Vol. 8, No. 3, June 2026

Journal of Information Systems and Informatics

Published By Asosiasi Doktor Sistem Informasi Indonesia

AI-Assisted Development Tools and Team Dynamics in South African Software Engineering Teams

Bassesy Isong[1], Tshipuke Vhahangwele[2]

[1, 2]Computer Science Department, North-West University, Mafikeng, South Africa

Received:
October 11, 2025
Revised:
May 17, 2026
Accepted:
June 1, 2026
Published:
June 25, 2026

Corresponding Author:
Author Name*:
Bassey Isong
Email*:
bassey.isong@nwu.ac.za

DOI:
10.63158/journalisi.v8i3.1615

Abstract. AI-assisted development tools are widely adopted in software engineering (SE), yet their effects on team dynamics and software delivery outcomes remain poorly understood in sub-Saharan African settings. This paper investigates how AI tool integration influences team roles, collaboration, skill requirements, and software delivery outcomes among South African software development professionals. A mixed-methods design was used, combining a structured survey with thematic analysis of open-ended responses from 40 participants across developer, tester, DevOps, and team lead roles. Multiple linear regression and Spearman's rank correlation were applied to quantitative data; thematic analysis followed the six-phase approach of Braun and Clarke. Findings show that GitHub Copilot was used by 75% of respondents. Interpersonal trust was the strongest predictor of development speed ($\beta = 0.485$, $p = 0.017$), exceeding all AI-specific variables in the model. AI use at the adoption onset reduced development speed; frequency of use increased it. Role transformation was reported by 95% of respondents and predicted team productivity. However, causal inference is not warranted given the cross-sectional design and reliance on self-reported measures. The findings are further constrained by a purposive sample of 40 drawn from networked professional communities, which limits statistical power and generalisability. To the authors' knowledge, no prior published study has examined AI adoption and team dynamics within a South African SE population, using this combination of methods, though a systematic literature review of that literature was beyond the scope of this study.

Keywords: AI-assisted development tools, software team dynamics, software delivery outcomes, South African software development.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

1. INTRODUCTION

The adoption of AI-assisted development tools has changed how software teams build and deliver systems. Currently, tools like GitHub Copilot, ChatGPT, Claude, etc, have moved beyond experimental use into routine engineering workflows, with industry projections placing adoption rates in software development at 75% by 2026 [1], [2]. Some of the reported productivity gains range from 20% to 55% for routine tasks, and some case studies document more dramatic improvements, a 600% reduction in lead time in one documented instance [3]. These projections bring about increased organisational interest in AI integration across the Software Development Life Cycle (SDLC). However, the empirical picture is less consistent than adoption rates suggest, as measured performance gains diverge from self-reported ones. For instance, developers estimate productivity improvements of approximately 31%, while objective completion-time data shows a 14.2% decrease [4], [5]. Likewise, security analyses of AI-generated code report vulnerability rates between 40% and 50% against MITRE's Top 25 Common Weakness Enumeration [4], [6]. Longitudinal repository data also shows a 4x increase in code cloning and a decline in refactoring activity from 25% to under 10% of commits following AI adoption [3]. Thus, tool availability does not translate directly into quality outcomes.

The organisational dimensions of this transition are less studied than the technical ones. Successful AI integration depends on team culture, leadership approach, and knowledge-sharing norms rather than on technical capability alone [7], [8], [9]. Today, the developer's role is shifting from code production toward validation and oversight. This is what researchers described as an agent architect model, where engineers direct and repair machine-generated output rather than write from scratch [10], [11], [12]. How this shift affects skill development, team collaboration, and workforce capability over time remains unresolved in the literature. Research on these organisational factors is limited even in well-resourced settings. Particularly, in South Africa, the gap is more pronounced. This is because senior developer emigration has reduced the experienced personnel available to facilitate team learning. On the other hand, domestic digital transformation initiatives are accelerating AI tool adoption before its effects are understood. Whether findings from North American or European perspectives transfer to this setting has not been empirically examined.

This study addresses that gap by applying a mixed-methods design, combining a structured survey with thematic analysis of open-ended responses, to data from 40 South African software development professionals. The study was conducted across developers, testers, DevOps, and team lead roles. The objective was to examine how AI-assisted tools influence roles, collaboration patterns, and software delivery outcomes in this setting [13]. To this end, three findings were observed from this study. First, interpersonal trust predicted development speed more strongly than any AI-specific variable measured. Second, AI adoption reduced development speed at the onset, while frequency of use increased it, a deferred pattern with direct implications for how organizations measured return on investment (ROI). Third, role change due to AI was positively associated with team productivity, while teams that did not reorganize responsibilities showed weaker delivery outcomes in this sample. These findings constitute context-specific evidence on AI integration from a South African SE population. This is a population that has not been previously investigated in this combination of methods or at this level of organizational detail.

This paper is structured as follows: Section 2 presents the related works, Section 3 presents the methodology used, and Section 4 presents the results. Section 5 discusses the findings, strengths, and weaknesses, while Section 6 concludes the paper.

2. RELATED WORK

2.1. Shifting role of developers in AI-assisted environments

In recent times, AI tools have moved developers' work away from code creation toward validation and oversight of machine-generated output [10], [12]. Now, developers increasingly operate as agent architects, directing, repairing, and evaluating AI suggestions rather than writing logic from scratch [3], [10]. This transition has been reported across multiple settings. For instance, Product Owners take on a requirement strategy focused on high-quality prompts, while developers move toward architectural evaluation [3], [14]. A recurring strategy in practice is “Accept, but Modify,” where developers retain responsibility by manually refining AI-generated logic before committing it [15], [16]. Mendes et al. [16] described this using the metaphor of a bicycle with a small motor: helpful for velocity but requiring constant steering. The tool accelerates routine work but does not remove the need for judgment.

Effective adoption depends on what researchers term context engineering, providing the AI with file trees, database schemas, and CI/CD logs rather than relying on prompt phrasing alone [5], [10]. However, two structural barriers limit this approach. Spatial context blindness refers to the model's inability to grasp codebase interdependencies [10]. Temporal context decay refers to pre-training cutoffs that cause the model to suggest deprecated implementations [10]. Both barriers are pronounced in environments where codebases are less standardised and documentation is sparse.

2.2. Team dynamics, collaboration, and mentorship

SE depends on team interaction, and AI adoption disrupts established collaboration patterns. Empirical data show that 45% of developers report relying less on colleagues after adopting GenAI tools [17]. AI provides a low-judgment environment where junior developers can ask questions freely, what Ferino et al. [18] call Large Language Model (LLM) entrepreneurship. This benefit carries a cost. That is, the same tools reduce spontaneous human-to-human explanation episodes, weakening the mentorship channel that transfers tacit knowledge within teams [18], [19]. On the same note, Welter et al. [19] identify a specific risk in pair programming: the trust finish type, where a developer accepts AI-generated logic without understanding it. This pattern is efficient in the short term but undermines skill formation over time. Similarly, Chen [20] reported AI reduces productivity inequality within teams by approximately 40%, levelling output between high- and low-skilled developers. Whether this equalisation reflects genuine skill development or tool dependency remains unknown.

In Agile settings, team cohesion and knowledge-sharing culture are stronger predictors of successful AI integration than technical capability [7], [8]. The adoption is often driven by social influence and peer pressure rather than objective assessments of tool performance [21]. Stray et al. [17] identified what they referred to as “Fence-Sitters”, experienced developers who wait for peers to validate a tool before adopting it, as a distinct pattern in public sector organisations. Senior developer emigration in South Africa creates knowledge gaps where informal peer networks may carry greater influence than formal training, making this dynamic relevant to the present study.

