Pedagogical Research and Development

Volume: 5 Issue: 2

2026

ISSN: 2791-3627

KMF PUBLISHERS

http://kmf-publishers.com/prd/

DOI: https://doi.org/10.64907/xkmf.v5i2.prd.2

RESEARCH ARTICLE

OPEN ACCESS Freely available online

Soft Skills Integration in Programming Courses: A Multiple Case Study of Pedagogical Practices in CSE Departments

Mahedi Sharker Moin[1*]; Kazi Abdul Mannan[2]

Abstract: The increasing complexity of the global software industry has highlighted the critical need for integrating soft skills into programming education within Computer Science and Engineering (CSE) programs. This study investigates how soft skills, such as communication, teamwork, problem-solving, and adaptability, are embedded in programming courses through a qualitative multiple case study approach based on secondary data. Drawing on constructivist learning theory, experiential learning theory, and social learning theory, the research examines pedagogical practices across diverse institutional contexts. The findings reveal that strategies such as project-based learning, collaborative programming, peer assessment, reflective practices, and Agile methodologies play a significant role in fostering soft skills alongside technical competencies. However, challenges persist, including curriculum constraints, assessment difficulties, limited faculty preparedness, and student resistance. The study proposes that effective integration requires a holistic, context-driven approach that aligns pedagogical design with industry expectations. By synthesising existing practices and theoretical insights, this research contributes to the growing discourse on curriculum reform in programming education and offers practical implications for educators, curriculum designers, and policymakers seeking to enhance graduate employability in the digital era.

Keywords: soft skills integration, programming education, computer science education, project-based learning, collaborative learning, employability, pedagogical practices

Received: 2 May 2026
Accepted: 25 June 2026

Corresponding author:
[1*]Mahedi Sharker Moin
[1]Department of Computer Science & Engineering (CSE)
[2]Department of Business Administration
Shanto-Mariam University of Creative Technology
Dhaka, Bangladesh
E-mail: msmoin16@gmail.com

Disclosure statement
No potential conflict of interest was reported by the author(s).

Citation information: Cite this article as: Moin, M.S., & Mannan, K.A, (2026). Soft Skills Integration in Programming Courses: A Multiple Case Study of Pedagogical Practices in CSE Departments. Pedagogical Research and Development, 5(2), 1-17. DOI: https://doi.org/10.64907/xkmf.v5i2.prd.2

1. Introduction

The rapid advancement of digital technologies and the increasing complexity of software systems have significantly reshaped the expectations placed on graduates of Computer Science and Engineering (CSE) programs. Traditionally, programming education has emphasised the acquisition of technical competencies such as algorithmic thinking, coding proficiency, and system design. While these skills remain foundational, there is growing recognition that technical expertise alone is insufficient for success in modern, collaborative, and interdisciplinary work environments (Robles, 2012; Andrews & Higson, 2008). Consequently, the integration of soft skills into programming courses has emerged as a critical concern in higher education.

Soft skills, often defined as a combination of interpersonal, communication, and cognitive abilities, play a crucial role in shaping an individual's professional effectiveness. These include teamwork, problem-solving, adaptability, leadership, and emotional intelligence (Heckman & Kautz, 2012). In the context of software development, these competencies are indispensable. Modern software engineering practices, such as Agile development, DevOps, and collaborative version control, require developers to communicate effectively, work in teams, and respond to rapidly changing requirements (Beck et al., 2001; Lethbridge, 2000). As such, employers increasingly prioritise graduates who can demonstrate both technical and soft skills.

Despite this shift in industry expectations, many CSE curricula continue to prioritise technical instruction at the expense of soft skills development. Programming courses, in particular, are often structured around individual assignments, standardised assessments, and a focus on correctness and efficiency. This approach can limit opportunities for students to develop essential interpersonal and professional skills (Shuman et al., 2005). As a result, graduates may possess strong coding abilities but struggle to collaborate effectively, communicate ideas, or adapt to real-world challenges.

The gap between academic preparation and industry expectations has been widely documented in the literature. Employers frequently report dissatisfaction with graduates' soft skills, particularly in areas such as communication, teamwork, and problem-solving (Andrews & Higson, 2008; Robles, 2012). This mismatch has prompted calls for a more holistic approach to engineering education, one that integrates soft skills within technical courses rather than treating them as separate or supplementary components.

