The Journal for Transdisciplinary Research in Southern Africa
ISSN: (Online) 2415-2005, (Print) 1817-4434

Understanding the significance of Software Development Communities of Practice: A systematic review

Author:
Alfred H. Mazorodze[1]

Affiliation:
[1]Software Development Department, Faculty of Information Technology, The Belgium Campus iTversity, Pretoria, South Africa

Corresponding author:
Alfred Mazorodze,
mazorodzeah@yahoo.com

Dates:
Received: 05 June 2024
Accepted: 23 Aug. 2024
Published: 28 Apr. 2025

How to cite this article:
Mazorodze, AH. Understanding the significance of Software Development Communities of Practice: A systematic review. J transdiscipl res S Afr. 2025;21(1), a1475. https://doi.org/10.4102/td.v21i1.1475

Communities of Practice (CoPs) are groups of people who voluntarily share their knowledge and skills with others through continuous interaction. In the software development domain, CoPs are groups of software developers who share knowledge, experiences, ideas and best practices in coding, irrespective of the programming language. Knowledge sharing among software developers is very difficult. Considering the complexity of software development, most software projects succeed if the teams form Software Development Communities of Practice (SD CoPs). The study identifies the platforms used for knowledge sharing, the benefits of knowledge sharing and recognises the challenges faced by software developers. The study utilised the Preferred Reporting Items for Systematic Reviews and Meta-Analysis (PRISMA) technique to search, identify and filter articles over a 5-year period. The articles reviewed were drawn from accredited journals published between 2019 and 2024 and indexed in academic databases. The emergent themes included collaboration, effective communication, and professional development of the software developers. The study established that SD CoPs bring diverse skills set and expertise coupled with effective problem solving, a critical skill expected from every software developer. Code repositories, code review tools and social media platforms are used for effective knowledge sharing in these communities. These SD CoPs facilitate peer code reviews, an important process that improves code quality and promotes adherence to international coding standards.

Transdisciplinary contribution: The article contributes to the practical implementation of CoPs, particularly focusing on effective knowledge sharing and collaboration within organisations. Knowledge sharing is a crucial factor that drives innovation in organisations.

Keywords: Software Development Communities of Practice; knowledge sharing; software developer; best practices; coding standards; organisational learning.

Introduction

Communities of Practice (CoPs) are defined as groups of people who come together to learn from one another, share experiences, and develop their expertise.[1] These groups of people share a common concern in problem solving and deepening their knowledge. Multiple scholars[1,2,3] concur that CoPs provide platforms not only for knowledge sharing but also for skills development. In the field of software development, CoPs are groups of software developers who voluntarily share insights, tools, coding standards and experiences as they develop software for modern use. Communities of Practice can be considered as sources of innovation and also as mechanisms for fostering knowledge sharing. Across multiple organisations and teams, knowledge sharing is a voluntary process[4] deeply rooted in the willingness of the individuals to contribute their experiences and expertise to others. The inquiry prompted by this voluntary procedure is: 'What measures can be taken into consideration to encourage software developers to willingly share their knowledge?' An understanding of these measures could pave way for knowledge sharing among software developers, in different organisations.

Software refers to a set of systematically programmed instructions that solve a given task[5] in a specific programming language. Most of today's business problems are solved with robust and user-friendly software. Some examples of modern computer programming languages include Java, C#, JavaScript, Ruby and Python among others. The process of developing this robust software therefore calls for trained and experienced software developers, who must continuously share their expertise with others in their communities. The aim of this study is to provide insights that could guide the formation of Software Development CoPs. Because software development is a knowledge-intensive task,[6] this study systematically reviews literature guided by the following research questions:

- *Which platforms are used for knowledge sharing within Software Development Communities of Practice (SD CoPs)?*
- *To what extent do SD CoPs impact the coding standards and best practices?*
- *What are some of the challenges faced by software developers in SD CoPs?*

Software development process requires the application of tacit and explicit knowledge. On one hand, tacit knowledge refers to the knowledge held by the individuals and this knowledge is difficult to share[7] because it is placed inside the software developer's mind and cannot be totally revealed. One of the strategies to tap into tacit knowledge is to learn through observation. Sharing tacit knowledge requires trust between the knowledge holder and the receiver of that knowledge. A lack of trust becomes an obstacle, especially in the software development field, which is the focus of this study. On the other hand, explicit knowledge is documented and is publicly available. Explicit knowledge is codified, documented, and can be easily communicated through a formal language and appropriate symbols.[7] As knowledge is shared through social interaction, there is need for creating SD CoPs to share this tacit knowledge so that software projects are delivered successfully.

The study therefore sought to:

- understand the platforms used for knowledge sharing in SD CoPs;
- assess the impact of SD CoPs on coding standards and best practices;
- explore different challenges faced by software developers in SD CoPs.

The theoretical background and other important concepts are presented next to give us a more concrete understanding of CoPs and their significance in software development.

Theoretical background

In the software development domain, CoPs are groups of software developers who share experiences, ideas, coding standards and best practices, which are important for successful software project delivery. As described by Alsaqqa et al.,[5] software refers to a set of instructions that solve a particular problem. According to Alsaqqa et al.,[5] software is classified into two categories: systems software and applications software. Despite this important classification, software development involves collaboration among software developers, system designers, software testers, and other important stakeholders to ensure the quality, functionality, and usability of the final software product. Literature suggests that a good software application should be reliable, scalable, secure, efficient, flexible and compatible to other systems.[8] All these important features require diverse skills set from different software developers, hence the need to form and embrace SD CoPs for effective communication and continuous learning. In most organisations, CoPs can be formally established or can evolve naturally.[9] Communities of Practice provide access to new knowledge, generates new knowledge, and encourages the much-needed skills development, especially in the software development domain.

A CoPs is made up of three aspects: the community, the domain, and the practice. A community describes a group of people with a common interest[8] and a shared identity. These people continuously interact, collaborate, learn, and foster a sense of belonging through socialisation. The domain aspect represents the area of expertise around which the community is organised, which in this case is software development. Authors like Choi et al,[1] add that the domain provides the context for the community's activities. Similarly, domain describes the common ground around which the community revolves. This common interest unites the members of the community and provides these members with insights and best practices through interactions. Lastly, practice encompasses the different processes, techniques, and methodologies that members engage in within their community.[9] These practices are important to develop their skills and knowledge related to their domain. Some of these practices may include problem-solving strategies, tools and best practices that are shared collaboratively within the community, and this requires trust.[10] In addition to that, some of the best practices in software development include the writing of modular code, automated testing processes, code reviews and performance optimisation among others.