2.3. Measured gains and the perception gap in productivity

Productivity research reveals a gap between what developers believe AI tools provide and what controlled studies measure. Developers self-report improvements of approximately 31.4%, while meta-analytic data from 47 studies covering 77,000 organisations show a measured task-completion improvement of 14.2% [6]. Johnson et al. [6] described this 45.6%-point divergence as a Productivity Paradox, and attribute it partly to the Novelty and Hawthorne effects in self-reported data. At the task level, AI performs well on low-complexity work. Mo et al. [5] report GitHub Copilot correctness rates of 89.3% for easy problems, declining to 43.4% for hard ones across 8,132 annotated suggestions. In complex scenarios, experienced developers took 19% longer to complete tasks with AI assistance than without it, due to the cognitive cost of validating incorrect suggestions [4], [10], [22]. Tomaz et al. [11] reported a P-A-E Divergence, where teams show a 59.1% increase in story-point delivery while committed code volume remains flat. This finding makes lines of code an unreliable productivity indicator in AI-assisted environments. Kumar et al. [22] also reported a 31.8% reduction in pull request review cycle time and a 61% increase in code volume pushed to production across a 300-engineer deployment. These gains focused on specific workflow stages and were not distributed evenly. Paradis et al. [23] also found that speed gains in a randomised controlled trial varied by developer seniority and did not reach statistical significance across all levels.

2.4. Software quality, security, and technical debt

Security is the most consistently documented concern in the AI code generation literature. Multiple large-scale analyses report that 40-50% of AI-generated code contains exploitable vulnerabilities measured against MITRE's Top 25 Common Weakness Enumeration categories [4], [6]. Sarkar [4] found that these rates keep on in production frontend deployments. Muñoz et al. [24] compared eleven AI engines on identical CRUD tasks using SonarQube static analysis and found that output quality varies across engines and programming languages, with Bing and Gemini showing lower defect counts than Copilot on the same tasks.

Johnson et al. [6] used a credit card metaphor for the long-term cost of AI adoption. They reported that immediate velocity gains accumulate maintenance obligations that reduce future delivery speed by 12-18%. Accordingly, longitudinal repository analysis confirms

this, noting that AI adoption correlates with a 4x increase in code cloning and a reduction in refactoring activity from 25% to under 10% of commits [6]. Souza [25] equally reported an 85% reduction in bug density in a Brazilian public sector study, but this conflicts with repository-level evidence and is attributed to self-report bias and short measurement windows. They proposed alignment with quality standards such as ISO 25000 as a governance mechanism for managing the trade-off between delivery speed and long-term code reliability [25]. Mandatory security scanning of AI-generated output is the mitigation most consistently supported by the evidence [4], [25].

Table 1. Summary of Related Works

Study	Focus	Methodology	Sample	Finding	Limitation
[5]	Copilot correctness	Empirical benchmark analysis	8,132 code suggestions	Correctness drops from 89% (easy) to 43% (hard tasks)	Non-determinism of Copilot outputs
[6]	Productivity paradox	Systematic review (47 studies)	77,000 organisations	45.6-point gap between perceived and measured productivity	Variability in how "success" is defined across studies
[10]	Industry adoption patterns	Mixed-methods survey & interviews	109 submissions	"Context Wall" is the primary structural barrier to adoption	Cross-sectional; no longitudinal tracking
[11]	Agile team productivity	Longitudinal multi-case study	3 agile teams (13 months)	59.1% value increase with Flat code volume (P-A-E Divergence)	Skill atrophy flagged as a long-term concern
[17]	Public sector collaboration	Mixed methods interviews & survey	13 interviews / 114 surveys	45% of users rely less on colleagues after AI adoption	Findings may not generalise beyond 2025
[18]	LLM adoption experiences	Socio-technical grounded theory	22 semi-structured interviews	AI reduces senior-junior mentorship while creating a low-judgment learning space.	Demographic discrepancies in recruitment
[19]	Pair programming dynamics	Qualitative voice/screen recording	19 participants	AI reduces human explanation episodes; introduces "Trust" finish type	Small sample limits generalisability
[21]	Adoption drivers	UTAUT survey & regression	305 participants	Social influence outweighs technical merit in adoption decisions	Instruments not validated across professional vs. student populations
[22]	Enterprise AI deployment	Quasi-experimental longitudinal	300 software engineers	31.8% reduction in PR review time; 61% code volume increase	Single-organisation study

[23]	Development speed	Randomised controlled trial	96 participants	Speed gains vary by seniority; not significant across all levels	Single-time-point laboratory setting
[25]	Quality and team dynamics	Mixed-methods (K-Means/Gioia)	101 practitioners	85% bug reduction reported: 65% faster delivery	Predominantly Brazilian sample; no longitudinal data

Table 1 summarizes some of the existing empirical studies. Most samples in the reviewed studies are drawn from North American, European, or Brazilian organisations [10], [17], [24]. No study in this review investigates AI tool adoption within South African SE teams. This is because the South African setting differs from the studied environments in infrastructure reliability, senior developer availability, and the pace of organisational AI readiness, factors that affect both adoption patterns and team dynamics in ways the existing literature does not address. Therefore, longitudinal evidence beyond twelve months is rare. Skill wasting among junior developers who rely heavily on AI remains unmeasured, with researchers calling for studies of 24 months or longer to capture the effect [6], [10], [11]. The organisational dimensions of adoption: team cohesion, leadership strategy, and knowledge-sharing culture, are identified as determinants of integration success [7], [8], [9], yet most studies treat these as background variables rather than primary research objects. This study addresses both the geographic and thematic gaps by examining AI adoption through the lens of team dynamics in a South African industry setting.

3. METHODOLOGY

This study adopted a mixed-methods design [10], [17], [25], to investigate how AI-assisted development tools influence team dynamics, roles, and software delivery outcomes in South African software teams. The design followed a parallel convergent structure, shown in Fig. 1. We collected both quantitative and qualitative data from the same participants simultaneously, analysed independently, and integrated at the preparation stage. The quantitative strand used multiple linear regression (MLR) and Spearman's rank correlation to examine relationships between AI usage, team dynamics, and software outcomes. The qualitative strand used thematic analysis to provide contextual insight into how AI reshapes team interaction. We combined them after both strands were analysed. In this case, quantitative findings were compared against qualitative themes to

identify convergence, divergence, and complementary patterns. Where the two strands agreed, findings were treated as corroborated. Where they diverged, the discrepancy was noted and discussed as a limitation of the instrument or sample. The target population consisted of software development professionals in South Africa actively involved in development work. This included developers, testers, DevOps engineers, and team leaders working across different organisational and project settings. Purposive sampling was used to ensure all respondents had direct experience with AI tools in software development. We did not achieve even role distribution as developers constituted 75% of the final sample, which reflects the composition of the professional networks through which recruitment was conducted. This imbalance limits the extent to which findings can be interpreted as representative of non-developer roles, and results should be read with this in mind. Recruitment was conducted through LinkedIn, developer community forums, and direct outreach to industry professionals. A total of 40 valid responses were received.

