ORIGINAL ARTICLE

INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE

ISSN: 2692-515-x, Impact Factor: 9.23
American Academic publishers, volume 6, issue 01,2026

Journal: https://www.academicpublishers.org/journals/index.php/ijai

DEVELOPING SPEAKING COMPETENCE THROUGH COLLABORATIVE LEARNING IN SOFTWARE ENGINEERING EDUCATION

Khulkaroy Mamatalieva,
senior teacher, Department of Languages-2
Oriental University
E-mail: hulkar9091@gmail.com
Mehribonu Murodova Bakhtiyor kizi
Student, Faculty of Economics and Information technologies

Abstract

In the context of rapid technological advancement and globalization, software engineering graduates are increasingly expected to demonstrate not only strong technical expertise but also advanced communicative competence in English. Speaking competence, in particular, has become a critical component of professional effectiveness, enabling future engineers to participate in international teams, articulate design decisions, negotiate solutions, and collaborate within agile development environments. This article explores the potential of collaborative learning as a pedagogical approach to developing speaking competence among software engineering students in English-medium instruction contexts. The article argues that collaborative learning creates authentic communicative conditions that foster meaningful interaction, promote negotiation of meaning, and enhance learners' oral proficiency. Pedagogical implications for higher education institutions and curriculum designers are argued.

Key words

collaborative learning, speaking competence, software engineering, ESP, sociocultural theory, interactionist approach, pair programming, project-based learning, disciplinary discourse, professional identity

The increasing internationalization of higher education and the global nature of the software industry have significantly transformed the linguistic demands placed on software engineering graduates. English has become the dominant language of software documentation, programming communities, academic publications, and multinational development teams. Subsequently, universities are under pressure to ensure that future software engineers acquire not only technical knowledge but also the ability to communicate effectively in English, particularly in spoken interaction. Despite the recognized importance of speaking competence, traditional language instruction in technical universities often prioritizes reading and writing skills, while oral communication remains underdeveloped. This imbalance is especially problematic for software engineering students, whose professional activities frequently involve meetings, code reviews, sprint planning, presentations, and collaborative problem-solving discussions. Addressing this gap requires pedagogical approaches that align language learning with authentic professional practices.

Collaborative learning has emerged as a promising approach in both general education and language pedagogy. Rooted in sociocultural theory, collaborative learning emphasizes interaction, shared responsibility, and knowledge co-construction. In software engineering education, collaboration is not only a pedagogical choice but an inherent feature of the

ISSN: 2692-515-x, Impact Factor: 9.23
American Academic publishers, volume 6, issue 01,2026
Journal: https://www.academicpublishers.org/journals/index.php/ijai

discipline, reflected in agile methodologies, pair programming, and team-based project development.[1] This convergence creates a strong rationale for integrating collaborative learning into English language instruction for software engineering students. This article aims to examine how collaborative learning can be systematically employed to improve speaking competence in software engineering education. By synthesizing theoretical perspectives and empirical findings, the study proposes a pedagogical framework that connects collaborative learning principles with the communicative needs of future software engineers.[2]

Collaborative learning is broadly defined as an instructional approach in which learners work together in small groups to achieve shared learning goals. Unlike competitive or individualistic learning models, collaborative learning emphasizes positive interdependence, individual accountability, and promotive interaction. These core elements confirm that learners perceive their success as mutually linked, thereby creating communicative pressure to articulate ideas, clarify misunderstandings, and co-construct knowledge through spoken interaction. Johnson and Johnson argue that effective collaboration requires learners to engage in sustained face-to-face interaction, verbalize their reasoning processes, negotiate alternative viewpoints, and provide formative feedback to peers, all of which have direct and measurable implications for the development of speaking competence. From a pedagogical perspective, such interaction transforms speech from a peripheral classroom activity into a central cognitive tool, enabling learners to externalize thinking, rehearse disciplinary discourse, and gradually gain control over more complex oral language forms. In this sense, collaborative learning environments function not only as social spaces for cooperation but also as structured communicative ecosystems in which speaking development is systematically scaffolded and reinforced.