Programming courses present a unique opportunity for such integration. Unlike theoretical courses, programming inherently involves problem-solving, iterative development, and, increasingly, collaboration. By embedding soft skills into programming activities, such as group projects, pair programming, and peer code reviews, educators can create learning environments that mirror real-world software development practices (Williams & Kessler,

2003). These approaches not only enhance technical learning but also foster the development of communication, teamwork, and critical thinking skills.

However, integrating soft skills into programming courses is not without challenges. One major barrier is the lack of clear pedagogical frameworks for incorporating and assessing these skills. While technical competencies can be evaluated through objective measures, soft skills are often subjective and context-dependent, making assessment more complex (Shuman et al., 2005). Additionally, instructors may lack the training or resources needed to effectively facilitate soft skills development, particularly in large or resource-constrained classrooms.

Another challenge lies in student perceptions. Many students view programming as a purely technical discipline and may not immediately recognise the relevance of soft skills. This perception can lead to resistance or disengagement when soft skills are introduced into technical courses. Overcoming this challenge requires deliberate instructional design and clear communication of the value of soft skills in professional contexts.

In response to these challenges, educators and researchers have explored a range of pedagogical strategies aimed at integrating soft skills into programming education. These include project-based learning, collaborative learning, peer assessment, and experiential learning approaches. While these strategies have shown promise, their implementation varies widely across institutions, and there is a need for a more systematic understanding of effective practices.

This study addresses this gap by examining how soft skills are integrated into programming courses through a qualitative multiple case study approach based on secondary data. By analysing existing literature, institutional reports, and documented case studies, the research aims to identify common pedagogical practices, evaluate their effectiveness, and highlight challenges and opportunities for improvement.

The study is guided by the following research questions:

- How are soft skills integrated into programming courses in CSE departments?
- What pedagogical practices are most effective in fostering soft skills development?
- What challenges do educators face in integrating and assessing soft skills?

By answering these questions, the study contributes to the ongoing discourse on curriculum reform in engineering education. It also provides practical insights for educators, curriculum designers, and policymakers seeking to enhance the quality and relevance of programming education in the digital age.

2. Literature Review

Soft skills have been conceptualised in various ways across disciplines, often described as non-technical competencies that complement domain-specific knowledge.

Heckman and Kautz (2012) define soft skills as a broad set of personal attributes, including perseverance, motivation, and social skills, that influence an individual's ability to succeed in life and work. In engineering education, these skills are commonly associated with professional competencies such as communication, teamwork, ethical reasoning, and lifelong learning (Shuman et al., 2005).

The Accreditation Board for Engineering and Technology (ABET) has explicitly recognised the importance of soft skills by incorporating them into its student outcomes criteria. These include the ability to function effectively on teams, communicate with diverse audiences, and engage in continuous learning (Shuman et al., 2005). Such recognition underscores the growing consensus that engineering graduates must be equipped with both technical and soft skills to meet the demands of the profession.

In programming education, soft skills are particularly relevant due to the collaborative nature of modern software development. Developers must work in teams, interact with stakeholders, and adapt to changing requirements, all of which require strong interpersonal and cognitive abilities (Begel & Simon, 2008). Thus, the integration of soft skills into programming courses is not merely desirable but essential.

2.1 The Employability Imperative and Industry Expectations

The importance of soft skills in enhancing employability has been widely documented. Employers consistently rank soft skills among the most critical attributes for new hires (Robles, 2012). In a study by Andrews and Higson (2008), employers emphasised communication, teamwork, and problem-solving as key competencies that graduates often lack.

In the software industry, these skills are even more critical. Agile methodologies, which dominate contemporary software development practices, rely heavily on collaboration, communication, and adaptability (Beck et al., 2001). Developers are expected to participate in team discussions, provide feedback, and continuously improve their work processes. As such, the ability to work effectively in teams and communicate clearly is as important as technical proficiency.

Lethbridge (2000) highlights that software professionals require a combination of technical knowledge and soft skills to perform effectively. The study found that while technical skills are essential, soft skills such as communication and teamwork significantly influence job performance and career progression.