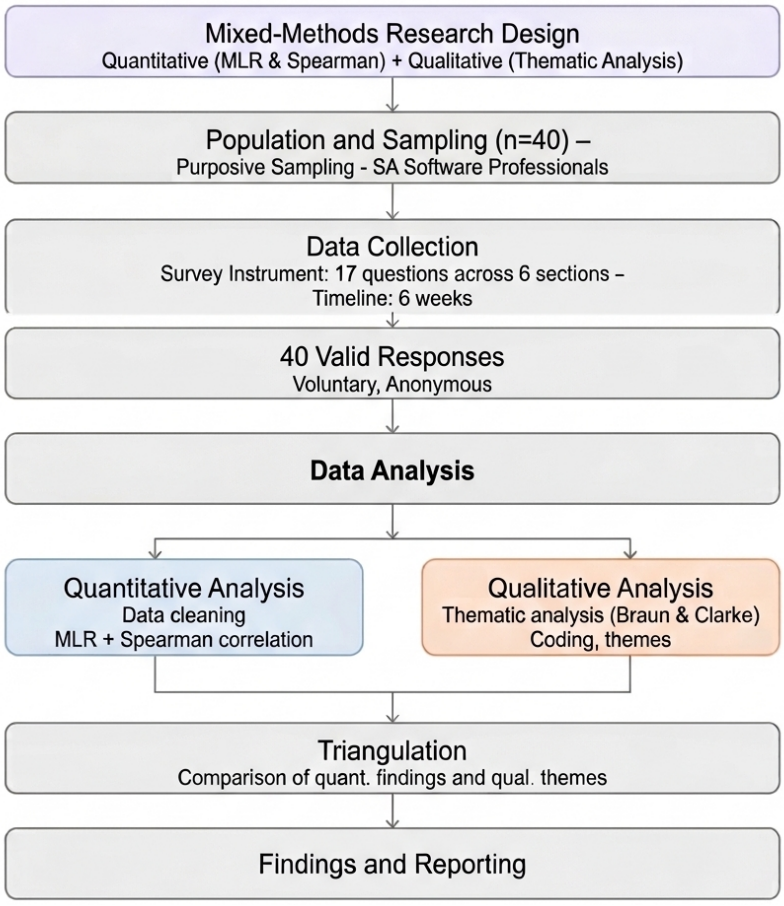


Figure 1. Workflow of the mixed-methods research design, showing how quantitative and qualitative analysis proceed in parallel before converging at triangulation and reporting.

Data were collected using a structured online survey developed from prior literature on team dynamics, AI-assisted development, and software productivity [10], [17], [21]. The questionnaire comprised 17 main questions organised into six sections. The first section captured demographic information: participant roles, team sizes, project focus, and development methodologies. The second section examined AI tool adoption and frequency of use. The third section addressed team dynamics, covering communication, trust, collaboration, and psychological safety (PS). The fourth section recorded skills and role changes resulting from AI integration. The fifth section measured software outcomes across development speed, code quality, and team productivity. The sixth section contained open-ended questions on participant experiences and perceptions of AI in development practice. Items combined Likert-scale, multiple-choice, and open-ended formats.

Before deployment, the instruments were pilot tested to assess clarity, relevance, and internal consistency. In this case, questions were reviewed and refined based on pilot feedback, with adjustments made to item wording and response scales before full distribution. Internal consistency was not formally assessed using Cronbach's alpha. This is because the survey combined single-item measures and multi-format response scales that do not meet the assumptions required for alpha computation. Construct validity was supported by grounding all items in established constructs from the AI adoption and team dynamics literature [10], [17], [21]. Also, it is by pilot testing for item clarity before full distribution, while confirmatory factor analysis was beyond the scope of this study. Construct validity claims are, therefore, limited to content and face validity.

Following ethical approval from the North-West University (NWU) Research Ethics Committee, the survey was administered online via Google Forms and distributed to participants across different roles, experience levels, and organisational contexts. Participation was voluntary. An introductory statement described the study's purpose and assured participants of confidentiality and anonymity. No personally identifiable information was collected. The survey remained open for approximately six weeks. Most participants completed it within 15 minutes. Collected data were cleaned and coded before analysis. Categorical variables were converted into dummy variables to facilitate regression analysis. Ordinal variables were scaled to preserve their ranked nature. Missing

data were handled using listwise deletion. Only complete cases were included in the analysis.

Two quantitative techniques were applied. MLR was used to examine the influence of independent variables on software outcomes. Although several outcome measures were captured on Likert-type scales, MLR was retained rather than ordinal regression for two reasons. First, prior SE studies applying similar constructs have treated Likert responses as interval-approximate under comparable conditions [29]. Second, ordinal logistic regression would have reduced the interpretability of the beta coefficient, which was central to the comparative analysis of predictor strength. This decision is acknowledged as a limitation. Results should be interpreted with caution, and replication with ordinal-appropriate models is recommended. The MLR model took the form:

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon \quad (1)$$

Where Y is the dependent variable, X_n are the predictors, β_n are the regression coefficients, and ε is the error term.

Before MLR analysis, four assumptions were assessed. Multicollinearity was evaluated using tolerance and Variance Inflation Factor (VIF) values. All VIF values ranged between 1.066 and 1.843, we below the recommended threshold of 5, indicating no multicollinearity concerns. Residual was examined using histograms and Normal P-P plots of standardized residual against predicted values. No major violations were detected. Independence of errors was confirmed using Durbin-Watson statistics ranging from 1.373 to 2.258, falling within acceptable bounds.

Spearman's rank correlation was used to assess relationships between ordinal variables, as it makes no assumptions of interval scaling or normality:

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} \quad (2)$$

Where d_i is the difference between paired ranks, and n is the number of observations. Furthermore, responses were analysed using thematic analysis following the six-phase approach of Braun and Clarke [26] and Ahmed et al. [27]. Responses were read repeatedly

to establish familiarity with the data, and important ideas and recurring patterns were identified and coded. Codes were grouped into broader themes, then reviewed and refined to ensure they were distinct and accurately represented the data. Each theme was defined and supported with participant quotes. Frequency counts identified the most common themes across the dataset. Qualitative findings were then compared against the quantitative results at the convergence stage described above.

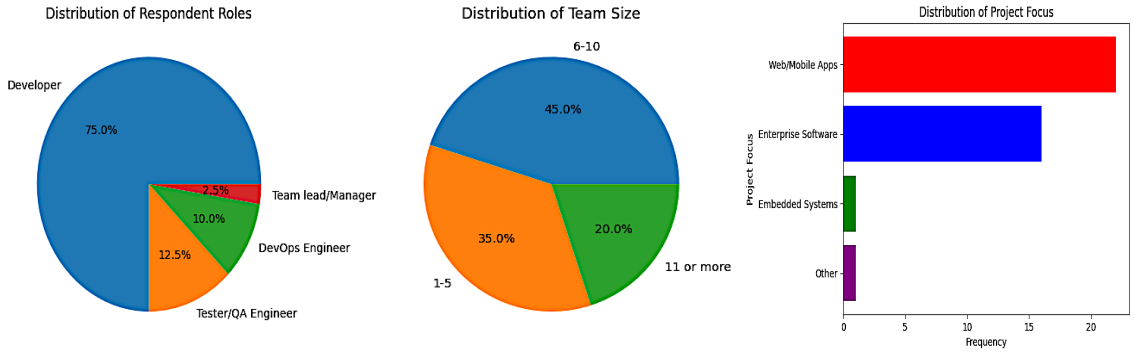
Furthermore, qualitative coding was conducted by the primary researcher. To assess consistency, a second independent reviewer coded a randomly selected subset of 10 responses, 25% of the qualitative dataset, using the same initial codebook. Inter-rater agreement was evaluated by comparing code assignments across the subset. Where disagreements arose, the two reviewers discussed each case and reached consensus through re-examination of the original response in context. Ambiguous responses that could not be resolved through discussion were excluded from theme frequency counts and noted as a limitation of the open-ended survey format.

The sample of 40 warrants explicit discussion. With this sample size, MLR models are statistically underpowered by conventional benchmarks, and role representation was uneven across the four groups; developers constituted 75% of respondents. These constraints mean the quantitative findings cannot support population-level inference, and all results should be treated as exploratory. The findings are intended to identify patterns and establish a baseline for future work, not to provide definitive estimates of effect sizes or causal relationships. A larger, stratified follow-up study is needed to confirm or revise these patterns. Confidence intervals for regression coefficients were not computed in the original analysis because generating them post-hoc risked introducing inconsistencies with the reported output. Given the exploratory framing, standardised coefficients and standard errors are reported as sufficient for hypothesis-generating purposes. All data were handled in compliance with the Protection of Personal Information Act (POPIA).

