarding as a situated, work-based learning process. The learning mechanisms described above provide the conceptual basis for operationalizing domain learning strategies, while individual and contextual factors support the analysis of variation in their use. This framing enables a structured investigation of how developers acquire domain knowledge in practice and how this process varies across different conditions.

3 Related Work

Prior research on onboarding in SE has examined how newcomers become productive, integrated, and able to contribute to development teams. This literature has emphasized task allocation, team socialization, mentoring, access to project artifacts, and technical knowledge transfer [17]. Although these studies recognize that newcomers must understand the system and its context, business domain knowledge is often treated as part of a broader adaptation process rather than as a specific learning outcome. Consequently, existing evidence explains several mechanisms through which onboarding occurs, but provides limited insight into how developers construct domain understanding when they enter a project with little or no prior familiarity with the business domain.

Focusing on task-based learning, Ju et al. [22] examine onboarding through the lens of engineering tasks and show that learning is strongly shaped by task execution. Activities such as bug fixing and feature implementation help newcomers acquire project knowledge and build confidence, while strategies such as "Simple-Complex" and "Exploration-Based" adjust task complexity according to developer experience. This perspective is particularly relevant because it frames tasks as units of learning during onboarding. However, the study does not distinguish which types of knowledge are produced through task execution, making it unclear how such tasks contribute specifically to the acquisition of business domain concepts when such knowledge is initially absent.

From a social perspective, Gregory et al. [17] analyze onboarding in agile teams and identify understanding the product and its domain as an important goal for newcomers. Their findings show that practices such as pair programming, stand-ups, and retrospectives support learning through interaction, feedback, and continuous alignment. This perspective highlights the social nature of onboarding, but it also leaves open an important question: whether interaction within the development team provides sufficient access to the business reasoning behind development decisions. In addition, the study is based on a single case, which limits evidence on how domain learning occurs across different organizational contexts.

Addressing knowledge transfer, Viana et al. [40] emphasize the role of tacit knowledge and informal interaction between senior and novice engineers. Their findings indicate that newcomers learn through observation, guidance, and participation in tasks, while formal documentation is often incomplete or outdated. This reinforces the idea that critical project knowledge is not always explicitly documented. However, the study is based on a small-scale qualitative setting and does not examine how different learning strategies are used across a broader population or how they relate specifically to business domain knowledge acquisition.

Considering the Brazilian context, Moraes et al. [29] investigate onboarding from the perspective of interns and identify mentoring, social interaction, and documentation as relevant resources for learning and integration. While consistent with prior findings, the focus on early-career participants limits the understanding of how developers with different levels of experience acquire business domain knowledge. This distinction is particularly relevant because limited domain familiarity is not restricted to novices, but also affects experienced developers transitioning across projects, organizations, or business sectors.

Prior studies explain onboarding as a process shaped by tasks, collaboration, tacit knowledge transfer, and contextual support. However, the remaining gap is not only identifying which onboarding resources newcomers use, but understanding how these resources support access to business rationale when developers must contribute before they fully understand the domain. This distinction matters because developers may learn to modify a system without fully understanding the business rationale behind such changes.

This study addresses this gap by focusing on business domain knowledge acquisition during onboarding. We analyze the relative use of multiple learning strategies and examine how their use varies across individual and organizational contexts in Brazil, providing a more explicit account of how developers construct domain understanding in practice.

4 Study Design

This study followed a structured survey process comprising scoping, planning, operation, and analysis [28]. We position this work as a descriptive empirical study, focusing on reported reliance on learning strategies rather than on cognitive processes or strategy effectiveness. A survey design was appropriate because it enables the collection of self-reported data from a broad population, allowing us to characterize which strategies developers report using to acquire business domain knowledge during onboarding and how this reported use varies across individual and contextual factors [25].

Scoping and Planning. The study was structured according to the Goal–Question–Metric framework [39]. The research goal was to investigate how developers with limited prior familiarity acquire business domain knowledge during onboarding. This goal was refined into two research questions: RQ1.1, which focuses on identifying domain knowledge acquisition strategies, and RQ1.2, which examines how individual and contextual factors influence the use of these strategies. The corresponding metrics captured reported strategy use and participant and organizational characteristics. The survey instrument was designed following established guidelines for SE survey research [16, 28].

The questionnaire comprised three sections: participant characteristics, contextual information (e.g., company size, sector, and work arrangement), and domain knowledge acquisition strategies. The latter was presented only to participants reporting no or low prior domain knowledge and included items derived from the learning mechanisms discussed in Section 2, covering experiential, social, structured, and self-directed forms of learning.

Each strategy was operationalized as an individual item and rated on a 0–10 scale, where 0 indicated no use and 10 indicated very frequent use. For example: "To what extent did you learn the business domain by executing tasks in the project?" Strategy-related variables were treated as ordinal measures and analyzed independently, without aggregation into composite indices, as the goal was to compare the relative use of specific strategies rather than estimate latent constructs. The 0–10 scale enabled fine-grained comparison across strategies.

To improve the quality of the instrument, a pilot study was conducted with three experienced software professionals to evaluate clarity, completeness, and response time. Based on their feedback,

the wording of items and instructions was refined. In addition, the research team conducted an internal review of the instrument, discussing item formulation, alignment with the research questions, and potential sources of ambiguity. This iterative process led to the final version of the questionnaire. The final instrument and anonymized dataset are available.

Operation (Data Collection). The survey was implemented as an online questionnaire. We adopted a non-probabilistic sampling strategy, in which the sampling frame consisted of software developers reachable through professional networks (LinkedIn) and professional groups (WhatsApp) [14], strategy to reach active practitioners in the software industry [11].

A total of 512 valid responses were collected. For the analyses presented in this study, we considered a subset of 257 participants who reported having either "none" or "low" prior business domain knowledge at the onset of onboarding. This inclusion criterion was used to focus the analysis on developers who had to acquire domain understanding from a baseline of limited familiarity, regardless of their technical seniority.

Because the study used non-probabilistic sampling, the sample should not be interpreted as statistically representative of the Brazilian developer population. Yamane's equation for sample size calculation [43] was used only as a contextual reference for the scale of the collected dataset, considering an estimated population of 600,000 to 800,000 software developers in Brazil[1]. The analyzed subset of 257 participants is suitable for the exploratory, non-parametric comparisons performed in this study, but the findings should be interpreted in light of the sampling strategy.