2.3 Pedagogical Approaches to Soft Skills Integration

2.3.1 Project-Based Learning (PBL)

Project-Based Learning (PBL) is one of the most widely adopted approaches for integrating soft skills into programming courses. PBL involves engaging students in complex, real-world projects that require collaboration, problem-solving, and critical thinking (Thomas, 2000). Through these projects, students develop both technical and soft skills in an authentic context.

Research indicates that PBL enhances student engagement and promotes deeper learning. It also provides opportunities for students to practice communication and teamwork, as they must coordinate with peers and present their work (Bell, 2010).

2.3.2 Collaborative Learning and Pair Programming

Collaborative learning strategies, including pair programming, are effective in fostering soft skills. Pair programming involves two students working together on the same task, with one acting as the "driver" and the other as the "navigator" (Williams & Kessler, 2003).

This approach encourages continuous communication, mutual support, and shared responsibility. Studies have found that pair programming improves not only technical outcomes but also students' confidence and interpersonal skills (Williams & Kessler, 2003).

2.3.3 Peer Assessment and Feedback

Peer assessment is another valuable tool for developing soft skills. By evaluating each other's work, students learn to provide constructive feedback and engage in critical reflection (Topping, 1998). This process enhances communication skills and fosters a sense of accountability.

Moreover, peer assessment aligns with social learning theory, as students learn from observing and interacting with their peers (Bandura, 1977).

2.3.4 Experiential and Reflective Learning

Experiential learning emphasises learning through experience, while reflective learning focuses on analysing those experiences to gain insights (Kolb, 1984). In programming courses, experiential learning can be facilitated through hands-on projects, internships, and simulations.

Reflective practices, such as journals and self-assessments, help students develop self-awareness and critical thinking skills. These practices are essential for continuous improvement and lifelong learning.

2.4 Challenges in Integrating Soft Skills

Despite the recognised importance of soft skills, several challenges hinder their integration into programming courses.

2.4.1 Curriculum Constraints

One of the primary challenges is the already crowded curriculum. Programming courses often have extensive technical content, leaving little room for additional activities focused on soft skills (Shuman et al., 2005).

2.4.2 Assessment Difficulties

Assessing soft skills is inherently complex due to their subjective nature. Unlike technical skills, which can be evaluated through tests and assignments, soft skills require more nuanced assessment methods, such as observations and peer evaluations (Topping, 1998).

2.4.3 Faculty Preparedness

Instructors may lack the training or experience needed to teach and assess soft

skills effectively. This can lead to inconsistent implementation and limited impact (Shuman et al., 2005).

2.4.4 Student Attitudes

Students may not initially recognise the importance of soft skills, particularly in technical disciplines. This can result in resistance or lack of engagement in activities designed to develop these skills.

2.5 Synthesis and Research Gap

The literature clearly demonstrates the importance of soft skills in programming education and identifies several effective pedagogical approaches. However, there is a lack of comprehensive studies that synthesise these approaches across multiple contexts.

Most existing research focuses on individual interventions or specific institutions, limiting the generalizability of findings. Additionally, there is a need for studies that examine how different pedagogical practices interact and contribute to soft skills development.

This study addresses these gaps by adopting a multiple case study approach based on secondary data, providing a broader perspective on pedagogical practices in CSE departments.

3. Theoretical Framework

The integration of soft skills into programming courses requires a robust theoretical foundation to guide pedagogical design and interpret learning processes. This study draws on three complementary theoretical perspectives: constructivist learning theory, experiential learning theory, and social learning theory. Together, these frameworks provide a multidimensional understanding of how students acquire both technical and soft skills within programming education contexts.

3.1 Constructivist Learning Theory

Constructivist learning theory posits that learners actively construct knowledge through interaction with their environment rather than passively receiving information (Piaget, 1970; Vygotsky, 1978). Learning, from this perspective, is a dynamic process shaped by prior knowledge, social interaction, and contextual engagement.

In programming education, constructivism manifests through active problem-solving, collaborative tasks, and inquiry-based learning. Students are not merely consumers of code but active participants in knowledge creation. This approach aligns with the integration of soft skills, as students must engage in communication, negotiation, and teamwork while constructing solutions.

Vygotsky's (1978) concept of the *Zone of Proximal Development (ZPD)* is particularly relevant. It suggests that learners achieve higher levels of understanding through interaction with more knowledgeable peers or instructors. In programming courses, practices such as pair programming and group projects create opportunities for scaffolding, where students support each other's learning. These interactions inherently foster soft skills, including communication and collaboration.