4. RESULTS AND DISCUSSION

4.1. Sample Characteristics

The sample comprised 40 software development professionals. As shown in Figure 2, developers constituted the largest role group at 75%, followed by tester/QA engineers (12.5%), DevOps engineers (10%), and team leads or managers (2.5%). Fig. 2 also shows that most respondents worked in teams of 6-10 members (45%), with 35% in teams of 1-5 and 20% in teams of 11 or more. Web and mobile application development was the dominant project focus (55%), followed by enterprise software (40%), with embedded systems and other project types each accounting for 2.5%. Agile was the most common development methodology (67.5%), followed by DevOps (10%), Waterfall (10%), hybrid Agile/DevOps (7.5%), and other approaches (5%). Developers made up 75% of the sample. These findings, therefore, reflect developer experience disproportionately and should be read accordingly.



(a) (b) (c)

Fig. 2. Distribution: (a) Roles (b) Team size (c) Project focus

4.2. AI Tool Adoption and Usage Patterns

Fig. 3 shows that 97.5% of respondents used AI tools during development; only 2.5% reported no AI use. GitHub Copilot was the most widely used tool (75%), followed by AI testing tools (27.5%), other AI tools (22.5%), AI code review tools (15%), and Tabnine (5%). Usage frequency was high, as 45% reported using AI tools often, 27.5% always, 25%

sometimes, and 2.5% rarely. Adoption was concentrated in generation rather than testing or review tools, a pattern relevant to interpreting the outcome models in Section 4.4.

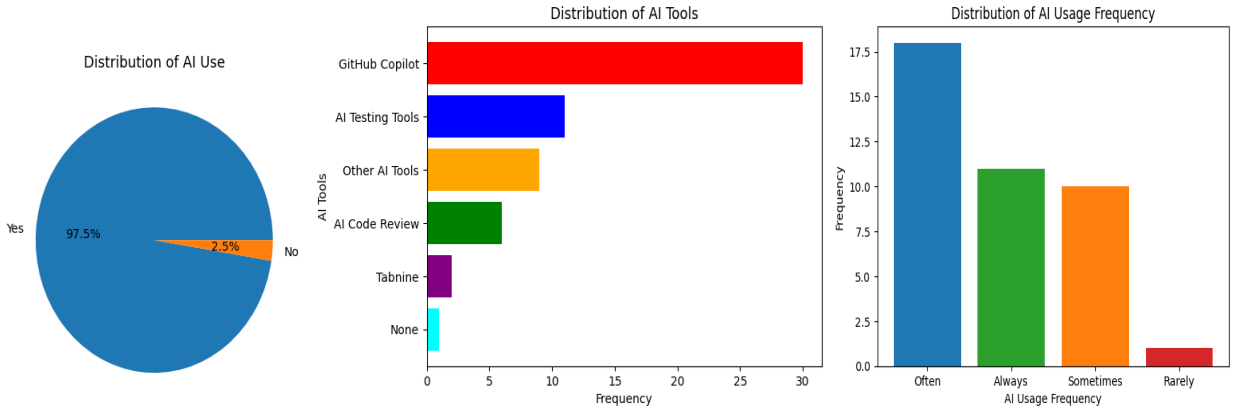


Figure 3. Distribution: (a) AI use (b) AI development tools (c) AI usage frequency

4.3. Team roles and responsibilities, impacts, and development practices distribution

As shown in Fig. 4, 77.5% of respondents reported that AI tools had slightly changed their team roles; 17.5% reported a significant role change, and 5% reported no change. Fig. 4 also presents the distribution of development practices. Accordingly, Agile was the most common development methodology with 67.5%, followed by DevOps and Waterfall at 10% each. Hybrid Agile/DevOps approach at 7.5%. Taken together, 95% of respondents reported at least some role change, a pattern examined further in the regression results as shown in Figure 4.

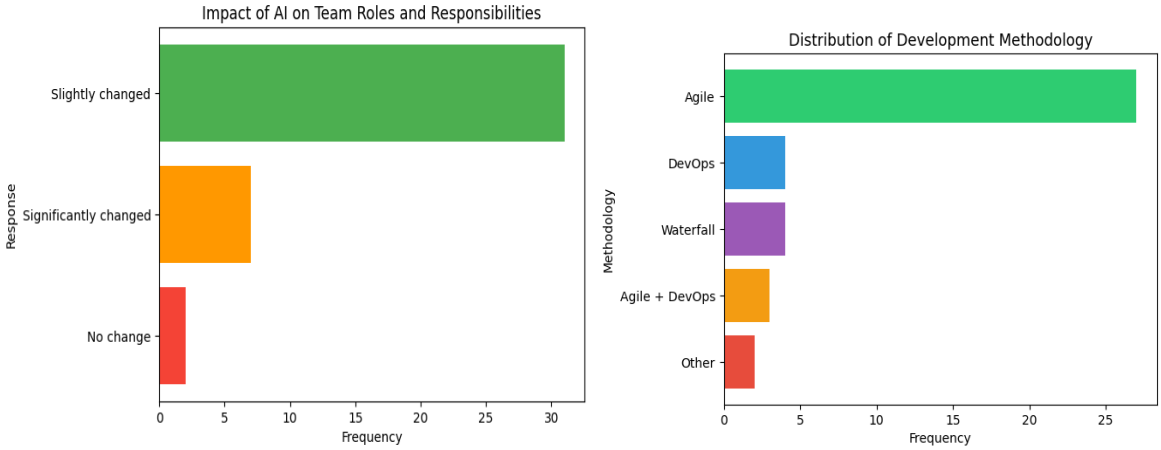


Figure 4. (a) AI impact on team roles (b) distribution of development methodologies

4.4. Multiple Linear Regression Results

With 40 observations and eight predictors in the development speed model, the observations-to-predictors ratio is lower than the conventional standard. All regression results are exploratory. Effect sizes and directions are reported to generate hypotheses, not to establish population-level estimates. Full regression output: unstandardized coefficients (B0, standard errors (SE), and standardized coefficients (β), t-statistics, and significant values (P), is presented in Tables 2a, 2b, and 2c.

4.4.1. Development Speed

Table 2a presents the MLR results for development speed (Model 10). The model accounts for 43.6% of variance (R=0.660, R^2=0.435, Adjusted R^2 = 0.291, SE=0.256, and p=0.031). Eight predictors were retained.

Table 2a. Development Speed (Model 10)

Model	Predictor	B	SE	β	t	p
10	(Constant)	1.783	0.456		3.912	<0.001
	Team size	0.180	0.072	0.435	2.483	0.019
	Development methodology	−0.117	0.056	−0.334	−2.085	0.045
	AI use during development	−0.723	0.308	−0.376	−2.348	0.025
	Frequency of AI tool usage	0.135	0.064	0.357	2.112	0.043
	AI improves team communication	−0.133	0.063	−0.309	−2.092	0.045
	AI causes conflicts/ misunderstandings	−0.296	0.111	−0.441	−2.671	0.012
	Openness about AI reliability	−0.156	0.071	−0.347	−2.189	0.036
	Interpersonal trust	0.218	0.086	0.485	2.523	0.017

Dependent variable: Development speed. All results are exploratory, given n = 40.

Team size was positively associated with development speed (β = 0.435, p = 0.019), and development methodology showed a negative association (β = -0.334, p = 0.045). AI use during development was negatively associated with development speed (β = -0.376, p = 0.025), while frequency of use showed a positive association (β = 0.357, p = 0.043). This divergence is further examined in the discussion section. AI improving team communication was negatively associated with development speed (β = -0.309, p = 0.045) while AI-related conflict showed the largest constraining association (β = -0.441, p = 0.012). On a similar note, openness about AI reliability was negatively associated with development speed (β = -0.347, p = 0.036). Interpersonal trust showed the largest positive association among all predictors (β = 0.485, p = 0.017). These are associations in a cross-sectional model; hence, no causal direction is implied.

4.4.2. Code Quality

Table 2b presents the MLR results for code quality (Model 16). The model accounts for 26% of variance (R=0.510, R^2=0.260, Adjusted R^2 = 0.220, SE=0.505, and p=0.085). The overall model did not reach conventional statistical significance (p=0.085). Hence, the two predictor associations below are reported for completeness only and should not be interpreted independently of replication in a larger sample.