Considering that software development is a knowledge-intensive task,[11] software developers are highly likely to benefit from these communities. A task that is knowledge-intensive demands a substantial level of specialised knowledge and expertise for successful execution. These types of tasks typically entail making informed decisions, solving problems, and employing creative thinking. Software development is mostly conducted in the form of projects[12] supervised by a project manager and from an organisational perspective, SD CoPs serve as vehicles for organisational learning.[9] Documenting the lessons learnt, success stories and best practices collectively contribute to the organisational knowledge base. Lessons learnt from software development are crucial for reducing risks, enhancing code quality, and ultimately ensuring the successful delivery of projects that satisfy both technical specifications and user needs. Lwakatare[13] submits that SD CoPs are very important for innovation, a process which is very crucial for any organisation's success. The author argues that, through meetings and brainstorming sessions, members can explore new approaches and methodologies to problem-solving in their communities. In addition, SD CoPs facilitate the exchange of expertise and best practices among software developers, and this accelerates individual learning and promotes collective intelligence of the entire community.

Software development is a knowledge-intensive process,[14] meaning multiple challenges are encountered within the software development teams. To begin with, software developers prefer to work in isolation and this results in knowledge being confined within certain software developers. Working in isolation hinders opportunities for growth and improvement. According to Weerasekara and Smedberg,[9] the formation of SD CoPs could close this gap of silos and team isolation. Communication within the software development teams is also another challenge that impedes effective knowledge sharing[15] in organisations. There is therefore an urgent need for clear and open channels of communication among software developers. Moving on the same direction, staff turnover has also been reported in literature[16] as a problem because knowledge is immediately lost when team members leave the project. Therefore, it is important that the knowledge is tapped into during the process of software development, an important reason why organisations should form SD CoPs. Other constraints that hinder knowledge sharing include the lack of time,[2] as many organisations prioritise project delivery over knowledge sharing and this is very common in software development. Fostering a culture of collaboration and knowledge sharing may address some of the challenges experienced in SD CoPs and this may require a multifaced approach.

Organisational and cultural factors also affect the successful delivery of software projects.[17] In an organisation where there is competition among software developers, the aspect of knowledge sharing is very limited, if any. Furthermore, the rigid organisational structures discourage knowledge sharing and collaboration.[18] Along the same spectrum, team members who are less motivated will not share their expertise with others. Rewarding excellence and knowledge sharing is encouraged; thus, the issue of executive support comes into play. Executive support is necessary to create an environment where knowledge sharing thrives.[18] Moreover, executive support sets priorities, removes barriers, overcomes resistance to change and ensures the success of all knowledge sharing initiatives. The challenges with documentation also pose significant problems in software development. The process of documentation is time-consuming, and this hinders the transfer of knowledge making it difficult for new team members to learn effectively. The methodology adopted to conduct this study is presented next.

Methods

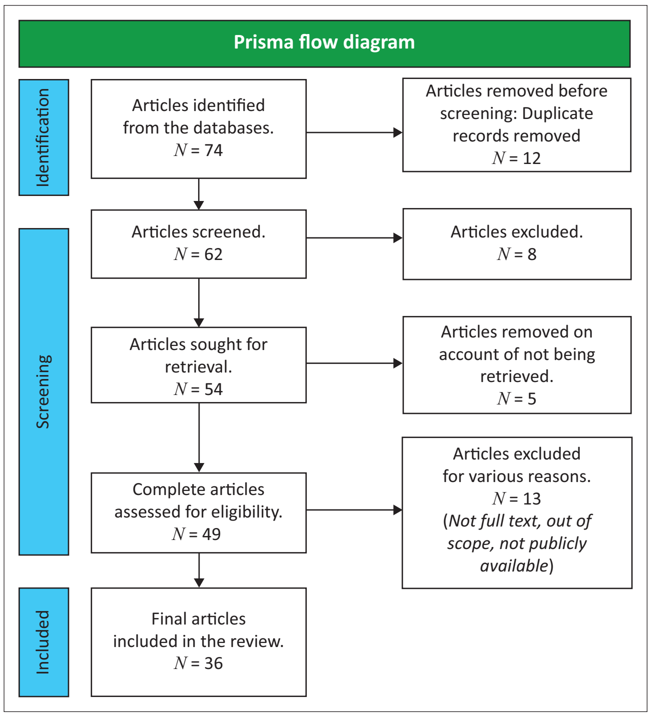
To provide a complete coverage and evidence-based decision making, the study conducted a Systematic Literature Review (SLR) to evaluate and analyse articles related to CoPs and software development. The SLR follows a pre-defined set of steps, as defined by Kitchenham[19] and this distinguishes the process from the unstructured reviews. The SLR offers a good approach to integrate existing research findings. The process followed in this study is therefore presented next, starting with the search strategy.

Search strategy and keywords

This study sought to understand the importance of SD CoPs from an organisational perspective. To assess this significance, Google Scholar, Science Direct, Web of Science and Elsevier databases were employed to assemble and integrate research published using the Preferred Reporting Items for Systematic Reviews and Meta-Analysis (PRISMA) principles. Academic databases offer a comprehensive coverage and access to scholarly work, which is of high quality.[20] Some of the keywords used in the search criteria included 'Software Development', 'Communities of Practice', 'Software Developer' 'projects' and 'knowledge sharing'. The logical operators such as the 'AND' and 'OR' were applied to create effective queries. The general SLR process is illustrated in Figure 1:

Inclusion criteria

Taking into account the rate of technological advances in the field of software development, reporting on contemporary research is important. The study only analysed research articles published between 2019 and 2024 in accredited journals and indexed in renowned databases to ensure that research remains relevant and accurate. To guarantee high quality research, all articles and conference papers used in this study were peer-reviewed and published in English language. The criteria included both theoretical and empirical studies with either quantitative and/or qualitative methods. Blending insights from both methodologies harnesses



Prisma flow diagram
Identification
Screening
Included
Articles identified from the databases. N = 74
Articles removed before screening: Duplicate records removed N = 12
Articles screened. N = 62
Articles excluded. N = 8
Articles sought for retrieval. N = 54
Articles removed on account of not being retrieved. N = 5
Complete articles assessed for eligibility. N = 49
Articles excluded for various reasons. N = 13 (Not full text, out of scope, not publicly available)
Final articles included in the review. N = 36

Source: Kitchenham B, Brereton P. A systematic review of systematic review process research in software engineering. Inform Softw Technol. 201355(12):2049–2075

PRISMA, Preferred Reporting Items for Systematic Reviews and Meta-Analysis.

FIGURE 1: Search process flow diagram.

the advantages of each, resulting in a more holistic comprehension of the research topic.

Exclusion criteria

To ensure the inclusion of only relevant and appropriate studies in the analysis, textbooks, theses, and dissertations were excluded. The study utilised the PRISMA framework to screen different data sources. In addition, studies not written in English and non-peer-reviewed work were excluded. Non-peer-reviewed articles, often considered opinions, were omitted to maintain the rigour of the study.