Furthermore, constructivist environments encourage reflection and metacognition, which are essential components of soft skills development. Students learn to articulate

their thought processes, evaluate their performance, and adapt their strategies, thereby enhancing critical thinking and self-regulation.

3.2 Experiential Learning Theory

Kolb's (1984) experiential learning theory provides a cyclical model of learning consisting of four stages: concrete experience, reflective observation, abstract conceptualisation, and active experimentation. This framework is particularly applicable to programming education, where learning is inherently practice-oriented.

Programming courses often involve hands-on activities such as coding assignments, projects, and debugging exercises. These activities constitute the "concrete experience" stage. Reflection occurs when students analyse their code, receive feedback, and identify areas for improvement. Through conceptualisation, they develop general principles and strategies, which they then apply in new contexts.

Experiential learning is closely linked to soft skills development. For example, group projects provide opportunities for students to experience teamwork, resolve conflicts, and manage time effectively. Reflection on these experiences helps students develop self-awareness and emotional intelligence (Kolb, 1984).

Moreover, experiential learning emphasises the importance of context. Real-world or industry-based projects enable students to understand the relevance of soft skills in professional settings. This contextualization enhances motivation and engagement, making learning more meaningful.

3.3 Social Learning Theory

Social learning theory, developed by Bandura (1977), emphasises the role of observation, imitation, and social interaction in learning. According to this theory, individuals learn not only through direct experience but also by observing others' behaviours and the consequences of those behaviours.

In programming courses, social learning occurs through collaborative activities such as group work, peer reviews, and discussions. Students observe how peers approach problems, communicate ideas, and provide feedback. These observations influence their own behaviours and learning strategies.

Bandura's concept of *self-efficacy*, the belief in one's ability to succeed, is particularly relevant. Collaborative environments can enhance self-efficacy by providing opportunities for students to receive encouragement and validation from peers and instructors. Increased self-efficacy, in turn, contributes to improved performance and persistence.

Social learning theory also highlights the importance of modelling. Instructors play a critical role as role models, demonstrating effective communication, problem-solving, and professional behaviour. By observing these behaviours, students learn to emulate them in their own practice.

3.4 Integrated Theoretical Perspective

While each of these theories offers valuable insights, their integration provides a more

comprehensive framework for understanding soft skills development in programming education.

- **Constructivism** emphasises active engagement and knowledge construction.
- **Experiential learning** focuses on learning through practice and reflection.
- **Social learning** highlights the importance of interaction and observation.

Together, these perspectives suggest that soft skills are best developed in environments that are interactive, practice-oriented, and socially rich. Programming courses that incorporate collaborative projects, peer interactions, and reflective activities align well with this integrated framework.

This theoretical foundation informs the study's analysis of pedagogical practices, guiding the identification of strategies that effectively support both technical and soft skills development.

4. Methodology

This study adopts a qualitative multiple case study design based on secondary data. A qualitative approach is appropriate for exploring complex educational phenomena, such as the integration of soft skills into programming courses, where context and interpretation play a significant role (Creswell & Poth, 2018).

The multiple case study design enables the examination of pedagogical practices across different institutional contexts, providing a richer and more nuanced understanding than a single case study (Yin, 2018). By comparing multiple cases, the study identifies patterns, similarities, and differences in how soft skills are integrated into programming education.

4.1 Data Sources and Collection

The study relies on secondary data, which includes previously published and publicly available materials. These sources were selected to ensure a comprehensive and diverse dataset, encompassing different geographical regions, institutional types, and pedagogical approaches.

Data sources include:

- Peer-reviewed journal articles on programming education and soft skills
- Conference proceedings (e.g., ACM, IEEE education conferences)
- Institutional curriculum documents and course syllabi
- Accreditation reports (e.g., ABET criteria and evaluations)
- Published case studies and educational reports

The use of secondary data allows for the synthesis of existing knowledge and avoids the time and resource constraints associated with primary data collection. It also enables the inclusion of a wide range of cases, enhancing the study's generalizability.