All survey questions were mandatory, except for an optional open-ended field. The study was conducted in accordance with ethical guidelines and was approved by the Research Ethics Committee of University X, CAAE XXXXXXXX.X.XXXX.XXXX. All participants provided informed consent prior to participation.

Analysis and Interpretation. The analysis followed a three-step procedure. First, the dataset was cleaned and filtered. Only participants who reported "none" or "low" prior domain knowledge were included, while participants with "medium" to "very high" prior knowledge were excluded. This filtering strategy ensured alignment between the research goal and the analyzed sample.

Second, statistical analyses were conducted. Normality was assessed using the Shapiro–Wilk test. Because the strategy variables did not follow a normal distribution and were treated as ordinal measures, we used medians and interquartile ranges to summarize the data. Differences across strategies were analyzed using the Friedman test, while differences across independent groups were analyzed using the Kruskal–Wallis test. The analyses were conducted in SPSS and R.

Third, the interpretation of the results was reviewed by the authors. The initial analysis was performed by the first author and subsequently examined by the remaining authors. Disagreements were discussed until consensus was reached, supporting the reliability of the interpretations.

[1]https://www.jetbrains.com/lp/devecosystem-2023/

5 Results

The data analyzed in this study are part of a broader survey on onboarding practices in the Brazilian SE context, conducted between March 18 and April 18, 2026. From an initial pool of 655 responses, 512 remained after removing incomplete entries. The analysis focuses on a targeted subset of 257 participants (50.2% of the valid sample) who met the inclusion criterion of having no or low prior business domain knowledge at the onset of onboarding.

Within this subset, 77 respondents (30.0%) reported no prior knowledge, while 180 (70.0%) indicated low levels of familiarity. Participants with moderate to very high levels of prior domain knowledge were excluded to isolate the process of acquiring new domain expertise from scratch, ensuring the findings reflect the learning journey of those without pre-existing domain context.

Geographic Distribution. While respondents represent all Brazilian regions (Figure 1), the sample is predominantly concentrated in the South (61.5%) and Southeast (15.6%). This distribution primarily reflects the reach of the professional networks used for recruitment and should be considered when interpreting the findings within the broader national context.

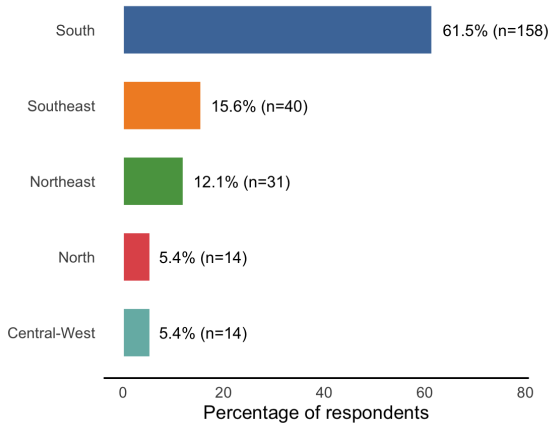

Figure 1: Geographic distribution of respondents across Brazilian regions.

Individual Characteristics. The sample reflects the contemporary SE landscape in Brazil, with a predominantly male presence (85.2%, Figure 2) and a diverse range of professional experience (Figure 3).

Although prior business domain knowledge was low by design, technical seniority was not: 32.3% of the participants had more than four years of experience in software development (Figure 3). This contrast highlights that even experienced developers face the challenge of rebuilding domain understanding when transitioning between projects or industries.

Organizational Context. Participants are primarily embedded in large organizations (70.8%), with a strong concentration in the Technology and Software (50.6%) and Finance (20.6%) sectors (Figure 4). The remaining participants are distributed across sectors such as Manufacturing (7.8%), Retail (5.8%), and Healthcare (5.1%), among

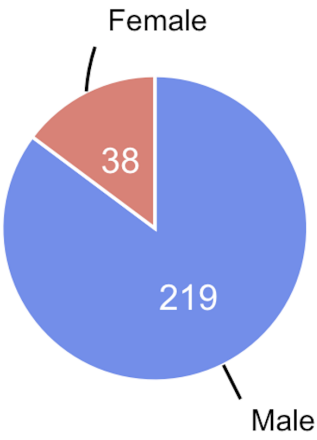

Figure 2: Distribution of respondents by gender.

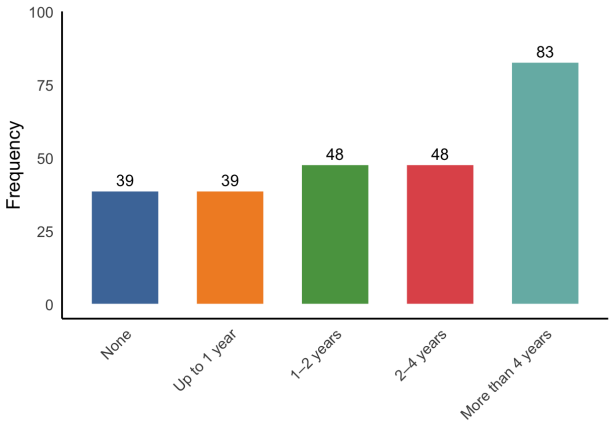

Figure 3: Distribution of respondents by years of experience in software development.

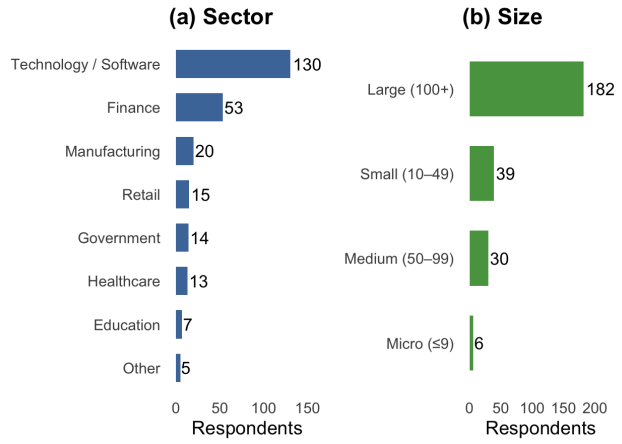

Figure 4: Distribution of respondents by (a) industry sector and (b) company size.