Data extraction

The data extraction procedure was based on the research methodology utilised. The data extracted include the objectives and the results. The data mined from these articles sought to completely understand knowledge sharing in CoPs, specifically in the software development field.

Data synthesis and analysis

Microsoft Office Excel was used to identify trends and patterns in SD CoPs that emerged from the articles selected. As a result of the nature of research, data were analysed in a storytelling format. Storytelling in research reporting transforms data into exciting narratives, making research more engaging, understandable, and impactful.

Ethical considerations

The article reports on secondary data, there was no ethical clearance required. However, the researcher observed all the ethical principles in reporting the findings. All sources used are shown with in-text citations and there is a complete reference list at the end of the article to acknowledge other people's work. This article followed all ethical standards for research without direct contact with human or animal subjects.

Review findings and discussion

After the screening and selection process, a total of 36 articles were considered for further analysis. To maintain relevance, the bar graph below shows the distribution of the selected articles per year, from 2019 to 2024. A bar graph is simply a visual representation of data with bars of varying lengths used to show the frequency, and distribution of different categories. As stated earlier on in the inclusion criteria, different research approaches were used in all these articles (Figure 2).

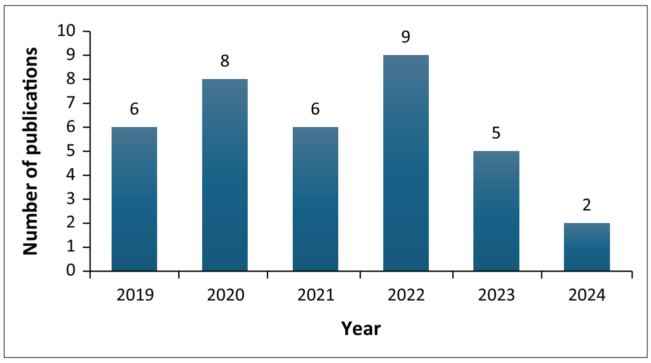


FIGURE 2: Distribution of research articles per year (2019–2024).

Multiple themes emerged from these articles, but this study was focused on understanding CoPs, specifically for knowledge sharing in software development. In the realm of software development, new tools, frameworks and methodologies are continuously evolving, hence the need to understand these in detail. Frameworks support software developers by offering pre-existing solutions to typical challenges, reducing the necessity for extensive reworking. Thus, the use of frameworks facilitates effective software application development. Appendix 1 shows all the publications included in this SLR.

Given the objectives of the study and based on the summary of the studies ($N = 36$) included in the analysis, only three areas of interest were formed and categorised as follows:

1. Knowledge sharing and the knowledge sharing platforms ($n = 14$)
2. Coding practices and best practices ($n = 10$)
3. Knowledge sharing challenges ($n = 12$).

A knowledge sharing platform facilitates the exchange of knowledge among software developers. Some of the key features of knowledge sharing platforms include content creation tools, collaboration features and identification of experts. Comprehending the platforms used for knowledge sharing in SD CoPs is important.

Platforms for knowledge sharing in Software Development Communities of Practice

Multiple platforms are adopted for knowledge sharing in CoPs.[21] From a software development perspective, these important tools and platforms facilitate effective communication among software developers. According to Jadhav et al.,[22] version control systems are used for knowledge-sharing in software development teams. Basically, some of the knowledge sharing platforms include social media, online platforms and Knowledge Management Systems (KMS). However, in the SD CoPs, specific tools such as version control systems, code repositories and code review tools are used for effective knowledge sharing.[22] Wiki platforms are tools used for project communication and collaboration within the software development teams.[23] These software tools allow software developers to create, edit, and organise content collaboratively. Wiki platforms serve as knowledge repositories where software development teams can document their project requirements,[24] coding standards and the associated best practices.

Software development involves the process of writing instructions to solve a problem, commonly referred to as 'coding'. Code repositories or version control systems are used to track changes during code development.[6] The code repositories can either be centralised or distributed, with a core function of storing and managing source code files and their revisions. From the submissions by Jadhav et al.,[22] code repositories serve as a central hub for developers to collaborate on different projects. Centralised version control differs from distributed version control. In a centralised system exists a solitary, central repository acting as the definitive origin for the project's codebase, whereas in a distributed setup, each developer possesses a replica of the repository, encompassing its entire history, on their individual machines.[5] Some of the core features of code repositories include versioning, branching, merging and permitting developers to work on the same codebase simultaneously. It was established in this study that Git, GitHub, BitBucket and GitLab are gaining popularity in the software development field.[25,26] Code review tools play a very crucial role in maintaining code quality, making it easier to maintain and extend. Moreover, the code review tools permit knowledge sharing among software developers working on the same or different projects.

Combining knowledge, methodologies, and perspectives from various disciplines is crucial for effective problem-solving. Within the SD CoPs, this interdisciplinary approach is reflected in the diverse tools and platforms deployed. For instance, version control systems, fundamentally a computer science concept, are vital for managing code changes. Insights from information science enhance this technical aspect by highlighting the significance of KMS. Additionally, perspectives from social sciences provide an understanding of the human and organisational factors that impact team collaboration. Through the integration of these various methodologies and viewpoints, the SD CoPs can create effective strategies that address technical challenges while fostering an environment that promotes innovation and continuous improvement in software development. Table 1 shows the key findings from the studies.

Table 1 shows that various authors have used different approaches to study knowledge sharing platforms in organisations. Combining these approaches enhances study effectiveness by providing a comprehensive understanding of complex problems through multiple disciplinary perspectives. Technical methods offer precise solutions, while social science methodologies provide insights into human behaviour and organisational dynamics. This integration ensures solutions are technically sound, practical, sustainable, and aligned with organisational needs.

The code review tools further help to identify issues early in the software development process making sure that code changes meet quality standards before being merged into the main codebase. Some of the popular tools identified and reported in this article include pull requests, GitLab merge requests and the Bitbucket pull requests. Pull requests facilitate collaboration and code quality among software developers. Moreover, pull requests helps to ensure adherence to international coding standards. According to Zhang et al.,[28] pull requests can be integrated into the pipelines to automate the testing and deployment processes. Also, pull requests serve as a form of documentation for code changes during system development in teams. Documentation is an integral part of systems development that promotes transparency, compliance, and maintainability of systems. It acts as an essential resource for developers, offering comprehensive details on system architecture, design choices, and code functions.

TABLE 1: Sample of knowledge sharing platforms used in organisations.