Table 2b. Code Quality (Model 16)

Model	Predictor	B	SE	β	t	p
16	(Constant)	1.204	0.426		2.829	0.008
	Reduction in the need for technical skills	0.277	0.122	0.328	2.276	0.029
	AI improves team communication	0.374	0.117	0.462	3.200	0.003

Dependent variable: Code quality. The model did not reach significance at p<0.05. All results are exploratory, given n=40.

Reduction in the need for technical skills was positively associated with code quality (β = 0.328, p = 0.029). AI improving team communication showed a stronger positive association (β = 0.462, p = 0.003). The skill-reduction findings are discussed further in Section 5.

4.4.3. Team Productivity

Table 2c presents the MLR results for team productivity (Model 15). The model accounts for 24.0% of variance (R=0.490, R^2=0.240, Adjusted R^2 = 0.177, SE=0.398, and p=0.018). The three predictors were retained.

Table 2c. Team Productivity (Model 15)

Model	Predictor	B	SE	β	t	p
15	(Constant)	−1.288	0.613		−2.102	0.043
	Team size	0.194	0.094	0.326	2.054	0.047
	AI use during development	0.953	0.417	0.344	2.286	0.028
	Change in team roles due to AI	0.353	0.145	0.373	2.430	0.020

Dependent variable: Team Productivity. All results are exploratory, given n=40.

Team size (β = 0.326, p = 0.047), AI use during development (β = 0.344, p = 0.028), and role change due to AI (β = 0.373, p = 0.020) were positively associated with team productivity. The Adjusted R2 of 0.177 indicates that unmeasured factors account for most variance in this model.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

4.5. Spearman's Rank Correlation Results

Table 3 presents the Spearman's rank correlation results. Frequency of AI tool usage showed a strong positive correlation in the dataset, with role change ($\rho = 0.596$, $p < 0.001$). It was also positively associated with the need for new technical skills ($\rho = 0.326$, $p = 0.040$). Perceived skill reduction was negatively correlated with new skill requirements ($\rho = -0.373$, $p = 0.018$) and with change in review or pair programming frequency ($\rho = -0.461$, $p = 0.003$). Similarly, AI-enabled improvements were positively correlated with new technical skill requirements ($\rho = 0.416$, $p = 0.008$), shift to higher-level problem solving ($\rho = 0.439$, $p = 0.005$), and role change ($\rho = 0.340$, $p = 0.032$). AI reducing the need for discussions was negatively correlated with higher-level problem solving ($\rho = -0.422$, $p = 0.007$). AI-related conflict was negatively associated with higher-level problem solving ($\rho = -0.351$, $p = 0.026$). PS was negatively associated with development methodology ($\rho = -0.403$, $p = 0.010$). Openness about AI reliability correlated positively with team size ($\rho = 0.488$, $p = 0.001$) and a positive relationship with PS ($\rho = 0.393$, $p = 0.012$). Interpersonal trust was negatively associated with team size ($\rho = -0.320$, $p = 0.044$) and positively associated with AI-related conflict ($\rho = 0.403$, $p = 0.010$). Frequency of AI tool use showed a positive correlation with development speed ($\rho = 0.325$, $p = 0.041$).

Table 3. Spearman's rank correlation results

Relationship	ρ	p
AI use frequency ↔ Role change	0.596	<0.001
AI use frequency ↔ New technical skills	0.326	0.040
Skill reduction ↔ New technical skills	−0.373	0.018
Skill reduction ↔ Review/pair programming frequency	−0.461	0.003
Communication improvement ↔ New technical skills	0.416	0.008
Communication improvement ↔ Higher-level problem solving	0.439	0.005
Communication improvement ↔ Role change	0.340	0.032
AI reduces discussions ↔ Higher-level problem solving	−0.422	0.007
AI conflict ↔ Higher-level problem solving	−0.351	0.026
Psychological safety ↔ Development methodology	−0.403	0.010
Openness about AI reliability ↔ Team size	0.488	0.001
Openness about AI reliability ↔ Psychological safety	0.393	0.012
Interpersonal trust ↔ Team size	−0.320	0.044
Interpersonal trust ↔ AI conflict	0.403	0.010
AI use frequency ↔ Development speed	0.325	0.041

Note: All results are exploratory, given n=40. Correlations reflect association only.

4.6. Qualitative Findings

Open-ended responses were analysed using thematic analysis following [26],[27]. Five themes were identified. Table 4 presents theme frequencies.

Table 4. Qualitative theme frequencies

Theme	Frequency (n)
Productivity and Efficiency	18
Transformation of Developer Roles and Skills	10
Challenges in Trust, Accuracy, and Integration	15
Impact on Software Outcomes	9
Need for Improved AI Tool Capabilities	14

Theme 1: Productivity and Efficiency (n = 18) was the most frequent. Participants described AI tools as reducing repetitive coding tasks and accelerating development. This includes "Saves time on repetitive tasks" (R1), "Speeds up development significantly" (R3), and "Automates routine coding work" (R6). This aligns with the positive association between AI use frequency and speed of development discussed in Table 2a.

Theme 2: Transformation of Developer Roles and Skills (n = 10) captured reduced manual coding and a shift toward supervision and validation. This includes "Less manual coding is required" (R1), "Developers now supervise AI outputs" (R3), and "Shift from coding to validation" (R4). This corroborates the role-change association in Table 2c.

Theme 3: Challenges in Trust, Accuracy, and Integration (n = 15) was the second most frequent. Accuracy concerns, distrust of AI outputs, and integration difficulties were reported across roles. This includes "Accuracy is still a concern" (R1), "Outputs are not always reliable" (R2), and "Needs validation before use" (R6). This is consistent with the negative association between AI adoption onset and development speed in Table 2a.

Theme 4: Impact on Software Outcomes (n = 9) indicated that AI accelerates delivery, but with variable quality effects. This includes "Improves speed of delivery" (R1), "Quality can improve but depends on validation" (R3), and "Needs human oversight to ensure quality" (R6). Quality improvement conditional on validation is consistent with the code quality model in Table 2b.

Theme 5: Need for Improved AI Tool Capabilities (n = 14) showed that accuracy, contextual understanding, and reliability remain unmet across the sample. This includes "Needs better accuracy" (R1), "Should improve contextual understanding" (R2), and "Better

handling of complex tasks is required" (R4). This exists despite near-universal adoption, indicating a gap between expected and actual output quality that current tools have closed.

4.7. Discussion

This section examines the findings against prior literature. Association language is used throughout, and the cross-sectional design does not permit causal inference. All quantitative results are exploratory. The sample was developer-heavy: 75% developers, 2.5% team leads, meaning team-level interpretations rest primarily on developer perceptions and should not be read as team-level measurements.

Our findings reveal that AI tool adoption was near-universal in this sample, with 97.5% of respondents using AI during development, and GitHub Copilot accounting for 75% of tool usage. Adoption was concentrated in code-generation rather than testing, review, or governance tools. Felder et al. [10] reported the dominant mode of AI integration in SE. The governance gap identified by Johnson et al. [6], organisations prioritising velocity over quality oversight, accumulate technical debt at rates that reduce future delivery speed by 12-18%, is relevant here. This is because the tools adopted in this sample are not those that address output quality directly. For instance, Theme 5 (n = 14, Table 4) shows that accuracy and reliability remain unmet expectations even at near-universal adoption, suggesting that current tools do not meet production-level requirements across the full sample.

Table 2a shows a negative association between AI use during development and development speed (β = -0.376, p = 0.025) alongside a positive association between frequency of use and development speed (β = 0.357, p = 0.043). These associations point in opposite directions and require careful interpretation. This is consistent with the Productivity Paradox findings by Johnson et al. [6], who reported a 45.6%-point gap between perceived and measured productivity gains. The negative association at adoption onset is consistent with Mo et al. [5], who found that AI correctness drops to 43.4% for harder tasks, placing a verification burden on developers that slows initial delivery. Speed associations improve as teams accumulate familiarity with validation workflows, which is also consistent with Tomaz et al. [11] observations across 13 months of longitudinal data. Theme 3 (n = 15) confirms this. That is, accuracy concerns and

integration difficulty were the second most frequent barriers reported, regardless of overall positive assessments of productivity. Organizations that evaluate AI return on investment over short windows may record speed reductions that disappear with sustained use. The data here cannot establish how long the adjustment period lasts, which requires a longitudinal design.

