POLYPUBLIE

Attention to innovation: linking developer gaze and coding approaches

Yahya Anwar Lafhal, Zohreh Sharafi, Mahta Amini

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

Date	2026
Type	Communication de conférence / Conference or Workshop Item Communication écrite/Paper
Version	Version officielle de l'éditeur / Published version
URL de PolyPublie **PolyPublie URL**	https://publications.polymtl.ca/80445/
Évaluation par les pairs **Peer reviewed**	Révisé par les pairs / Refereed
Conditions d'utilisation **Terms of use**	Creative Commons Attribution-Utilisation non commerciale-Pas d'oeuvre dérivée 4.0 International / Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND)
Référence **Citation**	Lafhal, Y. A., Sharafi, Z., & Amini, M. (avril 2026). Attention to innovation: linking developer gaze and coding approaches [Communication écrite]. 34th International Conference on Program Comprehension (ICPC 2026), Rio de Janeiro, Brazil. https://doi.org/10.1145/3794763.3794790

Publication officielle
Official publication

Nom de la conférence **Conference name**	34th International Conference on Program Comprehension (ICPC 2026)
Date et lieu **Date and location**	2026-04-12 - 2026-04-13, Rio de Janeiro, Brazil
Maison d'édition **Publisher**	Association for Computing Machinery
URL officielle **Official URL**	https://doi.org/10.1145/3794763.3794790
Mention légale **Legal notice**	This work is licensed under a Creative Commons Attribution-NonCommercialNoDerivatives 4.0 International License (https://creativecommons.org/licenses/by-nc-nd/4.0).

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal
https://publications.polymtl.ca

Attention to Innovation: Linking Developer Gaze and Coding Approaches

Yahya Lafhal
Polytechnique Montreal
Montreal, Canada
yahya-anwar.lafhal@polymtl.ca

Zohreh Sharafi
Polytechnique Montreal
Montreal, Canada
zohreh.sharafi@polymtl.ca

Mahta Amini
Polytechnique Montreal
Montreal, Canada
mahta.amini@polymtl.ca

Abstract

Creativity is widely seen as a key quality of software developers, yet empirical software engineering rarely examines its manifestation in coding. We take a process-level view by linking psychometric creativity assessments to fine-grained eye-tracking and IDE interaction data from a controlled game implementation task. In a laboratory study with 40 Python-proficient participants implementing scoring logic for a Pong game, we derive programming strategy profiles from joint gaze and code editing behavior and relate them to scores on three well-validated creativity tests.

We identify three recurring strategies that differ in IDE interactions, gaze behavior, and the level of visual and structural elaboration. The efficient and functionally focused group shows consistently higher divergent scores and is characterized by a compact, targeted implementation style. These participants focus on a small number of well-placed changes to ensure correct game mechanics, rather than settling for a partial solution or investing heavily in decorative enhancements. Our results offer early evidence of how creativity appears in programming behavior and point to the potential for creativity-aware tools and pedagogical approaches.

CCS Concepts

• **Software and its engineering → Software creation and management**; • **Human-centered computing → Empirical studies in HCI**.

Keywords

Human factors, Software Engineering, Creativity, and problem-solving strategies, Empirical research, Eye tracking

ACM Reference Format:
Yahya Lafhal, Zohreh Sharafi, and Mahta Amini. 2026. Attention to Innovation: Linking Developer Gaze and Coding Approaches. In *34th IEEE/ACM International Conference on Program Comprehension (ICPC '26), April 12–13, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3794763.3794790

1 Introduction

Creativity is often framed as a defining quality of skilled software developers [12]. They are expected to design novel solutions, navigate constraints, and adapt their code to meet changing requirements. However, most empirical studies in software engineering (SE) either do not explicitly address creativity or only infer it from the final product, assessing how novel, useful, or high-quality a design or solution appears, rather than how developers engage with the code during the process. What is largely missing is a process-level understanding of creativity: how, if at all, does a more creative developer actually behave differently while programming?

Without connecting creativity to the concrete, moment-to-moment aspects of programming work, it becomes challenging to design tools or educational practices that genuinely support creative strengths, rather than merely rewarding speed or superficial polish.