From a sociocultural perspective, Vygotsky's theory of cognitive development provides a foundational framework for understanding the role of collaboration in learning. According to Vygotsky, higher mental functions develop through social interaction, and learning occurs within the Zone of Proximal Development (ZPD), where learners can perform tasks with the support of more capable peers.[3] In collaborative settings, spoken language serves as a mediating tool that enables learners to externalize thought processes, negotiate understanding, and internalize new linguistic forms.

Interactionist theories of second language acquisition further highlight the importance of collaborative interaction. Long's Interaction Hypothesis posits that language development is facilitated through negotiation of meaning during communicative breakdowns.[4] When learners collaborate on tasks, they are more likely to request clarification, reformulate utterances, and provide feedback, thereby creating conditions conducive to speaking development. Swain's Output Hypothesis also underscores the significance of collaborative learning for speaking competence. Swain argues that producing language—particularly in interaction with others—

---

[1] Kent Beck et al., Manifesto for Agile Software Development (2001), accessed March 10, 2025, https://agilemanifesto.org.

[2] Laurie Williams and Robert R. Kessler, Pair Programming Illuminated. Boston: Addison-Wesley, 2003, 1–8.

[3] Lev S. Vygotsky, Mind in Society: The Development of Higher Psychological Processes, ed. Michael Cole et al. (Cambridge, MA: Harvard University Press, 1978), 86–90.

[4] Michael H. Long, "The Role of the Linguistic Environment in Second Language Acquisition," in Handbook of Second Language Acquisition, ed. William C. Ritchie and Tej K. Bhatia (San Diego: Academic Press, 1996), 413–468.

ISSN: 2692-515-x, Impact Factor: 9.23
American Academic publishers, volume 6, issue 01,2026
Journal: https://www.academicpublishers.org/journals/index.php/ijai

pushes learners to process language more deeply, notice gaps in their linguistic knowledge, and refine their output. Collaborative tasks in software engineering contexts, such as discussing algorithms or explaining design choices, naturally elicit extended spoken output and promote metalinguistic awareness. Speaking competence in English for Specific Purposes (ESP) extends beyond general conversational ability and encompasses discipline-specific communicative practices that are shaped by the epistemology, standards, and interactional norms of a given professional field. For software engineering students, speaking competence involves not only linguistic fluency but also the ability to describe complex technical processes with precision, justify design decisions through logical and evidence-based argumentation, participate actively in collaborative problem-solving, and communicate effectively with both technical and non-technical stakeholders [1.8]. Importantly, such competence requires mastery of specialized discourse genres common to software engineering, including technical briefings, code explanations, sprint retrospectives, and design presentations. From a pedagogical standpoint, this implies that speaking instruction for software engineering students must move beyond generic communicative tasks and instead engage learners in discipline-embedded oral practices that mirror authentic professional communication. Developing speaking competence in this context therefore entails the integration of linguistic, pragmatic, and professional dimensions, enabling learners to use English as a functional tool for knowledge construction, collaboration, and decision-making within software engineering environments.