Title of article	Author	Approach	Key findings
Understanding community participation and engagement in open-source software projects: A systematic mapping study	Kaur et al.[27]	Quantitative with a survey	• Developers and users form communities in projects as they share an interest • Community participation requires motivation
The use of online communities of practice in the software industry: learnings from Facebook communities in Sri Lanka	Weerasekara and Smedberg[9]	Qualitative, with thematic analysis	• Software engineers engage in different practices to support each other in the communities • Software professionals collaborate in cyberspace using different tools
How do software developers use GitHub actions to automate their workflows?	Kinsman et al.[25]	Qualitative	• GitHub provides automated workflows • There is a positive perception of the technology
Wiki use for knowledge integration and learning: A three-tier conceptualisation.	Luo and Chea[23]	Quantitative survey	• Social interaction with teammates improves the learning experience • Wiki platforms are used for knowledge sharing and collaboration • Social presence is very important for learning
Social media for knowledge-sharing: A systematic literature review	Ahmed et al.[21]	Qualitative systematic review	• Social media can be used for knowledge sharing • Social media facilitates knowledge sharing, not only for individuals but for organisations too

Social media has just become pervasive in our day to day lives and in the software development business too. The social media platforms permit collaboration, learning, and career advancement within the software development community.[29] Software developers can easily leverage these platforms to stay updated and grow professionally.[15] In addition, social media platforms provide real-time updates on emerging technologies, best practices and other trends in software development. Thus, software developers stay informed about the latest advancements and adapt their skills accordingly. Social media serves as a significant asset within the software development domain, offering avenues for networking, teamwork and continuous learning. Effective utilisation of social media allows software developers and organisations to broaden their connections and remain updated on the most recent advancements in the technological industry. Figure 3 summarises the different tools and technologies, which are useful for knowledge sharing in SD CoPs.

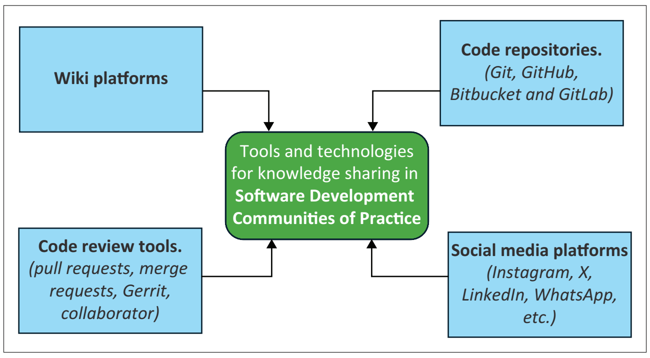

Wiki platforms
Code repositories. (Git, GitHub, Bitbucket and GitLab)
Tools and technologies for knowledge sharing in Software Development Communities of Practice
Code review tools. (pull requests, merge requests, Gerrit, collaborator)
Social media platforms (Instagram, X, LinkedIn, WhatsApp, etc.)

FIGURE 3: Tools and technologies for knowledge sharing in Software Development Communities of Practice.

The role of all platforms discussed in this review is to facilitate effective knowledge sharing and collaboration among software developers. These technologies have an important impact on the dynamics of SD CoPs, and they call for the implementation of KMS. In literature,[18] KMS are designed to harness and leverage the intellectual capital of an organisation and enable it to become innovative, competitive and agile. These organisational KMS play a pivotal role in fostering collaboration, learning, and innovation within the SD CoPs. The systems empower software developers to utilise collective expertise, leverage shared knowledge and ultimately tackle intricate challenges in their communities as alluded by Lwakatare et al.[30] Knowledge Management Systems further provide different mechanisms for knowledge capture, including documents, reports, best practices, lessons learnt and tacit knowledge. Capturing tacit knowledge involves informal conversations, storytelling,[18] CoPs and documenting the best practices. Although tacit knowledge poses challenges for documentation, organisations can capture elements of it by recording best practices, insights gained from experiences, and instances of success. The study further sought to assess the impact of SD CoPs on the coding standards and best practices as elaborated in the subsequent section.

Impact of Software Development Communities of Practice on coding standards and best practices

Coding standards are defined by Pargaonkar[31] as a set of guidelines that dictate how code should be written within a software development project. Adherence to the coding standards significantly help software developers to write consistent code, which is easier to read, maintain and debug. Thus, software developed according to the coding standards is easier to maintain and understand. The SD CoPs influence coding standards and best practices.[32] Such influence is seen in terms of the provision of resources for peer review, standardisation and continuous improvement among others. The SD CoPs facilitate peer code reviews where software developers can receive feedback on their code from peers.[33] As mentioned earlier on, this important process improves quality of code and promotes adherence to international coding standards and best practices. Feedback loops within these SD CoPs encourage software developers to align their coding styles with well-established standards.

Standardisation of code is an important process in software development.[34] Promotion of coding standards and best practices ensures that the code is easy to understand, extend and maintain. Furthermore, SD CoPs can help to foster a culture of continuous improvement where software developers are kept abreast with emerging trends and technologies in the field. It is worth noting that SD CoPs provide different forums for discussing the coding standards. Voluntary participation in SD CoPs therefore contributes to collective wisdom that elevates the quality of projects developed by the software developers. Engaging voluntarily in SD CoPs enhances the software developers' professional experiences, creative thinking, and this also aids in the overall progress and development of the software industry. Knowledge sharing takes place in these voluntary discussions with peers.

By default, CoPs are meant for knowledge sharing and knowledge transfer.[14] Knowledge transfer is very important for professional development of the software developers. Various mechanisms inclusive of socialisation, shared resources, collaborative projects and informal learning are all used for knowledge sharing and transfer during software development. Membership penetration and growth are key items, which should be analysed, especially in SD CoPs to see how the community has grown.[1] As knowledge sharing is a voluntary process, SD CoPs are therefore a powerful manifestation of informal learning. Thus, SD CoPs are effective as they enable software developers to manage change, especially in the general Information Technology (IT) field. Lastly, the study examines some of the challenges faced by software developers in their SD CoPs and the review findings are presented and discussed next.

Challenges faced by software developers in Software Development Communities of Practice

Software development is a continuously evolving field, which makes use of various tools, technologies and platforms. Multiple trends and practices are continuously emerging and software developers should be kept abreast of these trends to avoid software defects[35] in their SD CoPs. This emergence has both negative and positive impacts on the effectiveness of these communities. Achieving success in software development endeavours is notably challenging. Arguably, one of the primary hurdles in such projects lies in understanding how to improve software development processes to prevent failure.[36] Some of the issues which affect the application of tacit knowledge in SD CoPs include the lack of trust, lack of time, as well as the context of knowledge.[24] From the given submissions, the study can confirm that trust is very important in SD CoPs. Trust forms the foundation of SD CoPs, creating a space where members can confidently exchange knowledge, work together, and explore new ideas. By cultivating trust through elements such as respect, expertise, teamwork, feedback, responsibility, and conflict resolution, these communities foster a supportive and welcoming environment where emerging software developers can flourish and ultimately achieve their goals. The lack of trust among the software developers negatively affects the success of the project. Moreover, the lack of time is a great challenge in SD CoPs in the sense that projects have very strict deadlines.