In this paper, we take a creativity-centered view of programming behavior and present findings from a controlled human experiment with 40 participants. We combine established psychometric creativity measures with fine-grained eye-tracking and IDE interaction logs collected during a controlled software implementation task. Our goal is not only to discover clusters of "styles" of programming, but also to determine whether those styles exhibit a recognizable creative signature. We derive programming strategy profiles from joint gaze and code-editing behavior, and then examine how these profiles relate to divergent and convergent thinking, which together influence creative outcomes.

We identify three recurring programming strategy profiles that differ in how far participants push the core functionality and how much they elaborate on the visual and structural aspects of the game. Only one of these profiles(the efficient, functionally focused group) shows consistently higher divergent-association scores. The key pattern that emerges is that higher divergent-association creativity aligns with a compact, targeted implementation style: participants with higher creativity scores tend to focus on getting the core mechanics right with a small number of well-placed edits and relatively economical visual engagement, in contrast to peers who either stop at a partial solution or devote substantial effort to visual embellishments. Our findings offer an initial, process-level account of how creativity manifests in coding. They also highlight the need for creativity-aware models, tools, and teaching methods that recognize that more creative developers may literally look and behave differently when programming.

2 Related Work

Creativity, defined by originality and usefulness, has been widely studied in psychology [9, 30, 34], notably through Mooney's 4P [26], Amabile's framework [1], and Guilford's divergent thinking [13].

In SE, creativity underpins problem solving and design. Early research explored techniques such as brainstorming and creative triggers in requirements engineering [5, 10], while later studies examined how personality, motivation, and context shape creative



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
ICPC '26, Rio de Janeiro, Brazil
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2482-4/26/04
https://doi.org/10.1145/3794763.3794790

outcomes in practice [2]. Recent studies [3, 27] link creativity scores to programming behavior, with highly creative developers writing more diverse, expressive code. Emerging methods such as eye-tracking and generative AI are opening new perspectives on creativity. Eye-tracking research shows that gaze behavior can signal creative modes and moments of insight [17, 20, 24, 32]. At the same time, as generative AI becomes embedded in technical workflows, researchers are examining how human–AI collaboration influences innovation across domains [7, 14, 16, 19, 21, 23, 31]. Together, this work expands understandings of creativity from individual cognition to dynamic human–AI co-creation.

A parallel line of research has focused on modeling developers' behaviors to reveal underlying patterns in how they work. Prior work has identified behavioral patterns through EEG and eye-tracking [29], modeled planning behavior with Hidden Markov Models [15], and grouped developers using K-means for team and practice insights [22]. Other research has clustered developers based on code contributions [11], examined behavioral differences across open-source projects [6], and analyzed contributor roles and community dynamics [4]. In parallel, clustering has also been used in game development to reveal player behavior profiles [8].

We extend prior work by clustering multi-source coding behaviors and examining their relationship with creativity scores.

3 Experimental Methodology

We ran an eye-tracking study to identify programming strategy profiles and explore their links to creativity.

3.1 Procedure

We conducted the study in a controlled laboratory environment with minimal distractions. Participants used the PyCharm IDE while a Tobii Pro Fusion eye tracker (250 Hz) captured gaze data. IDE interactions were recorded through CodeGRITS. To protect privacy and ensure data confidentiality, each participant was assigned a randomized, anonymized ID, which was used for all recordings and analyses. All collected data have been made accessible here [1].

The experiment consisted of three stages. In the first stage, participants completed three creativity assessments. The Divergent Association Task (DAT) [28] required them to generate ten semantically distant words, providing an index of divergent associative thinking. The Remote Associates Test (RAT) [25] measured convergent insight: for each item, participants were given a triad of cue words (e.g., "safety", "cushion", "point") and asked to produce a single word that forms a compound or close association with all three (e.g., "pin"). Finally, participants completed the figural Torrance Test of Creative Thinking (TTCT) [36], in which they transformed 30 blank circles into unique drawings within a set time limit to evaluate non-verbal aspects of creativity.