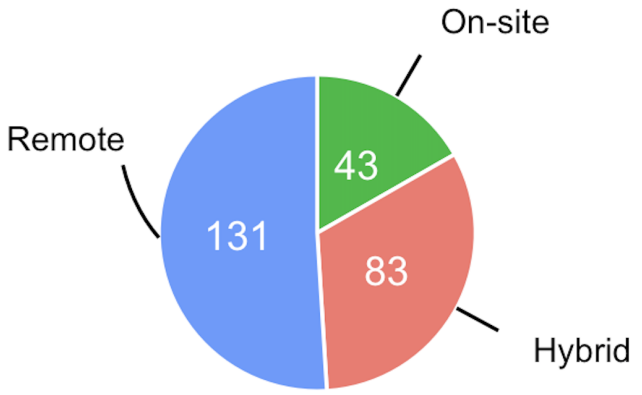

Figure 5: Distribution of respondents by work arrangement (remote, hybrid, and on-site).

> **Sample Finding**. More than half of the valid sample (50.2%) joined their projects with limited business domain familiarity: 70.0% of this subgroup reported low familiarity, while 30.0% reported no prior domain knowledge. This shows that limited domain familiarity at the start of onboarding is a common condition among the surveyed developers.

others. This profile indicates that the findings are grounded in corporate environments where domain knowledge is often specialized and tightly coupled with complex business processes.

Work Arrangement. The work context is predominantly distributed. As shown in Figure 5, remote work accounts for 51.0% of the sample, followed by hybrid arrangements (32.3%), while on-site work represents 16.7%. Together, remote and hybrid settings account for 83.3% of the cases. This environment is relevant for interpreting the results, as traditional opportunities for informal, co-located learning are reduced and replaced by mediated interactions and digital artifacts.

5.1 RQ1.1: Domain Knowledge Acquisition Strategies

The acquisition of business domain knowledge during onboarding is predominantly driven by experiential and social learning strategies. Rather than following formal or standardized learning paths, developers navigate the domain through direct engagement with project tasks and continuous interaction with their

peers. This pattern is statistically supported by the Friedman test ($\chi^2(12) = 683.984, p < 0.001$), indicating that the use of the investigated strategies differs significantly.

The Dominance of Practice and Interaction. As shown in Figure 6, a small set of strategies emerges as the primary means of domain learning. "Task execution in the project" stands out as the most utilized approach, achieving the maximum median score (10). This pattern suggests that developers often rely on task execution as an entry point for engaging with domain-related information, rather than depending exclusively on prior domain preparation before starting project work. Following closely, "Interaction with colleagues" (median = 8.5) and "Code and architecture reading" (median = 8) reinforce that knowledge is extracted from both social exchange and the technical artifacts that embody business rules.

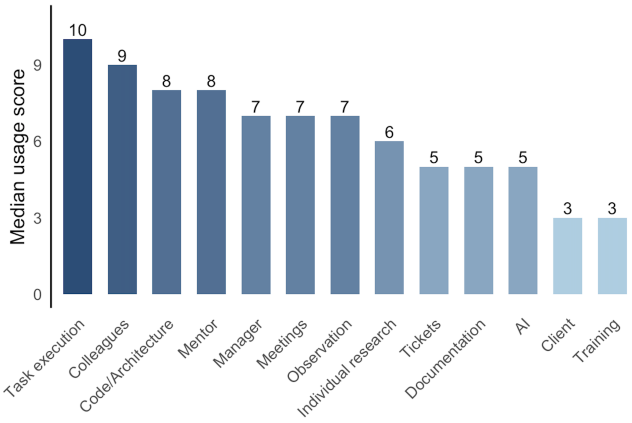

Figure 6: Median usage scores (0–10 scale) of domain knowledge acquisition strategies, ordered from least to most frequently used.

Secondary and Context-Dependent Strategies. A second tier of strategies includes support from mentors and managers, participation in meetings, and the observation of peers, all with median scores between 7 and 8. While still frequent, these strategies show greater dispersion in their adoption (Figure 7). This variability suggests that while social support is important, its availability and intensity are not uniform across the sample.

In contrast, strategies often associated with formal onboarding-such as "Internal documentation" (median = 5) and "Formal training" (median = 2), show remarkably lower usage. Even "AI tools" (median = 5), despite their current prominence in SE, do not appear as a primary source for domain-specific context in this dataset. The least utilized strategy, "Interaction with clients" (median = 3), suggests limited direct interaction between developers and end-users during the onboarding phase.

Consistency vs. Variability. The distribution analysis in Figure 7 reveals three distinct adoption patterns. "Task execution" is the only strategy with consistently high usage across the entire sample, representing a baseline across participants. In contrast, collaborative strategies (mentorship, meetings, peer observation) show high medians but wide interquartile ranges, indicating that their role in onboarding is highly situated. Finally, formal and externally-oriented mechanisms (training, documentation, client contact) are less frequent and also unevenly distributed, suggesting they are used only in specific organizational contexts.

> **Key Finding RQ1.1**. Domain learning during onboarding is primarily driven by practical and social strategies, while formal resources and AI-based assistance are used less consistently.

5.2 RQ1.2: Influencing Factors on Acquisition Strategies

The acquisition of domain knowledge is not a uniform process but is significantly shaped by the interplay between individual characteristics and organizational contexts. The Kruskal–Wallis test revealed several statistically significant differences across these factors, as summarized in Table 1. These results indicate that the choice and intensity of learning strategies are highly sensitive to the developer's background and the environment in which they are embedded.

The Kruskal–Wallis test revealed several statistically significant differences across these factors. Table 1 presents the corresponding p-values together with descriptive trends derived from the observed group medians. These trends summarize the direction of the median differences across groups and should not be interpreted as direct outputs of the statistical test.