4.2 Case Selection Criteria

Cases were selected using purposive sampling, guided by the following criteria:

- **Relevance:** The case must involve programming courses within CSE or related disciplines.
- **Evidence of Soft Skills Integration:** The case must explicitly describe pedagogical practices aimed at developing soft skills.
- **Data Availability:** Sufficient detail must be available to analyse the pedagogical approach and outcomes.
- **Diversity:** Cases should represent a variety of institutional contexts (e.g., universities, colleges, different countries).

This approach ensures that the selected cases are information-rich and relevant to the research questions (Patton, 2002).

4.3 Data Analysis

The study employs thematic analysis, a widely used method for identifying and interpreting patterns within qualitative data (Braun & Clarke, 2006).

The analysis process involved the following steps:

- **Familiarisation:** The researcher reviewed all selected documents to gain an overall understanding of the data.
- **Initial Coding:** Key segments of text were coded based on their relevance to soft skills integration, pedagogical practices, and challenges.
- **Theme Development:** Codes were grouped into broader themes, such as collaborative learning, project-based learning, and assessment strategies.
- **Review and Refinement:** Themes were reviewed to ensure consistency and coherence across cases.
- **Interpretation:** Themes were interpreted in relation to the theoretical framework, linking empirical findings to conceptual insights.

This systematic approach enhances the rigour and transparency of the analysis.

4.4 Ensuring Research Quality

To ensure the credibility and trustworthiness of the study, several strategies were employed:

Credibility: Credibility was enhanced through the use of multiple data sources, allowing for triangulation of findings (Creswell & Poth, 2018).

Transferability: By including diverse cases and providing detailed descriptions, the study enables readers to assess the applicability of findings to other contexts.

Dependability: A clear and systematic research process was followed, including documented procedures for data collection and analysis.

Confirmability: The study maintains objectivity by relying on documented evidence and minimising researcher bias.

4.5 Ethical Considerations

As the study is based on secondary data, it does not involve direct interaction with human participants. However, ethical considerations were still observed:

- Proper citation and acknowledgement of all sources
- Avoidance of plagiarism
- Accurate representation of original findings

These practices ensure academic integrity and respect for intellectual property (Mannan & Farhana, 2026).

4.6 Limitations of the Methodology

While the use of secondary data offers several advantages, it also has limitations:

- **Lack of Control:** The researcher has no control over data quality or collection methods.
- **Contextual Gaps:** Some cases may lack detailed contextual information.
- **Potential Bias:** Published studies may emphasise successful interventions, leading to publication bias.

Despite these limitations, the methodology provides a robust framework for synthesising existing knowledge and identifying broader trends.

5. Findings and Analysis

The thematic analysis of multiple secondary data sources reveals that the integration of soft skills into programming courses is increasingly recognised as a pedagogical necessity. Across the cases examined, several recurring themes emerge, including collaborative programming, project-based learning (PBL), peer assessment, reflective practices, and the adoption of industry-oriented methodologies such as Agile. These themes illustrate how soft skills are embedded within technical instruction and how they contribute to holistic student development.

5.1 Collaborative Programming as a Catalyst for Soft Skills Development

One of the most prominent findings is the widespread adoption of collaborative programming practices, particularly pair programming and team-based coding assignments. These approaches are grounded in both constructivist and social learning theories, as they require students to actively engage with peers in constructing knowledge (Vygotsky, 1978; Bandura, 1977).

Pair programming, where two students work together at a single workstation, has been shown to foster communication, mutual accountability, and shared problem-solving (Williams & Kessler, 2003). The “driver” and “navigator” roles necessitate continuous verbalisation of thought processes, thereby enhancing clarity of expression and active listening skills. This aligns with findings by Begel and Simon (2008), who observed that novice programmers often struggle to articulate their reasoning, and collaborative settings provide a structured opportunity to develop this ability.

Moreover, collaborative programming environments simulate real-world software development contexts, where teamwork is essential. Students learn to negotiate differing viewpoints, manage conflicts, and reach consensus, skills that are critical in professional settings. However, the effectiveness of collaborative programming depends on careful group formation and

instructor facilitation. Without proper guidance, issues such as unequal participation or "free riding" may arise, potentially undermining the intended learning outcomes.