Research in ESP emphasizes the central role of needs analysis as the methodological foundation for designing effective and context-sensitive language instruction. Dudley-Evans and St John argue that ESP courses must be firmly grounded in the communicative demands of specific professional contexts, as language use in such settings is shaped by institutional practices, professional roles, and task-specific discourse conventions. In the field of software engineering, needs analysis reveals that spoken communication is predominantly interactional, problem-oriented, and collaborative in nature. Typical communicative events include participating in stand-up meetings, explaining code functionality to peers, engaging in peer reviews, negotiating technical solutions, and presenting project outcomes to diverse audiences. A deeper analysis of these activities indicates that they require far more than surface-level linguistic accuracy. Effective participation in stand-up meetings, for instance, demands concise reporting, temporal structuring of information, and the ability to respond spontaneously to follow-up questions. Explanation of code functionality includes the use of procedural discourse, logical sequencing, and audience-aware simplification, particularly when interacting with non-technical stakeholders. Peer reviews require pragmatic sensitivity, as speakers must balance critique with politeness strategies in order to maintain professional rapport, while project presentations demand control over extended discourse, rhetorical organization, and persuasive language use. From a discourse-analytic perspective, these communicative practices illustrate that speaking competence in software engineering is inherently multi-dimensional, integrating grammatical competence, pragmatic competence, and discourse management skills. Consequently, ESP instruction that fails to address these layered communicative demands risks producing learners who are linguistically competent yet professionally inarticulate. A needs-based, analytically informed approach therefore provides the necessary bridge between language pedagogy and the communicative realities of software engineering practice, reinforcing the argument for collaborative learning as an optimal instructional response to these complex speaking requirements.

Traditional language teaching methods often fail to address these requirements, as they rely on decontextualized speaking exercises that bear little resemblance to authentic professional

ISSN: 2692-515-x, Impact Factor: 9.23
American Academic publishers, volume 6, issue 01,2026
Journal: https://www.academicpublishers.org/journals/index.php/ijai

communication. In contrast, collaborative learning tasks can be designed to mirror real-world software engineering practices, thereby enhancing the relevance and effectiveness of speaking instruction. Moreover, speaking competence in technical fields is closely linked to learners' confidence, professional identity, and their sense of legitimate participation within the disciplinary community. Students who lack opportunities to practice spoken English in meaningful and professionally relevant contexts may experience heightened communication anxiety, reduced self-efficacy, and a reluctance to participate actively in collaborative work. Such affective constraints are particularly evident among software engineering students, who may possess strong technical knowledge yet struggle to articulate ideas, defend design decisions, or engage in technical discussions in English. Collaborative learning environments, when properly organized, can significantly lower these affective barriers by fostering peer support, shared responsibility, and a non-threatening space for experimentation with language. Through sustained interaction with peers, learners gradually develop confidence in their spoken performance, internalize disciplinary discourse conventions, and begin to perceive themselves not merely as language learners but as emerging software engineering professionals capable of contributing meaningfully to collaborative technical dialogue.

Software engineering education has a long tradition of employing collaborative learning models, particularly in the form of project-based learning (PBL) and team-based instruction. These models align closely with industry practices and emphasize communication, coordination, and collective problem-solving. Project-based learning involves students working collaboratively over an extended period to design, develop, and evaluate a software product. Within this framework, spoken interaction is essential for planning tasks, resolving conflicts, and making joint decisions. This arrangement requires continuous verbal communication, including explaining code, asking questions, and providing feedback.[5] When integrated with language learning objectives, PBL can serve as a powerful context for developing speaking competence. Pair programming, a core practice in agile software development, represents another collaborative model with significant potential for speaking development. In pair programming, two students work together at one workstation, alternating roles as driver and navigator.

References

1. Kent Beck et al., Manifesto for Agile Software Development (2001), accessed March 10, 2025, https://agilemanifesto.org.
2. Laurie Williams and Robert R. Kessler, Pair Programming Illuminated. Boston: Addison-Wesley, 2003, 1–8.
3. Lev S. Vygotsky, Mind in Society: The Development of Higher Psychological Processes, ed. Michael Cole et al. (Cambridge, MA: Harvard University Press, 1978), 86–90.
4. Michael H. Long, "The Role of the Linguistic Environment in Second Language Acquisition," in Handbook of Second Language Acquisition, ed. William C. Ritchie and Tej K. Bhatia (San Diego: Academic Press, 1996), 413–468.
5. Laurie Williams and Robert Kessler, Pair Programming Illuminated (Boston: Addison-Wesley, 2002), 43–47.

[5] Laurie Williams and Robert Kessler, Pair Programming Illuminated (Boston: Addison-Wesley, 2002), 43–47.