Table 1: Summary of statistically significant differences in domain learning strategies across factors

Factor	Strategy	p	Trend
Experience	Mentor Support	0.034	Higher among less experienced developers
	Task Execution	0.001	Decreases with experience
	Use of AI Tools	<0.001	Decreases with experience
	Observing Others	0.007	Peaks at intermediate experience levels
Company Size	Formal Training	<0.001	Higher in medium and large organizations
	Observing Others	0.001	Higher in medium and large organizations
	Internal Documentation	0.045	Higher in medium and large organizations
Work Arrangement	Formal Training	0.017	Higher in hybrid settings
	Interaction with Clients	0.023	Higher in on-site arrangements
	Observing Others	0.037	Higher in hybrid settings
Gender	Mentor Support	0.038	Higher among female participants
	Observing Others	0.041	Higher among female participants

Professional Maturity and Learning Autonomy. Professional experience is the factor associated with the most extensive variations in strategy adoption. As developers progress in their careers, there is a marked shift in how they engage with the domain. Less experienced developers exhibit a higher reliance on "Mentor support"

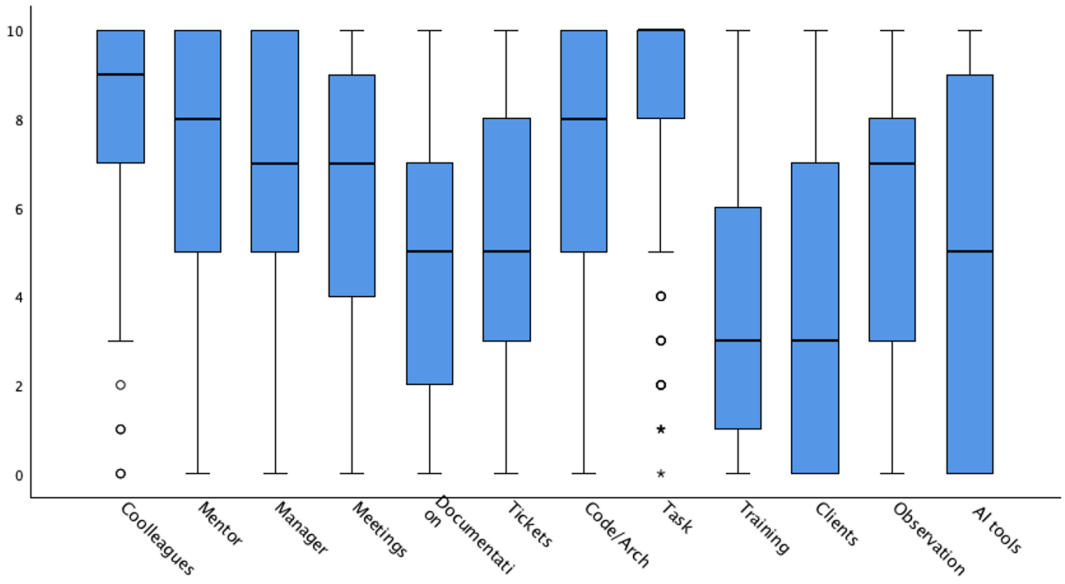

Figure 7: Distribution of usage scores (0–10 scale) for domain knowledge acquisition strategies. Strategies are ordered by median usage score.

and "Use of AI tools" ($p < 0.001$), suggesting a need for external guidance to navigate initial uncertainty. In contrast, technical seniority appears to foster greater autonomy, with a significant reduction in the use of these support mechanisms. Interestingly, "Observing other developers" follows a non-linear trajectory, peaking at intermediate experience levels (Figure 8). This suggests a transitional phase where developers are socially integrated enough to learn through observation but still lack the full autonomy of senior roles.

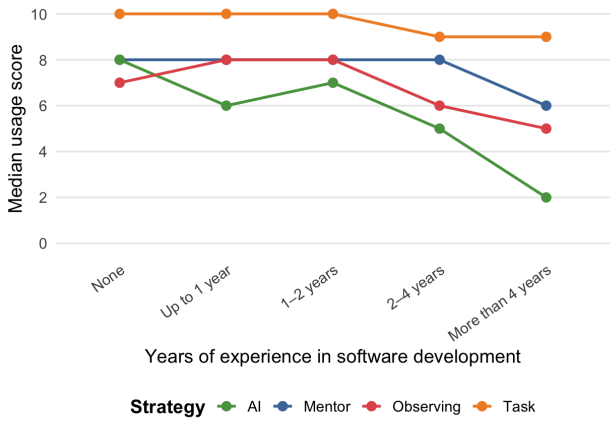

Figure 8: Median usage of domain knowledge acquisition strategies across experience levels (only statistically significant strategies are shown).

Organizational Scale and Structural Support. The scale of the organization is associated with the reported availability and use of structured learning resources. Medium and large organizations provide an environment significantly more "structure-heavy," with higher utilization of "Formal training," "Internal documentation," and "Peer observation" compared to smaller firms ($p \leq 0.001$). This indicates that the acquisition of domain knowledge in large-scale corporate settings is mediated by established knowledge management processes and formal onboarding programs that are often absent or less mature in micro and small enterprises.

Work Arrangement and Social Dynamics. The physical and digital environment in which onboarding occurs shapes the nature of social and client interactions. "Interaction with clients" is significantly more prevalent in on-site settings ($p = 0.023$), highlighting that physical presence remains a facilitator for direct business exposure. Conversely, hybrid arrangements appear to offer a unique balance, showing higher levels of "Formal training" and "Peer observation" compared to strictly remote or on-site models. This suggests that the hybrid context may encourage a more deliberate use of both formal and social learning mechanisms. Finally, gender differences were identified in "Mentor support" and "Observing others," both more prevalent among female participants ($p < 0.05$), pointing toward potential variations in social learning preferences.

Key Finding RQ1.2. Domain learning strategies vary across onboarding contexts. Less experienced developers rely more on guided and AI-supported learning, while organizational context shapes access to domain-learning opportunities.

6 Discussion

Our survey captures developers' reported reliance on learning strategies, rather than the cognitive processes involved in domain knowledge construction or the effectiveness of each strategy. This distinction is important when interpreting the findings. The results suggest that domain onboarding is less a matter of choosing among strategies and more a matter of accessing business rationale in practice. Developers rely on the learning resources made available by their career stage, team routines, organizational scale, and work arrangement.

The results indicate that business domain knowledge is not typically acquired as a prerequisite for work, but through work itself. Developers begin contributing while still lacking a complete understanding of the domain, progressively constructing this understanding through tasks, interactions, and artifacts. While prior onboarding research has emphasized task execution as a mechanism for becoming productive [22, 40], our findings show that, when the learning target is business domain knowledge, these same mechanisms also serve as the primary means for inferring business rules, assumptions, and rationale embedded in the system.