5.2 Project-Based Learning and Authentic Skill Development

Project-Based Learning (PBL) emerges as a central pedagogical strategy for integrating soft skills into programming courses. PBL involves engaging students in complex, real-world projects that require sustained inquiry, collaboration, and problem-solving (Thomas, 2000; Bell, 2010).

The analysis indicates that PBL provides a natural context for soft skills development. Students working on projects must coordinate tasks, manage time, communicate effectively, and adapt to changing requirements. These activities mirror industry practices, thereby enhancing the relevance of learning experiences. For instance, software development projects often involve iterative design, testing, and refinement, which require both technical and interpersonal competencies.

From an experiential learning perspective, PBL facilitates the full learning cycle described by Kolb (1984). Students gain concrete experience through project work, reflect on their performance, conceptualise lessons learned, and apply them in subsequent iterations. This cyclical process not only deepens technical understanding but also fosters self-regulation and critical thinking.

However, the implementation of PBL is not without challenges. Some cases highlight issues related to assessment, as evaluating individual contributions within group projects can be difficult. Additionally, students may experience increased workload and stress, particularly when project expectations are not clearly defined.

5.3 Peer Assessment and the Development of Critical and Communication Skills

Peer assessment is identified as an effective mechanism for fostering soft skills, particularly critical thinking and communication. By evaluating each other's work, students engage in analytical reasoning and learn to provide constructive feedback (Topping, 1998).

The findings suggest that peer code reviews are particularly valuable in programming courses. Students review peers' code for correctness, efficiency, and readability, and provide suggestions for improvement. This process not only enhances technical skills but also promotes a culture of collaboration and continuous improvement.

From a social learning perspective, peer assessment enables observational learning, as students learn from both giving and receiving feedback (Bandura, 1977). It also encourages accountability, as students become more aware of the standards expected in their work.

Nevertheless, the effectiveness of peer assessment depends on clear guidelines and training. Without these, feedback may be superficial or biased. Some cases report that students initially struggle with providing constructive criticism, highlighting the need for structured support.

5.4 Reflective Practices and Metacognitive Development

Reflective practices, including journals, self-assessments, and group discussions, are widely used to support soft skills development. These practices encourage students to analyse their experiences, identify strengths and weaknesses, and develop strategies for improvement (Kolb, 1984).

The analysis indicates that reflection plays a crucial role in fostering metacognitive skills, such as self-awareness and self-regulation. Students who engage in reflective activities are better able to understand their learning processes and adapt their approaches accordingly.

Furthermore, reflection enhances emotional intelligence by encouraging students to consider their interactions with peers and their responses to challenges. This aligns with the broader definition of soft skills as encompassing both cognitive and emotional competencies (Heckman & Kautz, 2012).

However, reflective practices are sometimes perceived as less important than technical tasks, leading to limited student engagement. To address this, instructors must clearly communicate the value of reflection and integrate it meaningfully into course requirements.

5.5 Integration of Agile and Industry-Oriented Practices

Several cases highlight the integration of Agile methodologies, such as Scrum, into programming courses. Agile practices emphasise collaboration, adaptability, and continuous feedback, making them well-suited for soft skills development (Beck et al., 2001).

Students participating in Agile-based courses engage in activities such as daily stand-up meetings, sprint planning, and retrospectives. These activities require clear communication, teamwork, and time management. They also provide opportunities for reflection and continuous improvement.

The adoption of industry-oriented practices enhances the authenticity of learning experiences and prepares students for professional environments. It also reinforces the importance of soft skills, as students experience firsthand their impact on project success.

5.6 Challenges and Barriers to Integration

Despite the positive outcomes, the analysis identifies several challenges that hinder the effective integration of soft skills:

Curriculum Constraints: Programming courses are often densely packed with technical content, leaving limited time for soft skills activities (Shuman et al., 2005).

Assessment Difficulties: Evaluating soft skills remains a significant challenge due to their subjective nature. Traditional assessment methods may not capture the complexity of these skills (Topping, 1998).

Faculty Preparedness: Instructors may lack the training or experience needed to effectively teach and assess soft skills, leading to inconsistent implementation.

Student Resistance: Some students prioritise technical skills and may perceive

soft skills as less important, resulting in reduced engagement.