Among all predictors in Table 2a, interpersonal trust produced the largest standardized coefficient ($\beta = 0.485$, $p = 0.017$), with every AI-specific variable in the model returning a smaller effect size. This association is in line with the team cohesion findings of Stray et al. [17] and the Agile literature [7], [8], which position relational factors as stronger correlates of team output than tool capability. AI-related conflict showed the largest negative association with development speed ($\beta = -0.441$, $p = 0.012$), larger than any technical variable in the model. Openness about AI reliability was negatively associated with development speed ($\beta = -0.347$, $p = 0.036$), meaning teams that actively interrogate AI output introduce additional verification cycles that may reduce short-term speed. The Spearman correlation between openness about AI reliability and PS ($\rho = 0.393$, $p = 0.012$, Table 3) shows that this critical stance is more prevalent where PS is present. Whether trust correlates with speed because trusted teams coordinate more efficiently, or whether faster teams develop stronger trust over time, is not determinable from this design.

Table 2c shows that role change due to AI was positively associated with team productivity ($\beta = 0.373$, $p = 0.020$). AI use during development was also positively associated with productivity ($\beta = 0.344$, $p = 0.028$) in this model. This is opposite in direction to its association with development speed in Table 2a. This divergence suggests that AI adoption may relate differently to speed and productivity as constructs, and these outcomes should not be treated as interchangeable in future work. The role-change association in this study is consistent with Felder et al. [1] and Tomaz et al. [11] findings. Both studies documented developer transitions toward validation and oversight. Theme 2 ($n = 10$) records the same shift in this sample. The strongest correlation in Table 3, between frequency of AI use and role change ($\rho = 0.596$, $p < 0.001$), shows that more intensive use is associated with greater structural team adaptation. This is consistent with Ferino et al. [18].

In the same vein, frequency of AI use was positively associated with the need for new technical skills (ρ = 0.326, p = 0.040), while perceived skill reduction was negatively correlated with new skill requirements (ρ = -0.373, p = 0.018, Table 3). These associations together suggest redistribution. That is, as routine coding tasks become less central, higher-order competencies in supervision, validation, and output management become more necessary. The negative correlation between skill reduction and review or pair programming frequency (ρ = -0.461, p = 0.003) indicates that teams that perceive skill erosion compensate through increased verification activity. This is consistent with the finding of Welter et al. [19]. However, the positive association between skill reduction and code quality (β = 0.328, p = 0.029, Table 2b) does not indicate that less skill produces better code. In this case, a more consistent interpretation is that when developers are less occupied with routine syntax tasks, attention may shift to higher-order quality concerns, consistent with Chen's [20] reports. This interpretation is speculative given the cross-sectional design.

AI improving team communication was positively associated with code quality in Table 2b (β = 0.462, p = 0.003). The overall code quality model did not reach conventional significance (p = 0.085), so these predictor associations should be treated with caution and not interpreted without replication in a larger sample. Table 3 shows that communication improvement co-occurs with shifts to higher-level problem solving (ρ = 0.439, p = 0.005) and structural role change (ρ = 0.340, p = 0.032). Hoffmann et al. [28] reported that chat-based AI interfaces reduce interruptions and improve perceived code quality at the individual level, but the present associations are at the team level. AI reducing the need for discussions was negatively correlated with higher-level problem solving (ρ = -0.422, p = 0.007), and AI-related conflict was negatively associated with it (ρ = -0.351, p = 0.026). Both associations are consistent with the mentorship conflict reported by Stray et al. [17] and Ferino et al. [18]. They reported that when AI replaces peer dialogue, teams lose the analytical engagement that human-to-human knowledge exchange produces.

Also, team size was positively associated with both development speed (β = 0.435, p = 0.019, Table 2a) and productivity (β = 0.326, p = 0.047, Table 2c) yet negatively associated with interpersonal trust (ρ = -0.320, p = 0.044, Table 3). Openness about AI reliability was positively correlated with team size (ρ = 0.488, p = 0.001), suggesting larger teams develop

more structured dialogue around AI limitations. This is consistent with Jensen et al. [29]. That is, larger organizations develop formalized AI expectation frameworks relative to the ad hoc adoption common in smaller teams. As team size increases, AI conflict management and structured communication practices become more consequential for delivery outcomes, particularly in South African teams where senior developer emigration reduces the experienced personnel available to anchor trust and mentorship.

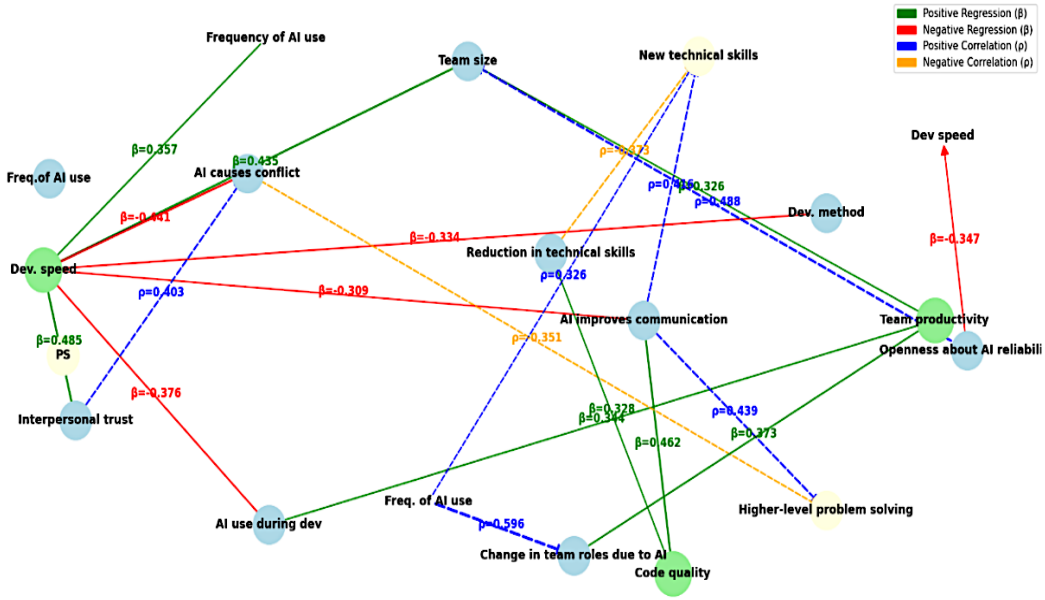


Figure 5. AI-team dynamics-SE outcomes relationships

Figure 5 shows development speed positioned between two opposing sets of associations: interpersonal trust and team size on the enabling side, and AI-related conflict, methodological constraints, and coordination overhead on the constraining side. Code quality and productivity appear in the upper portion of the figure, linked to communication improvement and role restructuring rather than to adoption frequency alone. The correlation pathways in the lower portion confirm that frequent AI use co-occurs with role transformation and shifting skill requirements. The figure should be read as a visual summary of associations, not a causal model.

No prior study in the reviewed literature examines AI adoption within South African SE teams using this combination of methods. The 97.5% adoption rate in this purposively sampled professional population is in line with social influence as a primary adoption

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

driver, as Shao and Ishengoma [21] report. The focus on GitHub Copilot at 75% may reflect training data density advantages for English-language codebases, consistent with Koyanagi et al. [30]. The ongoing trust and accuracy concerns in Theme 3 (n = 15, Table 4) show that South African teams face the same productivity-quality trade-off documented globally. Table 5 positions these findings against prior studies across six dimensions. Here, confirmations indicate directional consistency, contradictions indicate divergence, and extensions indicate dimensions where prior work left gaps. These findings are associations from a small purposive sample. However, whether they hold in a larger, more representative South African population remains to be established.