This reveals a tension in how onboarding is commonly structured. Formal mechanisms such as training and documentation are often designed to reduce uncertainty [3, 7], yet they were not reported as primary sources of domain learning in our data. This does not imply that such mechanisms are irrelevant, but rather that they are insufficient to convey the contextual and tacit dimensions of domain knowledge, particularly the reasoning behind business rules and decisions. This interpretation aligns with prior work on tacit knowledge transfer [34, 40], reinforcing that domain understanding depends on situated engagement rather than on explicit information alone.

The centrality of task execution also complicates a common assumption in onboarding: that newcomers can be prepared first and then safely exposed to real work. Our results suggest the opposite. Onboarding into the business domain occurs under conditions of partial understanding, where developers must act before they fully know how to interpret the domain. This finding reinforces Ju et al.'s view of tasks as important units of onboarding [22], but it also introduces a distinction that prior work has not fully explored. Tasks may help developers become productive, but productivity does not necessarily imply domain understanding. A developer may learn where to change the code, which component to modify, or how to satisfy an issue description without fully understanding the business rationale behind the requested change. This distinction is important because domain misunderstanding can remain hidden when technical output appears correct.

Social interaction emerged as another key component of domain learning, consistent with prior studies on collaboration, mentoring, and feedback in onboarding [8, 17, 40]. However, our results show that access to social learning is uneven. Mentorship, meetings, and observation depend on team availability, work arrangements, and local practices. As a result, onboarding should be understood not only as an organizational process, but as a situated experience shaped by the immediate team environment. Developers within the same organization may have substantially different opportunities to access domain knowledge depending on the availability of experienced colleagues and exposure to decision-making contexts.

The main analytical contribution of this study lies not only in identifying which strategies are most frequently reported, but in showing that reliance on these strategies varies across individual and organizational conditions. This suggests that domain onboarding should not be treated as a uniform experience: developers' access to domain-related resources appears to depend on career stage, organizational scale, and work arrangement.

This uneven access to support is particularly evident across experience levels. Less experienced developers reported greater reliance on mentors and AI tools, suggesting that uncertainty increases the need for guided and on-demand support, consistent with prior research on onboarding [29, 38]. However, our findings also indicate that support needs are not static. As developers accumulate experience, the usefulness of different strategies changes. Observation, for instance, may be limited at early stages due to lack of context, but becomes more meaningful as developers develop a preliminary understanding of the system and domain. This suggests that onboarding strategies should be sequenced rather than offered uniformly.

The results also highlight the role of work arrangement in shaping access to domain knowledge. Remote and hybrid settings can support flexibility but may reduce incidental learning and direct exposure to stakeholders, consistent with prior research on distributed onboarding [9, 20, 31]. Our findings extend this view by showing that the impact is not only social, but also epistemic. With fewer opportunities to interact with clients or domain experts, developers rely more on mediated representations such as tickets, documentation, code, and peer explanations. While useful, these sources can filter or simplify the underlying business reasoning, increasing the risk that developers learn how the system works without fully understanding why.

The use of AI tools introduces another tension. AI appears to function as an accessible first line of support, especially when colleagues or mentors are not immediately available, reducing friction during task execution and helping developers move forward [24, 33]. This is particularly relevant for less experienced developers. However, our findings suggest that AI-supported learning should be interpreted cautiously. While AI can support operational understanding, such as terminology, code behavior, or generic business concepts, it does not reliably provide access to organization-specific business rationale, including historical decisions, stakeholder priorities, or tacit exceptions, unless these are explicitly represented in accessible artifacts. As a result, AI may accelerate action without necessarily deepening domain understanding. This distinction indicates that faster access to explanations does not automatically translate into more effective domain learning.

The findings challenge a purely structured view of onboarding. Business domain learning does not appear to function primarily as a linear transfer of knowledge from organization to newcomer.

Instead, it unfolds as a situated and uneven process in which developers construct understanding through real tasks, social mediation, technical artifacts, and available support. This interpretation helps explain why increasing the amount of training or documentation alone may not solve the problem. The core issue is not only whether information exists, but whether developers can connect that information to concrete work, stakeholder reasoning, and the history of decisions embedded in the system.

For practice, these results suggest that onboarding should be centered on guided participation in real work rather than relying primarily on formal mechanisms. Organizations should connect technical tasks to business explanations, for example through domain-aware mentorship, explicit rationale in tickets and reviews, and exposure to stakeholder discussions. The goal is not to eliminate uncertainty, but to support developers while they make sense of the domain through work. To translate these findings into practice, Table 2 maps each observed dependency pattern to an onboarding risk and a corresponding intervention. Rather than prescribing a single onboarding model, it links observed patterns of strategy use to risks that emerge when developers must learn domain rationale through tasks, artifacts, and social interaction under different contextual conditions.

Table 2: Context-sensitive domain onboarding guide grounded in RQ1.1 and RQ1.2

Context (Evidence)	Onboarding Risk	Operational Implication
Less experienced developers	Higher dependence on external guidance to interpret domain concepts	Pair initial tasks with domain-aware mentors [8, 22] and make business rationale explicit in tickets and code reviews to increase the visibility of domain knowledge during onboarding.
More experienced developers	Domain knowledge may remain implicit when learning relies primarily on autonomous task execution	Introduce domain understanding checkpoints beyond technical delivery and encourage explicit knowledge sharing practices [40].
Large organizations	Structured resources may be disconnected from real task execution (RQ1.1: low reliance on training/documentation)	Link training and documentation to real cases, historical decisions, and business exceptions [21]
Small organizations	Lack of structured support increases reliance on informal and uneven learning processes	Introduce lightweight domain artifacts (e.g., glossary, stakeholder map, critical rules) to stabilize access to domain knowledge [15]
Remote work	Reduced access to mentoring and rich communication opportunities may limit exposure to domain knowledge	Schedule explicit domain sessions and document decision rationale in shared channels [6]
Hybrid / on-site work	Learning opportunities exist but may remain implicit or unevenly leveraged	Use shadowing, stakeholder meetings, and business process walkthroughs to make domain learning explicit [5]

For research, this study establishes a descriptive understanding of how developers rely on different strategies to acquire domain knowledge during onboarding. The next step is to move beyond usage and investigate what these strategies actually produce in terms of learning outcomes.

Future work should examine how different strategies contribute to distinct forms of domain understanding, and under which conditions they are more or less effective. This includes not only evaluating individual strategies, but also how they are combined and sequenced throughout the onboarding process.