Overall, the findings indicate that soft skills integration in programming courses is most effective when it is embedded within authentic, collaborative, and experiential learning activities. The alignment of pedagogical practices with theoretical frameworks enhances both technical and soft skills development. However, addressing challenges related to curriculum design, assessment, and faculty training is essential for sustainable implementation.

6. Discussion

The findings of this study provide significant insights into the pedagogical practices used to integrate soft skills into programming courses and their broader implications for engineering education. This section interprets these findings in relation to the theoretical framework and existing literature, while also exploring implications for curriculum design, teaching practices, and policy development.

6.1 Reconceptualising Programming Education

The integration of soft skills into programming courses reflects a broader shift in educational paradigms, from a focus on technical knowledge acquisition to a more holistic approach that emphasises professional competencies. This shift aligns with constructivist and experiential learning theories, which view learning as an active, contextualised process (Piaget, 1970; Kolb, 1984).

The findings suggest that programming education should be reconceptualised as a socio-technical discipline, where technical and interpersonal skills are developed simultaneously. This perspective challenges traditional teaching methods that isolate technical content from real-world contexts.

6.2 Alignment with Theoretical Framework

The study's findings strongly support the integrated theoretical framework:

- **Constructivism:** Collaborative programming and PBL exemplify active knowledge construction through interaction and problem-solving (Vygotsky, 1978).
- **Experiential Learning:** Hands-on projects and reflective practices align with Kolb's learning cycle, emphasising experience and reflection (Kolb, 1984).
- **Social Learning:** Peer assessment and group work highlight the importance of observation and interaction (Bandura, 1977).

This alignment underscores the importance of designing learning environments that are interactive, experiential, and socially rich.

6.3 Implications for Curriculum Design

The findings highlight the need for curriculum reform that integrates soft skills into core programming courses rather than treating them as separate modules. This can be achieved through:

- Embedding collaborative activities in assignments
- Incorporating real-world projects
- Aligning learning outcomes with both technical and soft skills

Such integration ensures that soft skills are developed in context, enhancing their relevance and applicability.

6.4 Rethinking Assessment Practices

Assessment emerges as a critical challenge in soft skills integration. Traditional assessment methods, which focus on individual performance and technical correctness, are insufficient for evaluating soft skills.

The study suggests the adoption of alternative assessment strategies, including:

- Rubrics that capture communication and teamwork
- Peer and self-assessment
- Reflective portfolios

These methods provide a more comprehensive evaluation of student learning and align with constructivist principles.

6.5 Role of Instructors as Facilitators

The findings emphasise the evolving role of instructors from knowledge transmitters to facilitators of learning. Instructors must create environments that encourage interaction, provide guidance, and support student development.

This requires professional development and training in pedagogical strategies for soft skills integration. Without such support, instructors may struggle to implement these approaches effectively.

6.6 Addressing Student Perceptions

Student resistance to soft skills integration highlights the need for awareness and motivation. Educators must clearly communicate the relevance of soft skills to career success and provide opportunities for students to experience their value.

Industry collaborations, guest lectures, and real-world projects can help bridge the gap between academic learning and professional expectations.

6.7 Policy and Institutional Implications

At the institutional level, policies should support the integration of soft skills through curriculum guidelines, accreditation standards, and resource allocation. Accreditation bodies such as ABET already emphasise professional skills, but further efforts are needed to ensure their effective implementation. Institutions should also invest in faculty development and provide support for innovative teaching practices.

6.8 Contribution to Knowledge and Future Research

This study contributes to the literature by providing a comprehensive synthesis of pedagogical practices for soft skills integration in programming education. It highlights the importance of aligning theory and practice and identifies key challenges and opportunities.

Future research should explore:

- Empirical validation through primary data
- Longitudinal studies on career outcomes
- The impact of cultural and institutional contexts

The integration of soft skills into programming courses is not merely an educational trend but a necessity in the evolving digital landscape. By adopting holistic, theory-informed approaches, educators can prepare students to succeed in complex, collaborative environments.

7. Conclusion

The findings of this study underscore the growing importance of integrating soft skills into programming courses within Computer Science and Engineering (CSE) education. As the software industry continues to evolve toward collaborative, interdisciplinary, and agile work environments, the demand for graduates who possess both technical expertise and strong interpersonal competencies has become increasingly evident. This research demonstrates that programming education must move beyond its traditional focus on technical proficiency to embrace a more holistic approach that fosters communication, teamwork, problem-solving, and adaptability.