Delivering a project on time in software development requires careful planning, effective communication, skilled team collaboration, and proactive risk management. Implementing agile methodologies, setting clear and realistic deadlines, effective communication and collaboration are important strategies to successfully deliver a project. Thus, by implementing these strategies and best practices, software development teams can surely improve their chances of delivering projects on time while meeting quality standards and stakeholder expectations. It is one of the critical success factors for measuring project success; thus, the project must be delivered on time, within the allocated budget to the satisfaction of the client. As SD CoPs are knowledge-intensive in nature,[36] the source of knowledge together with the context of knowledge affects the success of the SD CoPs. Some of the critical factors contributing to the success of on-going agile software development projects include customer involvement and team capabilities.[36] It is therefore important to understand that the capabilities of a team can be improved by forming and effectively collaborating in SD CoPs. These communities should work together to deliver a successful software product, which satisfies the customer on time.

Some of the critical challenges faced by software developers in SD CoPs include staff turnover, knowledge hoarding and resistance to change. Knowledge hoarding describes the act of individuals within an organisation to deliberately withhold expertise from others and this behaviour hinders collaboration within an organisation. Knowledge hoarding often occurs when individuals perceive that sharing their knowledge might diminish their own importance, and this normally manifests in different forms such as not documenting the processes or not mentoring others. Software developers generally find it more convenient to continue doing something as they have always been doing. Resistance to change requires proactive communication and appropriate change management strategies. Together, researchers and practitioners should understand these challenges and implement the best strategies and practices to mitigate these challenges.

Transdisciplinary contribution

Firstly, the SD CoPs facilitate knowledge transfer and continuous learning. The insights from SD CoPs can inform best practices in educational settings, particularly in teaching programming, software development and collaborative problem-solving skills. Secondly, an understanding of how SD CoPs facilitate knowledge sharing extends the benefits to other fields focused on organisational learning and Knowledge Management (KM), offering different models for effective knowledge transfer and knowledge retention. The SD CoPs further contributes to the organisational behaviour and management fields, through team collaboration. Collaborative work environments in software development can be optimised, providing strategies for enhancing team dynamics, leadership, and conflict resolution in various organisational contexts. The SD CoPs also contribute towards innovation management and the findings can be applied to manage the innovation processes in different sectors, encouraging creativity and continuous improvement.

The article further contributes to information systems and technology adoption in organisations. The insights into the tools, technologies and platforms that support SD CoPs can guide the design and adoption of collaborative technologies in other domains. Best practices identified in the SD CoPs can inform the broader software engineering methodologies, improving practices across different software development environments. Lastly, the study contributes to the social sciences field, specifically communication studies. The different ways in which SD CoPs communicate and share knowledge can offer valuable data for studies in communication, including the use of digital communication tools. The subsequent section provides the implications and recommendations of the study from a transdisciplinary perspective.

Implications and recommendations

The SD CoPs have an important impact on the software development processes and practices. Firstly, software development brings together experts from different fields working on a single project supported by appropriate technologies. Secondly, these experts include project managers, systems analysts, system users and software developers themselves among other stakeholders. All these stakeholders will be working together to ensure that the project is delivered on time, within the allocated budget to the satisfaction of the client. Exploring the benefits and challenges of cross-disciplinary collaboration within the SD CoPs is important so that knowledge is shared through the lessons learnt and best practices. The implications discussed in this study are guided by the study objectives, which sought to understand the platforms for knowledge sharing, assess the impact of SD CoPs on coding standards and lastly the discussion on the challenges faced by software developers in these communities. From a transdisciplinary perspective, the implications of SD CoPs are discussed in point form next.

Impact on knowledge sharing and knowledge transfer

The SD CoPs facilitate knowledge sharing and knowledge transfer among software development teams. Knowledge sharing, using different platforms increases knowledge acquisition not only in software development organisations but also in other organisations too. Insights gained from SD CoPs can be applied to the management and education sectors, fostering innovation and collective problem-solving beyond the confines of software development. Furthermore, SD CoPs permit the capturing of tacit knowledge, which is deeply rooted in the specific software developers' minds. Tacit knowledge which is difficult to articulate becomes accessible through collaborative tools and shared practices, benefiting diverse fields such as organisational behaviour. The software tools allow software developers to create, edit, and organise content collaboratively. These tools are not only beneficial within the boundaries of software development but also in other disciplines where collaborative efforts are needed.

Academic researchers and practitioners in industry should emphasise the knowledge sharing mechanisms within the SD CoPs. Knowledge sharing creates a sense of trust within the software development teams, as developers feel valued when their contributions are acknowledged in their respective organisations. This trust is essential in disciplines such as sociology, where the focus is on understanding the dynamics of group behaviour and the organisational culture. Moreover, knowledge sharing facilitates effective decision-making and is the cornerstone of successful organisations driving innovation and competitive advantage. The principles of knowledge sharing, and collaborative practices observed in SD CoPs can inform strategies in other sectors, promoting a culture of continuous improvement to address different challenges.

Effective communication and collaboration

Effective communication and collaboration are very important for the success of any project. Without effective communication and continuous collaboration, the project is highly likely to fail. This principle applies not only to software development but also to various other fields. For example, the different methods used to enhance communication and collaboration in software development can be adapted in project management teams and academic research groups, fostering innovation in diverse contexts. Cross functional communication is therefore highly encouraged across the software development teams. The study recommends that effective communication channels are established within the SD CoPs. In most cases, software developers use digital communication channels for code reviews and the general project management.

Furthermore, organisations should identify all the factors and variables, which could encourage software developers to voluntarily collaborate and share their tacit knowledge. Organisations should foster a collaboration culture for their SD CoPs as a means of enhancing their collective capabilities in the software development process. By leveraging insights from sociology, and communication studies, organisations can create effective strategies that enhance internal and external collaboration. Through executive support, organisations should create open dialogue and brainstorming sessions so that software developers can drive creative solutions to complex coding problems. These practices can inspire similar initiatives in other fields, encouraging innovation and creative problem-solving across various industries.

Technological infrastructure

Researchers should explore the use of collaborative platforms, effective communication tools, and KMS within the SD CoPs. Organisations can easily identify best practices and innovative approaches from various fields to enhance these systems. Software developers are encouraged to use all the publicly available tools and technologies for collaboration and knowledge sharing during project execution. These tools have applications in education and business and they promote a culture of open communication and collective problem-solving. Organisations are therefore requested to invest in suitable technology solutions that facilitate seamless communication, knowledge sharing, and collaboration among SD CoPs members. Investing in technological infrastructure empowers software developers to collaborate effectively and innovate continuously. Based on research in technology adoption and organisational behaviour, these investments can be customised to enhance user engagement and effectiveness. As SD CoPs grow, the technological infrastructure must scale and adapt to the changing needs and the ever-evolving technologies. The study recommends that organisations embrace the power of technology in enabling all the knowledge sharing efforts for software developers. By integrating insights from IT, organisational behaviour, and communication studies, different organisations create comprehensive strategies that support their internal teams and contribute to the wider knowledge ecosystems.