This study relies on self-reported data, which captures perceived use but not actual behavior or learning outcomes. Advancing this line of research requires observational and longitudinal studies that connect onboarding practices to concrete indicators of domain understanding and its impact on technical decision-making.

The increasing use of AI tools introduces a critical open question. While AI can support task execution, it remains unclear whether it reduces the need for domain knowledge or makes it even more important for interpreting and validating generated outputs. Understanding this shift is essential for explaining how domain knowledge is constructed and sustained in contemporary software development.

7 Threats to Validity

We discuss threats to validity following the classification proposed by Wohlin et al. [42]. As this study is based on a cross-sectional survey with self-reported data and non-probabilistic sampling, the findings should be interpreted as evidence about reported practices and perceived use of domain knowledge acquisition strategies, rather than as causal or statistically representative claims about the entire population of software developers.

Construct Validity. Construct validity concerns whether the study instruments adequately capture the concepts under investigation. In this study, business domain knowledge acquisition was operationalized through a set of learning strategies derived from prior work on onboarding, workplace learning, tacit knowledge transfer, and SE practice. These strategies covered experiential, social, structured, and self-directed forms of learning. However, domain knowledge is a complex and context-dependent construct, and no single survey item can capture all aspects of how developers understand a business domain.

A related threat is that participants may have interpreted some strategies differently depending on their organizational context. For instance, "mentor support" may refer to a formal mentor in some companies and to occasional guidance from a senior colleague in others. Similarly, "interaction with colleagues" may involve different levels of depth, from quick technical clarification to sustained explanation of business rules. To mitigate this threat, the instrument was reviewed internally and piloted with experienced professionals before deployment. Nevertheless, variation in interpretation cannot be fully eliminated and should be considered when reading the results.

Another construct validity threat concerns the use of self-reported measures. Participants rated how frequently they used each strategy during onboarding, but these responses reflect their perceptions and recollections rather than directly observed behavior. This is appropriate for the purpose of understanding developers' reported onboarding experiences, but it may be affected by recall bias, social desirability, or differences in how participants evaluate frequency of use. To reduce this risk, responses were anonymous, all strategy

items used the same scale, and the analysis focused on aggregate patterns rather than individual claims.

Importantly, frequency of reported use should not be interpreted as evidence of effectiveness. Highly reported strategies may be more accessible rather than more effective, while less frequently reported strategies may still be valuable when available. Therefore, the findings characterize reliance on strategies rather than their learning outcomes.

Internal Validity. Internal validity concerns whether the observed results can be attributed to the phenomenon under study rather than to uncontrolled factors. Since this study adopts a cross-sectional survey design, the results indicate associations and differences in reported strategy use, but they do not support causal claims. For example, differences across experience levels or work arrangements should not be interpreted as evidence that these factors directly cause changes in domain learning strategies. They indicate patterns that require further investigation through longitudinal or qualitative studies.

The filtering procedure also affects internal validity. We included only participants who reported having no or low prior business domain knowledge when joining the project. This criterion strengthens alignment with the research objective, because the study focuses on developers who had to acquire domain knowledge during onboarding. However, prior domain knowledge was self-assessed, and participants may have used different standards to classify their initial familiarity. To mitigate this threat, the analysis treated this criterion as a practical filter for identifying the target population rather than as an objective measurement of domain expertise.

Data quality was supported through cleaning procedures and the exclusion of responses that did not meet the inclusion criteria. All survey questions were mandatory, except for an optional open-ended field, which reduced missing data. The statistical analysis was also aligned with the characteristics of the data: strategy variables were treated as ordinal measures, normality was assessed, and non-parametric tests were used when comparing strategies and groups.

External Validity. External validity concerns the extent to which the findings can be generalized beyond the studied sample. The survey used non-probabilistic sampling through professional networks and groups, which may introduce sampling bias. Therefore, the results should not be interpreted as statistically representative of all software developers in Brazil or other countries. Instead, they provide exploratory evidence about how a diverse set of Brazilian developers reported acquiring business domain knowledge during onboarding.

Although participants came from all Brazilian regions, the sample was unevenly distributed and concentrated in the South. The sample was also predominantly male, which limits the strength of conclusions about gender-related differences. In addition, some organizational contexts, such as large companies and technology-oriented sectors, were more represented than others. These characteristics may influence the availability of strategies such as formal training, documentation, mentoring, and client interaction.

The study intentionally focuses on developers with limited prior domain knowledge. This decision increases the coherence of the analysis, because it isolates the situation in which domain learning is most relevant during onboarding. However, it also restricts the applicability of the findings. Developers who join projects with medium or high prior domain knowledge may rely on different strategies, face different challenges, or require less support. Future studies should compare these profiles to examine how prior domain familiarity changes onboarding needs.

Conclusion Validity. Conclusion validity concerns the reliability of the inferences drawn from the data. The study used standardized items, a consistent response scale, and statistical procedures appropriate for ordinal and non-normally distributed data. The use of medians and non-parametric tests reduces the risk of drawing conclusions based on assumptions that are not supported by the data distribution.

Nevertheless, the interpretation of statistical patterns requires caution. The survey captures the reported intensity of strategy use, but it does not measure the effectiveness or quality of learning produced by each strategy. Furthermore, AI tools were treated as a broad category, preventing conclusions about the effects of specific AI technologies on business domain learning. Therefore, the discussion interprets the results as evidence of reported use and variation rather than as direct evidence of learning outcomes or the effectiveness of particular approaches.

To improve interpretive reliability, the initial analysis was conducted by the first author and reviewed by the remaining authors. Disagreements were discussed until consensus was reached. Although this procedure does not eliminate researcher judgment, it reduces the likelihood that the findings depend on a single interpretation of the data.

8 Conclusion

This study investigated which strategies software developers report using to acquire business domain knowledge during onboarding and how the use of these strategies varies across individual and organizational contexts. Based on a survey with 257 software developers in Brazil, the results show that domain learning is primarily experiential and social, especially task execution, interaction with colleagues, and reading code or architecture artifacts. In contrast, formal training, internal documentation, customer interaction, and AI-based assistance were reported less consistently.

These findings position business domain knowledge as a distinct and situated dimension of developer onboarding. The main lesson is that domain understanding is rarely acquired before work begins; it is constructed through work, but only when tasks, artifacts, peers, and stakeholders expose the rationale behind system behavior. Consequently, onboarding practices should verify domain reasoning instead of treating productivity as evidence of domain understanding.