Table 5. Comparison of study findings against existing literature

Dimension	Study	Context	Method	Prior Finding	This Study	Relationship
Productivity / Development Speed	[6], [5], [23]	77,000 organisations (global meta-analysis)	Systematic review (47 studies)	45.6-point gap between perceived (31.4%) and measured (14.2%) productivity gains	AI adoption initially reduced development speed (β = −0.376, p = 0.025); frequency of use increased it (β = 0.357, p = 0.043)	Confirms: speed benefits are not immediate and depend on accumulated experience
Productivity / Development Speed	[11]	Large IT consulting firm (Brazil)	Longitudinal multi-case study (13 months)	59.1% increase in story-point value with flat code volume (P-A-E Divergence)	Interpersonal trust was the strongest predictor of development speed (β = 0.485, p = 0.017), exceeding all AI-specific variables	Extends: relational factors predict speed more strongly than AI tool variables in this sample
Code Quality / Security	[5], [6], [23]	Global	Systematic review	40–50% of AI-generated code contains exploitable vulnerabilities	AI improving team communication was the strongest predictor of code quality (β	Extends: code quality is predicted by team communication, not tool use alone

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Dimension	Study	Context	Method	Prior Finding	This Study	Relationship
					= 0.462, p = 0.003)	
Code Quality / Security	[25]	Public sector (Brazil)	Mixed-methods (K-Means/Gioia)	85% reduction in bug density reported	Theme 4 (n = 9) showed mixed quality impact; improvement depended on validation practice	Contradicts: self-reported quality gains in prior work are not replicated; quality effects are inconsistent in this sample
Role Transformation	[10]	German SE industry	Mixed-methods survey and interviews	Developers shift from code creation to validation; "Agent Architect" transition documented	95% of respondents reported role change; role change was a significant predictor of productivity (β = 0.373, p = 0.020)	Confirms: role transformation is structural and tied to productivity outcomes
Role Transformation	[18]	Cross-continental practitioners	Socio-technical grounded theory (22 interviews)	AI adoption requires reconfiguration of responsibilities to produce performance gains.	Frequency of AI use showed the strongest correlation with role change in the dataset (ρ = 0.596, p < 0.001)	Confirms: more intensive AI use is associated with greater structural role adaptation
Team Collaboration / Communication	[28]	Large software organisation	SPACE Framework Likert-scale analysis	Chat-based AI reduces interruptions and improves perceived code quality at the individual level.	AI improving communication predicted code quality at team level (β = 0.462, p = 0.003)	Extends: communication improvement predicts quality outcomes at team level, not only individually
Team Collaboration / Communication	[17]	Norwegian public sector	Mixed-methods (interviews and survey)	45% of users rely less on colleagues after AI adoption	AI reducing discussions was negatively correlated with higher-level problem solving (ρ = -	Confirms: reduced peer dialogue constrains analytical team performance

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Dimension	Study	Context	Method	Prior Finding	This Study	Relationship
					0.422, p = 0.007)	
Skill Shifts	[19]	Computer science students	Qualitative voice and screen recording	"Trust" finish type erodes skill formation; AI reduces human explanation episodes.	Skill reduction was negatively correlated with review frequency (ρ = -0.461, p = 0.003), indicating compensatory verification behaviour	Extends: teams aware of skill shifts compensate through increased verification activity, partially offsetting atrophy risk
Adoption Drivers	[21]	University of Dodoma (academic and professional)	UTAUT survey and regression	Social influence outweighs technical merit as the primary adoption driver	97.5% adoption rate in a purposively sampled South African professional population, with GitHub Copilot at 75%	Confirms: high adoption in the SA context is consistent with social influence as the primary driver

4.8. Strengths and Limitations

To the authors' knowledge, this study is the first to examine AI tool adoption and team dynamics within a South African SE population, addressing a gap in the literature concentrated in North American, European, and Brazilian settings. The mixed-methods design combined MLR and Spearman's rank correlation on the same quantitative dataset, allowing cross-validation of findings across two analytical techniques. Thematic analysis followed the six-phase procedure of Braun and Clarke [26], making the qualitative analysis replicable and auditable. Purposive sampling ensured all 40 respondents had direct experience with AI tools in production settings. Ethical approval was obtained from the NWU Research Ethics Committee, and all data handling complied with the POPIA.

Several limitations constrain the interpretation of the findings. The sample of 40 reduces the statistical power of the MLR models. Model 10 includes eight predictors against 40 cases, a ratio of five observations per predictor, which falls below the recommended threshold of 10-20 per predictor. All findings should therefore be treated as exploratory, and a post-hoc power analysis was not conducted. No correction for multiple

comparisons was applied to either the regression models or the Spearman correlations. With 15 correlations tested simultaneously, the probability of at least one false positive is elevated, and significance values should not be treated as individually confirmatory without replication. The survey instrument was developed from prior literature rather than validated against an established scale, and internal consistency was not formally assessed, as the survey combined single-item measures and mixed response formats that do not meet the assumptions required for Cronbach's alpha computation. Confidence intervals for regression coefficients were not computed in the original analysis. This is because generating them post-hoc risked introducing inconsistencies with the reported output, and standardised coefficients and standard errors are reported as sufficient for hypothesis-generating purposes.

All outcome variables, such as development speed, code quality, and team productivity, were respondent-assessed rather than extracted from repositories or project management systems, introducing self-report bias. Common method bias is a further threat. For instance, both independent and dependent variables were collected from the same respondent in the same survey, which may inflate observed correlations. Developers constituted 75% of the sample, and team leads 2.5%. Team-level interpretations, particularly those concerning leadership decisions, governance structures, and cross-role dynamics, rest primarily on developer perceptions and should be read as developer-reported accounts rather than team-level measurements. Participants were recruited through LinkedIn and developer community forums, which introduces self-selection bias and overrepresents urban, formally employed developers. The study is cross-sectional and cannot establish causality or track skill atrophy or role change over time. The qualitative strand used open-ended survey responses rather than interviews, which produced thematic patterns without the contextual depth that semi-structured interviews provide. Qualitative coding was conducted by the primary researcher and independently checked by a second reviewer across a 25% subset. In this case, where disagreements arose, consensus was reached through discussion, and ambiguous responses were excluded from theme frequency counts.

5. CONCLUSION

This study examined how AI-assisted development tools relate to team dynamics and software delivery outcomes among 40 South African software development professionals. We used a mixed-methods design that combines structured survey data with thematic analysis. Within this purposive sample, AI tool adoption was near-universal and concentrated in generative code-assistance tools, while governance and quality-focused tools were used by fewer than 16% of respondents. The analysis of the findings shows three consistent patterns across both strands. Interpersonal trust showed the strongest positive association with development speed among all predictors, exceeding every AI-specific variable in the model, while AI-related conflict showed the strongest negative association. This suggests that relational conditions matter more to delivery speed than tool adoption alone. AI use at adoption onset was negatively associated with development speed, while frequency of use was positively associated with it, a pattern consistent with a learning-curve effect that short-window ROI assessments would miss. Also, role change due to AI was positively associated with team productivity, while teams that did not reorganise responsibilities showed weaker delivery outcomes. These findings are exploratory in nature. The sample of 40 limits statistical power, the cross-sectional design precludes causal inference, and self-reported measures introduce response bias. To the authors' knowledge, no prior published study has examined this combination of factors within a South African SE population, and these results are best read as a baseline for future work. Larger stratified samples, longitudinal designs tracking skill formation under sustained AI reliance, and dedicated investigation of leadership factors in AI adoption are needed to confirm or revise these patterns, including a literature review.