Best practices and coding standards

Coding standards should be prioritised in SD CoPs to develop code, which is easier to maintain and debug. Peer code reviews are an important process that improves quality of code and promotes adherence to international coding standards and best practices. These standards are not just technical, but they derive insights from other fields such as organisational behaviour. When developers adhere to uniform naming conventions and established design patterns, it becomes easier to implement modifications and continuous improvement. Software developers are highly encouraged to align their code with international coding standards in their SD CoPs. Some of the best practices often incorporate security guidelines to help software developers write more secure code. It is further recommended that SD CoPs provide different forums for discussing the coding standards and this is a voluntary process.

The adoption of coding standards aligns with the principles of human-computer interaction (HCI), ensuring that the software is user-friendly and easily accessible. Participation in such SD CoPs therefore contributes to collective wisdom of the software developers, leveraging the diverse expertise within the community. As best practices and coding standards promote consistency, reliability, maintainability and collaboration, it is important that software developers embed these practices in their work. This integration is supported by organisational studies, which show that standardisation leads to higher efficiency and effectiveness. Ultimately, incorporating best practices contributes to the overall success of software development projects.

Professional development

The SD CoPs provide opportunities for continuous professional development, and this development helps to overcome some of the challenges faced by software developers. These are very important skills needed by every software developer, learning a new programming language, a new framework, a new tool and so forth. Insights from the educational psychology suggest that ongoing learning and skill acquisition are key to maintaining cognitive engagement. It is therefore recommended that organisations support and promote the software developers to participate in these SD CoPs. Attending discussions and workshops will enhance the skills of the software developers. Support is backed by organisational behaviour research, which indicates that investments in employee development leads to increased job satisfaction and loyalty. Continuous professional development allows software developers to continually update and expand their coding skills and acquire additional knowledge, which benefits the organisations. In addition, professional development leads to greater motivation and productivity in the workplace. Organisations should further invest in professional development courses to unlock the new opportunities for their software developers. This important investment makes meaningful contributions to the organisations and the communities at large. This investment is not just a cost, but a strategic initiative, as highlighted by human resources management research.

Organisational learning

Continuous improvement and innovation are very important for organisational learning. From the review, SD CoPs offer strategies that can leverage collective intelligence of the software development teams. This collective intelligence is supplemented by insights from social network theory, which emphasises the value of diverse and interconnected knowledge sources in solving complex problems. Organisational learning helps to upskill the capabilities of the software developers enabling them to learn from failures. Embracing failures as learning opportunities can lead to greater innovation and problem-solving skills. Software development organisations should continuously embrace a culture of learning so that they unlock their full potential and grow. Researchers and practitioners should focus on fostering a supportive environment that encourages active participation and collaboration among community members. Social learning theory suggests that such environments are crucial for effective knowledge transfer and innovation. The SD CoPs help to foster collaboration, knowledge sharing as well as professional development among software development teams. Organisations can therefore harness the full potential of SD CoPs to drive innovation and continuous learning in software development.

Conclusion

The SD CoPs are groups of software developers who voluntarily share insights, tools, coding standards and experiences as they develop software for different clients. These communities operate on the principle of collective intelligence: a concept that spans computer science and organisational behaviour. There are many platforms used for knowledge sharing and knowledge transfer in SD CoPs inclusive of version control systems, social media platforms, code repositories and code review tools among others. All these tools and platforms are meant to facilitate effective communication and collaboration among software developers.[27] Social media platforms foster informal communication and community building, as explored in communication studies. These tools and technologies have an impact on the dynamics of SD CoPs, and they call for the implementation of KMS to foster collaboration, continuous learning and innovation. Knowledge Management Systems integrate methodologies from KM information systems ensuring that knowledge is effectively captured and shared. The SD CoPs facilitate peer code reviews where software developers can receive feedback on their code from their peers. The purpose of these code reviews is to improve the quality of the software product and adherence to international coding standards, drawing from quality assurance and standardisation theories. Furthermore, SD CoPs promote standardisation of code, an important process in software development which must be prioritised to show consistency. To be precise, promotion of coding standards and best practices ensures that the code is easy to understand and maintain.

The study further confirms that active participation in SD CoPs contributes to collective wisdom that elevates the quality of software projects developed and implemented. The study established that achieving success in software development endeavours is challenging and these challenges could be solved by forming and utilising SD CoPs. Some of the issues that affect the application of knowledge in SD CoPs include the lack of trust, lack of time and the context of knowledge. The lack of trust among the software developers negatively affects the success of the software development projects. As SD CoPs are knowledge-intensive in nature, the source of knowledge together with the context of knowledge affects success of the software development teams. The study concludes that people factors contribute to the success of software projects in SD CoPs. The study therefore recommends that software developers form SD CoPs as they embark on their software projects. Participation in these SD CoPs is a voluntary process, and different factors and variables should be put in place to encourage effective knowledge sharing and continuous collaboration among software developers. Lastly, software developers should continuously exchange ideas and experiences in their communities so that other software developers can learn in the process, a practice which could be applied to other fields too. Directions for future research: Future research should focus on exploring the trust-building mechanisms in SD CoPs.

Acknowledgements

The author would like to thank the Belgium Campus iTversity for granting him time to conduct this systematic literature review. The author further thanks the language editor for reviewing the article so that it conforms to the academic standards. Many thanks are also accorded to all those who assisted by reading drafts before the final submission was accepted for publication. Your inputs and comments are highly valued.

Competing interests

The author declares that they have no financial or personal relationship that may have inappropriately influenced them in writing this article.

Author's contributions

A.H.M. is the sole author of this review article.

Funding information

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data availability

The article reports on secondary data from published articles. This data are publicly available and copies of the same can be requested from the corresponding author, A.H.M.

Disclaimer

The views and opinions expressed in this article are those of the author and are the product of professional research. It does not necessarily reflect the official policy or position of any affiliated institution, funder, agency, or that of the publisher. The author is responsible for this article's results, findings, and content.