Future work should examine how different combinations and sequences of strategies contribute to actual domain understanding over time, especially as AI tools become part of onboarding. This requires qualitative and longitudinal studies capable of connecting reported strategy use to concrete indicators of domain reasoning in software decisions.

Artifact Availability

The replication package associated with this study is available: Zenodo repository.

References

[1] Alejandrina M. Aranda, Oscar Dieste, and Natalia Juristo. 2016. Effect of Domain Knowledge on Elicitation Effectiveness: An Internally Replicated Controlled Experiment. *IEEE Transactions on Software Engineering* 42, 5 (2016), 427–451. doi:10.1109/TSE.2015.2494588

[2] Marcos Araújo, Juliana Araújo, Rafael Oliveira, Lucas Romao, and Marcos Kalinowski. 2026. Domain Knowledge in Requirements Engineering: A Systematic Mapping Study. In *Software Engineering and Advanced Applications (Lecture Notes in Computer Science, Vol. 16082)*. Springer, Cham. doi:10.1007/978-3-032-04200-2_29

[3] Talya N. Bauer. 2010. *Onboarding New Employees: Maximizing Success.* SHRM Foundation.

[4] Talya N. Bauer, Berrin Erdogan, Allison M. Ellis, Donald M. Truxillo, Gregory M. Brady, and Todd Bodner. 2025. New Horizons for Newcomer Organizational Socialization: A Review, Meta-Analysis, and Future Research Directions. *Journal of Management* 51, 1 (2025), 344–382. doi:10.1177/01492063241277168

[5] Ricardo Britto, Daniela S. Cruzes, Darja Smite, and Aivars Sablis. 2018. Onboarding software developers and teams in three globally distributed legacy projects: A multi-case study. *Journal of Software: Evolution and Process* 30, 4 (2018), e1921. doi:10.1002/smr.1921

[6] R. Britto, D. Smite, L. O. Damm, and J. Börstler. 2020. Evaluating and Strategizing the Onboarding of Software Developers in Large-Scale Globally Distributed Projects. *Journal of Systems and Software* 169 (2020), 110699. doi:10.1016/j.jss.2020.110699

[7] Vebjørn Brødsjø, Benjamin Sandøy, and Eli Hustad. 2023. Exploring Onboarding Processes for IT Professionals: The Role of Knowledge Management. In *Proceedings of the European Conference on Knowledge Management (ECKM).*

[8] Jim Buchan, Stephen G. MacDonell, and Jennifer Yang. 2019. Effective Team Onboarding in Agile Software Development: Techniques and Goals. In *Proceedings of the 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM).* IEEE/ACM, 1–11. doi:10.1109/ESEM.2019.8870189

[9] Petros Chamakiotis, Niki Panteli, and Diana Pérez-Arechaederra. 2025. Practices for Effective Onboarding in Dynamic Global Virtual Teams. *Organizational Dynamics* 54, 1 (2025), 101076. doi:10.1016/j.orgdyn.2024.101076

[10] Paul Clarke and Rory V. O'Connor. 2015. Changing Situational Contexts Present a Constant Challenge to Software Developers. In *Systems, Software and Services Process Improvement (Communications in Computer and Information Science, Vol. 543).* Springer, Cham. doi:10.1007/978-3-319-24647-5_9

[11] Rafael Maiani de Mello and Guilherme Horta Travassos. 2016. Surveys in Software Engineering: Identifying Representative Samples. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM).* ACM, New York, NY, USA, Article 55. doi:10.1145/2961111.2962632

[12] Michael Eraut. 2000. Non-Formal Learning and Tacit Knowledge in Professional Work. *British Journal of Educational Psychology* 70 (2000), 113–136. doi:10.1348/000709900158001

[13] Denae Ford, Margaret-Anne Storey, Thomas Zimmermann, Christian Bird, Sonia Jaffe, Chandra Maddila, Jenna L. Butler, Brian Houck, and Nachiappan Nagappan. 2022. A Tale of Two Cities: Software Developers Working from Home during the COVID-19 Pandemic. *ACM Transactions on Software Engineering and Methodology* 31, 2, Article 27 (2022), 37 pages. doi:10.1145/3487567

[14] Ahmad Nauman Ghazi, Kai Petersen, Sri Sai Vijay Raj Reddy, and Harini Nekkanti. 2019. Survey Research in Software Engineering: Problems and Mitigation Strategies. *IEEE Access* 7 (2019), 24703–24725. doi:10.1109/ACCESS.2018.2881041

[15] Verica Gluvakov, Mila Kavalić, Maja Gaborov, Igor Vecštejn, and Željko Stojanov. 2025. Onboarding in Small Software Enterprises: Practical Recommendations from a Qualitative Case Study. *Journal of Engineering Management and Competitiveness* 15, 1 (2025), 16–31. doi:10.5937/JEMC2501016G

[16] Daniel Graziotin, Per Lenberg, Robert Feldt, and Stefan Wagner. 2022. Psychometrics in Behavioral Software Engineering: A Methodological Introduction with Guidelines. *ACM Transactions on Software Engineering and Methodology* 31, 1, Article 7 (2022). doi:10.1145/3469888

[17] Peggy Gregory, Diane E. Strode, Ruba AlQaisi, Helen Sharp, and Leonor Barroca. 2020. Onboarding: How Newcomers Integrate into an Agile Project Team. In *Agile Processes in Software Engineering and Extreme Programming (Lecture Notes in Business Information Processing, Vol. 383).* Springer, Cham. doi:10.1007/978-3-030-49392-9_2

[18] Irit Hadar, Pnina Soffer, and Keren Kenzi. 2014. The role of domain knowledge in requirements elicitation via interviews: an exploratory study. *Requirements Engineering* 19, 2 (2014), 143–159. doi:10.1007/s00766-012-0163-2

[19] Gary S. Insch, Nancy McIntyre, and David Dawley. 2008. Tacit Knowledge: A Refinement and Empirical Test of the Academic Tacit Knowledge Scale. *The Journal of Psychology* 142, 6 (2008), 561–580. doi:10.3200/JRLP.142.6.561-580

[20] Debora Jeske and David Olson. 2022. Onboarding New Hires: Recognising Mutual Learning Opportunities. *Journal of Work-Applied Management* 14, 1 (2022), 63–76. doi:10.1108/JWAM-04-2021-0036