REFERENCES

[1] N. Gupta, "The Rise of AI Copilots: Redefining Human-Machine Collaboration in Knowledge Work," *Int. J. Humanities Inf. Technol.*, vol. 7, no. 3, 2025. https://doi.org/10.21590/

[2] S. Panyam and P. Gujar, "How AI Agents Are Transforming Software Engineering and the Future of Product Development," *Computer,* vol. 58, no. 5, pp. 71–77, May 2025, doi: 10.1109/MC.2024.3488378.

[3] B. Tabarsi et al., "LLMs' Reshaping of People, Processes, Products, and Society in Software Development: A Comprehensive Exploration with Early Adopters," *arXiv*, preprint. arXiv:2503.05012, 2025.

[4] S. Sarkar, "The Effect of AI Tools on Modern Software Development for Frontend Engineering: An Empirical Analysis," *SSRN Electronic Journal*, Jan. 2025, doi: 10.2139/ssrn.5442494.

[5] R. Mo et al., "Assessing and Analyzing the Correctness of GitHub Copilot's Code Suggestions," *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 7, Art. no. 194, Sep. 2025, doi: 10.1145/3715108.

[6] C. Johnson et al., "The AI Productivity Paradox in Software Development: A Meta-Analysis of Implementation Outcomes and ROI Patterns (2024-2025)," *SSRN Electronic Journal*, 2025, doi: 10.2139/ssrn.5902163.

[7] Z. S. Li, N. N. Arony, A. M. Awon, D. Damian, and B. Xu, "AI Tool Use and Adoption in Software Development by Individuals and Organizations: A Grounded Theory Study," *arXiv,* preprint*,* arXiv:2406.17325, 2024.

[8] M. O. Ahmad, H. Ghanbari, T. Gustavsson, and B. R. Upreti, "It all starts with structure: Investigating learning dynamics in large-scale agile software development," *J. Syst. Softw.*, vol. 230, p. 112561, 2025, doi: 10.1016/j.jss.2025.112561.

[9] L. M. Restrepo-Tamayo, G. P. Gasca-Hurtado, and J. Valencia-Calvo, "Characterizing Social and Human Factors in Software Development Team Productivity: A System Dynamics Approach," *IEEE Access*, vol. 12, pp. 59739–59755, 2024, doi: 10.1109/ACCESS.2024.3388505.

[10] T. Felder et al., "Adoption of Generative Artificial Intelligence in the German Software Engineering Industry: An Empirical Study," *arXiv,* preprint, arXiv:2601.16700, 2026.

[11] R. Tomaz, P. Guenes, A. A. Araújo, M. T. Baldassarre, and M. Kalinowski, "Impacts of Generative AI on Agile Teams' Productivity: A Multi-Case Longitudinal Study," *arXiv*, preprint, arXiv:2602.13766, 2026.

[12] S. Maatta, "*How Do Programmers Evaluate AI-Generated Code?*" in Proc. 2025 ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM), Honolulu, HI, USA, 2025, pp. 522–527, doi: 10.1109/ESEM64174.2025.00057.

[13] T. Vhahangwele, B. Isong, and A. A. Mahfouz, "The Impact of Team Dynamics on Software Quality and Productivity: Evidence from South Africa," *Journalisi,* vol. 8, no. 1, pp. 809–840, Mar. 2026, doi: 10.63158/journalisi.v8i1.1438.

[14] P. Diebold, "From backlogs to bots: Generative AI's impact on agile role evolution," *J. Softw.: Evol. Process*, vol. 37, no. 1, p. e2740, 2025, doi: 10.1002/smr.2740.

[15] J. Zhou et al., "Exploring the problems, their causes and solutions of AI pair programming: A study on GitHub and Stack Overflow," *J. Syst. Softw.,* vol. 219, p. 112204, 2025, doi: 10.1016/j.jss.2024.112204.

[16] W. Mendes, S. Souza, and C. de Souza, "*You're on a bicycle with a little motor": Benefits and Challenges of Using AI Code Assistants*," in Proc. 2024 IEEE/ACM 17th Int. Conf. Cooperative Hum. Aspects Softw. Eng. (CHASE), Lisbon, Portugal, 2024, pp. 45–56, doi: 10.1145/3643750.3643758.

[17] V. Stray, A. Barbala, and V. T. Wivestad, "*Human-AI Collaboration in Software Development: A Mixed-Methods Study of Developers'* Use of GitHub Copilot and ChatGPT," in Companion Proc. 33rd ACM Int. Conf. Foundations Softw. Eng. (FSE Companion '25), New York, NY, USA: ACM, 2025, pp. 1325–1332, doi: 10.1145/3696630.3730566.

[18] S. Ferino, R. Hoda, J. Grundy, and C. Treude, "Walking the Tightrope of LLMs for Software Development: A Practitioners' Perspective," *arXiv*, preprint, arXiv:2511.06428, 2025.

[19] A. Welter et al., "*An Empirical Study of Knowledge Transfer in AI Pair Programming*," in Proc. 2025 40th IEEE/ACM Int. Conf. Autom. Softw. Eng. (ASE), 2025, pp. 210–222, doi: 10.1109/ASE62340.2025.00021.

[20] T. Chen, "*The impact of AI-pair programmers on code quality and developer satisfaction: Evidence from Timi Studio*," in Proc. 2024 Int. Conf. Generative Artif. Intell. Inf. Secur. (GAIIS), 2024, pp. 88–94, doi: 10.1109/GAIIS62410.2024.00015.

[21] D. Shao and F. Ishengoma, "Empirical Analysis of Generative AI Tool Adoption in Software Development," Inf. Softw. Technol., vol. 182, p. 108036, Jun. 2026, doi: 10.1016/j.infsof.2025.108036.

[22] A. Kumar et al., "Intuition to Evidence: Measuring AI's True Impact on Developer Productivity," *arXiv*, arXiv:2509.19708, 2025.

[23] E. Paradis et al., "*How much does AI impact development speed? An enterprise-based randomized controlled trial*," in Proc. 2025 IEEE/ACM 47th Int. Conf. Softw. Eng.: Softw. Eng. Practice (ICSE-SEIP), 2025, pp. 312–323, doi: 10.1109/ICSE-SEIP64321.2025.00034.

[24] M. Muñoz et al., "Comparative study of AI code generation tools: Quality assessment and performance analysis," *Lat. Am. J. Inf. Technol. (LatIA),* vol. 2, pp. 104–104, 2024.

[25] M. E. Souza and E. C. S. Murillo, "How Generative Artificial Intelligence Tools Can Improve the Quality of Software Produced by Software Development Teams," J. *Softw. Eng. Res. Dev.,* vol. 13, no. 1, pp. 45–58, 2025, doi: 10.5753/jserd.2025.4125.

[26] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qual. Res. Psychol.*, vol. 3, no. 2, pp. 77–101, 2006, doi: 10.1191/1478088706qp063oa.

[27] S. K. Ahmed et al., "Using thematic analysis in qualitative research," J. *Med., Surgery, Public Health,* vol. 6, p. 100198, 2025, doi: 10.1016/j.glmedi.2025.100198.

[28] P. Hoffmann, D. Zercher, and T. C. Schimmer, "*How AI communication capabilities affect developer productivity in AI pair programming: Insights from a large software development organization*," in Research-in-Progress Papers, European Conference on Information Systems (ECIS 2025 TREOs), Paper 23, 2025. [Online]. Available: https://aisel.aisnet.org/treos_ecis2025/23

[29] V. V. Jensen, A. Alami, A. R. Bruun, et al., "Managing expectations towards AI tools for software development: a multiple-case study," *Inf. Syst. E-Bus. Manage.*, vol. 23, pp. 869–901, 2025, doi: 10.1007/s10257-025-00704-7.

[30] K. Koyanagi et al., "*Exploring the effect of multiple natural languages on code suggestion using GitHub Copilot*," in Proc. 21st Int. Conf. Min. Softw. Repositories (MSR), Lisbon, Portugal, 2024, pp. 115–126, doi: 10.1145/3643990.3644012.