References

1. Choi HJ, Ahn JC, Jung SH, Kim JH. Communities of practice and knowledge management systems: Effects on knowledge management activities and innovation performance. Knowl Manag Res Pract. 2020;18(1):53–68. https://doi.org/10.1080/14778238.2019.1598578
2. Bicchi F. Communities of practice and what they can do for International Relations. Rev Int Stud. 2022;48(1):24–43. https://doi.org/10.1017/S0260210521000528
3. Castaneda DI, Cuellar S. Knowledge sharing and innovation: A systematic review. Knowl Process Manag. 2020;27(3):159–173. https://doi.org/10.1002/kpm.1637
4. Tabatabaee M, Sharma R. Social networks and knowledge sharing in communities of practice: A developing societies' perspective. Turkish J Comput Math Educ [serial online]. 2021;12(13):5626–5633. [cited 2023 Jan 20] Available from: https://www.turcomat.org/index.php/turkbilmat/article/view/9807
5. Alsaqqa S, Sawalha S, Abdel-Nabi H. Agile Software Development: Methodologies and trends blockchain technologies view project social emdia networks view project. Artic Int J Interact Mob Technol. 2020;246–270. https://doi.org/10.3991/ijim.v14i11.13269
6. Chen L. AI-enabled cloud platforms: Revolutionizing Software Development. Asian Am Res Lett J 2024;1(1):1.
7. Castellani P, Rossato C, Giaretta E, Davide R. Tacit knowledge sharing in knowledge-intensive firms: The perceptions of team members and team leaders. Rev Manag Sci. 2021;15(1):125–155. https://doi.org/10.1007/s11846-019-00368-x
8. Alsanad AA, Chikh A, Mirza A. A Domain ontology for software requirements change management in global software development environment. IEEE Access. 2019;7:49352–49361. https://doi.org/10.1109/ACCESS.2019.2909839
9. Weerasekara M, Smedberg AB. The use of online communities of practice in the software industry: Learnings from Facebook communities in Sri Lanka. Int J Web Based Communities. 2022;18(2):186–209. https://doi.org/10.1504/IJWBC.2022.124782
10. Wermke D, Wohler N, Klemmer JH, Fourne M, Acar Y, Fahl S. Committed to trust: A qualitative study on security & trust in open source software projects. Proc – IEEE Symp Secur Priv. 2022;2022:1880–1896. https://doi.org/10.1109/SP46214.2022.9833686
11. Khalil C, Khalil S. Exploring knowledge management in agile software development organizations. Int Entrep Manag J. 2020;16(2):555–569. https://doi.org/10.1007/s11365-019-00582-9
12. Shastri Y, Hoda R, Amor R. The role of the project manager in agile software development projects. J Syst Softw. 2021;173:110871. https://doi.org/10.1016/j.jss.2020.110871
13. Azad N, Hyrynsalmi S. DevOps critical success factors – A systematic literature review. Inf Softw Technol. 2023;157:107150. https://doi.org/10.1016/j.infsof.2023.107150
14. Dei DGJ, Van der Walt TB. Knowledge management practices in universities: The role of communities of practice. Soc Sci Humanit Open. 2020;2(1):100025. https://doi.org/10.1016/j.ssaho.2020.100025
15. Politze M, Christoph U, Decker B, et al. Supporting Software Development Processes for Academia with GitLab. Eur J High Educ. 2023;95:229–218.
16. Baham C, Hirschheim R. Issues, challenges, and a proposed theoretical core of agile software development research. Inf Syst J. 2022;32(1):103–129. https://doi.org/10.1111/isj.12336
17. Prenner N, Unger-Windeler C, Schneider K. Goals and challenges in hybrid software development approaches. J Softw Evol Process 2021;33(11):1–25. https://doi.org/10.1002/smr.2382
18. Mazorodze AH, Buckley S. Knowledge management in knowledge-intensive organisations: Understanding its benefits, processes, infrastructure and barriers. SA J Inf Manag. 2019;21(1):1–6. https://doi.org/10.4102/sajim.v21i1.990
19. Kitchenham BA. Systematic review in software engineering: Where we are and where we should be going. EAST'12: Proceedings of the 2nd International Workshop on Evidential Assessment of Software Technologies; 2012 Sep 22; Lund, Sweden. p. 1–2.
20. Valente A, Holanda M, Mariano AM, Furuta R, Da Silva D. Analysis of academic databases for literature review in the computer science education field. IEEE Frontiers in Education Conference (FIE); 2022 Oct 8–11; Uppsala, Sweden. p. 1–7.
21. Ahmed YA, Ahmad MN, Ahmad N, Zakaria NH. Social media for knowledge-sharing: A systematic literature review. Telemat Inform. 2019;37:72–112. https://doi.org/10.1016/j.tele.2018.01.015
22. Jadhav D, Kundale J, Bhagwat S, Joshi J. A Systematic review of the tools and techniques in distributed Agile Software Development. Agile Software Development: Trends, Challenges and Applications. Agil Softw Dev Trends Challenges. 2023;161–186.
23. Luo MM, Chea S. Wiki use for knowledge integration and learning: A three tier conceptualization. Comput Educ. 2020;154:103920. https://doi.org/10.1016/j.compedu.2020.103920
24. Rasheed A, Zafar B, Shehryar T, et al. Requirement engineering challenges in Agile Software Development. Math Probl Eng. 2021;2021:1–18. https://doi.org/10.1155/2021/6696695
25. Kinsman T, Wessel M, Gerosa MA, Treude C. How do software developers use github actions to automate their workflows? Proc – 2021 IEEE/ACM 18th Int Conf Min Softw Repos MSR. 2021;2021:420–431. https://doi.org/10.1109/MSR52588.2021.00054
26. Dakhel AM, Majdinasab V, Nikanjam A, Khomh F, Desmarais MC, Jiang ZMJ. Github copilot ai pair programmer: Asset or liability? J Syst Softw. 2023;203:111734. https://doi.org/10.1016/j.jss.2023.111734
27. Kaur R, Chahal KK, Saini M. Understanding community participation and engagement in open-source software Projects: A systematic mapping study. J King Saud Univ Inf Sci. 2022;34(7):607–625. https://doi.org/10.1016/j.jksuci.2020.10.020
28. Zhang M, Liu H, Chen C, Liu Y, Bai S. Consistent or not? An investigation of using Pull Request Template in GitHub. Inf Softw Technol. 2022;144:106797. https://doi.org/10.1016/j.infsof.2021.106797
29. Nisar TM, Prabhakar G, Strakova L. Social media information benefits, knowledge management and smart organizations. J Bus Res. 2019;94:264–272. https://doi.org/10.1016/j.jbusres.2018.05.005
30. Lwakatare LE, Kilamo T, Karvonen T, et al. DevOps in practice: A multiple case study of five companies. Inf Softw Technol. 2019;114:217–230. https://doi.org/10.1016/j.infsof.2019.06.010
31. Pargaonkar S. A comprehensive research analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model advantages, disadvantages, and application suitability in software quality engineering. Int J Sci Res Publ. 2023;13(8):120–124. https://doi.org/10.29322/IJSRP.13.08.2023.p14015
32. Akhtar A, Bakhtawar B, Akhtar S. Extreme programming Vs scrum: A comparison of Agile Models. Int J Technol Innov Manag. 2022;2(2):80–96. https://doi.org/10.54489/ijtim.v2i2.77

33. Kellogg LH, Hwang LJ, Gassmoller R, Bangerth W, Heister T. The role of scientific communities in creating reusable software: Lessons from geophysics. Comput Sci Eng. 2019;21(2):25–35. https://doi.org/10.1109/MCSE.2018.2883326
34. Khan HU, Ali F, Nazir S. Systematic analysis of software development in cloud computing perceptions. J Softw Evol Process. 2024;36(2):1–26. https://doi.org/10.1002/smr.2485
35. Thota MK, Shajin FH, Rajesh P. Survey on software defect prediction techniques. Int J Appl Sci Eng. 2020;17(4):331–344.
36. Tam C, Da Costa Moura EJ, Oliveira T, Varajao J. The factors influencing the success of on-going agile software development projects. Int J Proj Manag. 2020;38(3):165–176. https://doi.org/10.1016/j.ijproman.2020.02.001

Appendix 1 starts on the next page →

Appendix 1

TABLE 1–A1: Publications included in the systematic literature review, sorted by author.