Through a qualitative multiple case study analysis based on secondary data, the study identifies several effective pedagogical practices that support soft skills integration. These include collaborative programming techniques such as pair programming, project-based learning (PBL), peer code reviews, reflective practices, and the incorporation of Agile methodologies. These approaches align closely with constructivist, experiential, and social learning theories, emphasising active engagement, experiential practice, and social interaction as key drivers of learning. When implemented effectively, such practices not only enhance technical understanding but also cultivate essential soft skills that are critical for professional success.

However, the study also highlights significant challenges that must be addressed to achieve meaningful integration. Curriculum constraints often limit the time available for soft skills development, while the subjective nature of soft skills makes assessment complex and inconsistent. Additionally, many instructors lack the necessary training to effectively facilitate and evaluate soft skills, and some students may initially undervalue their importance. Addressing these challenges requires a coordinated effort at multiple levels, including curriculum redesign, faculty development, and the adoption of innovative assessment strategies.

Importantly, the study suggests that soft skills should not be treated as supplementary components but rather as integral elements of programming education. Embedding these skills within core technical courses ensures that they are developed in authentic contexts, enhancing their relevance and applicability. Furthermore, aligning educational practices with industry expectations can help bridge the gap between academic preparation and professional requirements.

In conclusion, the integration of soft skills into programming courses is not merely an educational enhancement but a necessity in the contemporary digital landscape. By adopting theory-informed, student-centred, and practice-oriented pedagogical approaches, CSE departments can better prepare graduates to navigate the complexities of modern software development and contribute effectively to the global workforce. Future research should focus on empirical validation through primary data and explore the long-term impact of such integration on career outcomes and professional growth.

References

Andrews, J., & Higson, H. (2008). Graduate employability, 'soft skills' versus 'hard' business knowledge: A European study. *Higher Education in Europe, 33*(4), 411–422.

Bandura, A. (1977). *Social learning theory*. Prentice Hall.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., & Grenning, J. (2001). *Manifesto for Agile Software Development*.

Begel, A., & Simon, B. (2008). Novice software developers, all over again. In *Proceedings of the Fourth International Workshop on Computing Education Research* (pp. 3–14).

Bell, S. (2010). Project-based learning for the 21st century: Skills for the future. *The Clearing House, 83*(2), 39–43.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3*(2), 77–101.

Creswell, J. W., & Poth, C. N. (2018). *Qualitative inquiry and research design: Choosing among five approaches* (4th ed.). Sage.

Heckman, J. J., & Kautz, T. (2012). Hard evidence on soft skills. *Labour Economics, 19*(4), 451–464.

Johnson, D. W., & Johnson, R. T. (2009). An educational psychology success story: Social interdependence theory and cooperative learning. *Educational Researcher, 38*(5), 365–379.

Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development*. Prentice Hall.

Lethbridge, T. C. (2000). What knowledge is important to a software professional? *Computer, 33*(5), 44–50.

Mannan, K.A., & Farhana, K.M. (2026). *The Principles of Qur'anic Research Methodology: Deriving the Process of Knowledge from Revelation.* KMF Publishers. Open Access (CC BY 4.0). DOI: https://doi.org/10.64907/xkmf.book.pqrm.26.02.12

Patton, M. Q. (2002). *Qualitative research and evaluation methods* (3rd ed.). Sage.

Piaget, J. (1970). *Science of education and the psychology of the child*. Orion Press.

Robles, M. M. (2012). Executive perceptions of the top 10 soft skills needed in today's workplace. *Business Communication Quarterly, 75*(4), 453–465.

Shuman, L. J., Besterfield-Sacre, M., & McGourty, J. (2005). The ABET professional skills: Can they be taught? Can they be assessed? *Journal of Engineering Education, 94*(1), 41–55.

Thomas, J. W. (2000). A review of research on project-based learning. *Autodesk Foundation*.

Topping, K. (1998). Peer assessment between students in colleges and universities. *Review of Educational Research, 68*(3), 249–276.

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Harvard University Press.

Williams, L., & Kessler, R. (2003). *Pair programming illuminated*. Addison-Wesley.

Yin, R. K. (2018). *Case study research and applications: Design and methods* (6th ed.). Sage.