[21] Maggie Johnson and Max Senges. 2010. Learning to be a programmer in a complex organization: A case study on practice-based learning during the onboarding process at Google. *Journal of Workplace Learning* 22, 3 (2010), 180–194. doi:10.1108/13665621011028620

[22] An Ju, Hitesh Sajnani, Scot Kelly, and Kim Herzig. 2021. A Case Study of Onboarding in Software Teams: Tasks and Strategies. In *Proceedings of the 43rd International Conference on Software Engineering (ICSE).* IEEE/ACM, 613–623. doi:10.1109/ICSE43902.2021.00063

[23] Mihai Alin Lazar. 2025. The Impact of Digital Natives' Expectations on IT Onboarding Processes. *Economics and Applied Informatics* 1 (2025), 146–153. doi:10.35219/eai15840409494

[24] Tong Li, Xinran Zhang, Yunduo Wang, Qixiang Zhou, Yiting Wang, and Fangqi Dong. 2024. Machine learning for requirements engineering (ML4RE): A systematic literature review complemented by practitioners' voices from Stack Overflow. *Information and Software Technology* 172 (2024), 107477. doi:10.1016/j.infsof.2024.107477

[25] Johan Linåker, Sardar Muhammad Sulaman, Rafael Maiani de Mello, and Martin Höst. 2015. *Guidelines for Conducting Surveys in Software Engineering.* Technical Report. Lund University, Department of Computer Science, Lund, Sweden. https://lup.lub.lu.se/search/files/6062997/5463412.pdf

[26] Gustavo Matturro, Karina Barrella, and Pablo Benitez. 2017. Difficulties of Newcomers Joining Software Projects Already in Execution. In *Proceedings of the 2017 International Conference on Computational Science and Computational Intelligence (CSCI).* IEEE, 993–998. doi:10.1109/CSCI.2017.171

[27] Annabell Mitschelen and Simone Kauffeld. 2025. Workplace learning during organizational onboarding: integrating formal, informal, and self-regulated workplace learning. *Frontiers in Organizational Psychology* 3 (2025). doi:10.3389/forgp.2025.1569098

[28] Jefferson Seide Molléri, Kai Petersen, and Emilia Mendes. 2016. Survey Guidelines in Software Engineering: An Annotated Review. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM).* ACM, New York, NY, USA, 58:1–58:6. doi:10.1145/2961111.2962619

[29] Riordan Moraes, Allysson Allex Araújo, Luis Rivero, and Davi Viana. 2024. Investigating the Onboarding Process for Interns in Software Development: An Initial Survey. In *Anais do XXXVIII Simpósio Brasileiro de Engenharia de Software (SBES).* Sociedade Brasileira de Computação, Porto Alegre, Brazil, 693–699. doi:10.5753/sbes.2024.3652

[30] Johan Olaisen and Øystein Revang. 2018. Exploring the Performance of Tacit Knowledge: How to Make Ordinary People Deliver Extraordinary Results in Teams. *International Journal of Information Management* 43 (2018), 295–304. doi:10.1016/j.ijinfomgt.2018.08.016

[31] Sara Petrilli, Laura Galuppo, and Silvio Carlo Ripamonti. 2022. Digital Onboarding: Facilitators and Barriers to Improve Worker Experience. *Sustainability* 14, 9 (2022), 5684. doi:10.3390/su14095684

[32] Ana Beatriz Cavalcanti Ribeiro and Carina Alves. 2026. Onboarding in Software Engineering: A Multivocal Literature Review. *Information and Software Technology* 196 (2026), 108120. doi:10.1016/j.infsof.2026.108120

[33] Eva Ritz, Fabio Donisi, Edona Elshan, and Roman Rietsche. 2023. Artificial Socialization? How Artificial Intelligence Applications Can Shape A New Era of Employee Onboarding Practices. In *Proceedings of the 56th Annual Hawaii International Conference on System Sciences (HICSS 2023).* IEEE Computer Society, 155–164. doi:10.24251/HICSS.2023.020

[34] Sharon Ryan and Rory V. O'Connor. 2013. Acquiring and Sharing Tacit Knowledge in Software Development Teams: An Empirical Study. *Information and Software Technology* 55, 9 (2013), 1614–1624. doi:10.1016/j.infsof.2013.02.013

[35] Italo Santos, Katia Romero Felizardo, Igor Steinmacher, and Marco A. Gerosa. 2025. Software Solutions for Newcomers' Onboarding in Software Projects: A Systematic Literature Review. *Information and Software Technology* 177 (2025), 107568. doi:10.1016/j.infsof.2024.107568

[36] Gaurav G. Sharma and Klaas-Jan Stol. 2020. Exploring Onboarding Success, Organizational Fit, and Turnover Intention of Software Professionals. *Journal of Systems and Software* 159 (2020), 110442. doi:10.1016/j.jss.2019.110442

[37] Dilruba Showkat. 2018. Determining Newcomers Barrier in Software Development: An IT Industry Based Investigation. In *Companion of the 2018 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '18 Companion).* Association for Computing Machinery, New York, NY, USA, 165–168. doi:10.1145/3272973.3274046

[38] Darya B. Shtrikova, A. B. Shtrikov, and A. L. Busygina. 2020. Onboarding New Employees Taking Into Account Gender Differences. In *Global Challenges and Prospects of the Modern Economic Development (European Proceedings of Social and Behavioural Sciences, Vol. 79)*, S. I. Ashmarina and V. V. Mantulenko (Eds.). European Publisher, 592–600. doi:10.15405/epsbs.2020.03.85

[39] Rini van Solingen, Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. 2002. Goal Question Metric (GQM) Approach. In *Encyclopedia of Software Engineering*, J. J. Marciniak (Ed.). Wiley. doi:10.1002/0471028959.sof142

[40] Davi Viana, Tayana Uchôa Conte, and Cleidson R. B. de Souza. 2014. Knowledge Transfer between Senior and Novice Software Engineers: A Qualitative Analysis.

In *Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering (SEKE)*. 329–334.

[41] Titus Winters, Tom Manshreck, and Hyrum Wright. 2020. *Software Engineering at Google: Lessons Learned from Programming Over Time*. O'Reilly Media, Sebastopol, CA, USA.

[42] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-29044-2

[43] Taro Yamane. 1973. *Statistics: An Introductory Analysis* (3 ed.). Harper & Row.