Article number	Author(s) and year	Title
1	Ahmed et al. 2019	Social media for knowledge-sharing: A systematic literature review. *Telematics and informatics*, *37*, pp. 72–112.
2	Akhtar et al. 2022	Extreme programming vs scrum: A comparison of agile models. *International Journal of Technology, Innovation and Management (IJTIM)*, *2*(2), pp. 80–96.
3	Alsanad et al. 2019	A domain ontology for software requirements change management in global software development environment. *IEEE Access*, *7*, pp. 49352–49361.
4	Al-Saqqa et al. 2020	Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies*, *14*(11).
5	Azad and Hyrynsalmi, 2023	DevOps critical success factors – A systematic literature review. *Information and Software Technology*, *157*, p. 107150.
6	Baham and Hirschheim, 2022	Issues, challenges, and a proposed theoretical core of agile software development research. *Information Systems Journal*, *32*(1), pp. 103–129.
7	Bicchi, 2022	Communities of practice and what they can do for International Relations. *Review of International Studies*, *48*(1), pp. 24–43.
8	Castaneda and Cuellar, 2020	Knowledge sharing and innovation: A systematic review. *Knowledge and Process Management*, *27*(3), pp. 159–173.
9	Castellani et al. 2021	Tacit knowledge sharing in knowledge-intensive firms: the perceptions of team members and team leaders. *Review of managerial science*, *15*(1), pp. 125–155.
10	Chen and Li, 2024	AI-Enabled Cloud Platforms: Revolutionizing Software Development. *Asian American Research Letters Journal*, *1*(1).
11	Choi et al. 2020	Communities of practice and knowledge management systems: effects on knowledge management activities and innovation performance. *Knowledge Management Research & Practice*, *18*(1), pp. 53–68.
12	Dakhel et al. 2023	GitHub copilot ai pair programmer: Asset or liability?. *Journal of Systems and Software*, *203*, p. 111734.
13	Dei and van der Walt, 2020	Knowledge management practices in universities: The role of communities of practice. *Social sciences & humanities open*, *2*(1), p. 100025.
14	Jadhav et al. 2023	A Systematic Review of the Tools and Techniques in Distributed Agile Software Development. *Agile Software Development: Trends, Challenges and Applications*, pp. 161–186.
15	Kaur et al. 2022	Understanding community participation and engagement in open-source software projects: A systematic mapping study. *Journal of Jing Saud University – Computer and Information Sciences*, *34*(7), pp. 4607–4625.
16	Kellogg et al. 2019	The Role of Scientific Communities in Creating Reusable Software: Lessons from Geophysics', *Computing in Science & Engineering*, 21(2), pp. 25–35.
17	Khalil and Khalil, 2020	Exploring knowledge management in agile software development organisations. *International Entrepreneurship and Management Journal*, *16*(2), pp. 555–569.
18	Khan et al. 2024	Systematic analysis of software development in cloud computing perceptions. *Journal of Software: Evolution and Process*, *36*(2), p. e2485.
19	Kinsman et al. 2021	How do software developers use GitHub actions to automate their workflows?. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)* (pp. 420–431). IEEE.
20	Luo and Chea, 2020	Wiki use for knowledge integration and learning: A three tier conceptualization. *Computers & Education*, *154*, p. 103920.
21	Lwakatare et al. 2019	DevOps in practice: A multiple case study of five companies. *Information and Software Technology*, *114*, pp. 217–230.
22	Mazorodze and Buckley, 2019	Knowledge Management in knowledge-intensive organisations: Understanding its benefits, processes, infrastructure and barriers. *South African Journal of Information Management (SAJIM),* 21(1): 1–6.
23	Nisar et al. 2019	Social media information benefits, knowledge management and smart organisations. *Journal of Business Research*, *94*, pp. 264–272.
24	Pargaonkar, 2023.	A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering. *International Journal of Scientific and Research Publications (IJSRP)*, *13*(08).
25	Politze et al. 2023	Supporting Software Development Processes for Academia with GitLab. *Proceedings of European University*, *95*, pp. 229–238.
26	Prenner et al. 2021	Goals and challenges in hybrid software development approaches. *Journal of Software: Evolution and Process*, *33*(11), p. e2382.
27	Rasheed et al. 2021	Requirement engineering challenges in agile software development. *Mathematical Problems in Engineering*, *2021*, pp. 1–18.
28	Shastri et al. 2021	The role of the project manager in agile software development projects. *Journal of Systems and Software*, *173*, p. 110871.
29	Tabatabaee and Sharma, 2021	Social Networks and Knowledge Sharing in Communities of Practice: A Developing Societies' Perspective. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(13), pp. 5626–5633.
30	Tam et al. 2020	The factors influencing the success of on-going agile software development projects. *International Journal of Project Management*, *38*(3), pp. 165–176.
31	Thota et al. 2020	Survey on software defect prediction techniques. *International Journal of Applied Science and Engineering*, *17*(4), pp. 331–344.
32	Valente et al. 2022	Analysis of academic databases for literature review in the computer science education field. In *2022 IEEE Frontiers in Education Conference (FIE)* (pp. 1–7). IEEE.
33	Weerasekara and Smedberg, 2022	The use of online communities of practice in the software industry: learnings from Facebook communities in Sri Lanka. *International Journal of Web Based Communities*, *18*(2), pp. 186–209.
34	Wermke et al. 2022	Committed to trust: A qualitative study on security & trust in open-source software projects. In *2022 IEEE Symposium on Security and Privacy (SP)* (pp. 1880–1896). IEEE.
35	Yao et al. 2020	Knowledge sharing and technological innovation capabilities of Chinese software SMEs. *Journal of Knowledge Management*, Vol. 24 No. 3, pp. 607–634.
36	Zhang et al. 2022	Consistent or not? An investigation of using Pull Request Template in GitHub. *Information and Software Technology*, *144*, p. 106797.