In the second stage, each participant was given a partially implemented version of the classic Pong game with the scoring logic removed and was asked to implement the missing functionality without any external AI-based assistance. The task order was counterbalanced across participants. We also scheduled short rest breaks to reduce visual fatigue. Finally, participants completed a post-experiment questionnaire, to report perceived cognitive workload and provide open-ended feedback about the study.

[1] https://doi.org/10.6084/m9.figshare.31198186

3.2 Participants and Recruitment

In our IRB-approved study, we recruited 40 participants (85% male, 15% female) from undergraduate and graduate computer science programs. We applied a rigorous screening process to include only those with proficient Python skills. All participants had prior experience with Python, with an average of 3.75 years of use (standard deviation, SD = 2.23). Participants provided informed consent under institutional ethics approval and received compensation.

3.3 Experiment Measures

Code navigation and coding-strategy measures: During the programming task, we captured how participants navigated and modified the code. Using CodeGRITS IDE logs and Tobii Pro Fusion eye tracker, we collected code-navigation metrics such as *total task time*, *on-screen and idle time*, *fixation counts and durations*, and aggregated measures of *attention, navigation, and IDE interaction*. In parallel, we extracted coding-strategy metrics describing what participants changed in the codebase, including scoring-logic accuracy, edit counts, and modifications to the timer, sound, color, and animations, as well as structural code metrics such as the number of files modified and lines of code changed.

Psychometric creativity measures: Participants completed three standardized creativity tests, as described in 3.1. Together, these scores provide complementary indices of divergent associative thinking, convergent insight, and visual–figurative creativity. In our analysis, these psychometric scores were treated as external outcome variables and were not used as inputs to the clustering.

4 Analysis and Results

This section reports our results and answers two research questions:

RQ1. What programming strategy profiles emerge when participants are clustered by their code-navigation behavior and coding strategies during the game task?

RQ2. How do these programming strategy profiles differ in their creativity scores?

4.1 Data Processing and Preparation

We began from a merged dataset with 39 participants and 86 features, after excluding one participant whose data was corrupted.

We converted all candidate features to numeric types, treating non-parseable entries as missing. Missing values were then imputed using each feature's median, allowing us to retain all 39 participants while using a summary statistic robust to outliers and suitable for binary, count, and continuous variables. Next, we removed near–zero-variance features (threshold = 0.01), as features that are nearly constant across participants add noise and contribute little to cluster separation. The remaining code navigation and coding strategy features were standardized separately. Finally, to avoid overfitting with 39 participants and 67 behavioral features, we applied Principal Component Analysis (PCA) [18] within each subset (eye and code) to reduce dimensionality and denoise the

Table 1: Kruskal–Wallis tests for representative coding-navigation features that differ significantly across the clusters. Means are shown for each cluster: C_0 = Minimal Implementers, C_1 = Efficient Functional Implementers, C_2 = Elaborative Implementers.

#	Feature	Description	Type	H	p	ε^2	Mean_{C0}	Mean_{C1}	Mean_{C2}
1	scoring_correct_change_num	Scoring changes implemented correctly (count)	Code	24.69	$< .001$	0.63	0.35	2.38	2.00
2	score_accuracy	Scoring functionality implemented correctly (0/1)	Code	22.66	$< .001$	0.57	0.22	1.00	1.00
3	va_total_change_num	Total visual/audio (VA) changes (count)	Code	21.34	$< .001$	0.54	0.48	3.00	2.50
4	clock_total_change_num	Total timer changes (count)	Code	16.57	$< .001$	0.40	0.13	1.25	0.25
5	sound_total_change_num	Total sound changes (count)	Code	12.44	$< .05$	0.29	0.00	1.13	0.75
6	LOC_ball	Lines of code in `ball.py`	Code	12.25	$< .05$	0.28	39.00	39.00	40.13
7	animation_total_change_num	Total animation changes (count)	Code	9.32	$< .05$	0.20	0.04	0.13	0.50
8	total_fixation_time_s	Total fixation time on task (s)	Eye	8.43	$< .05$	0.18	233.27	99.93	451.18

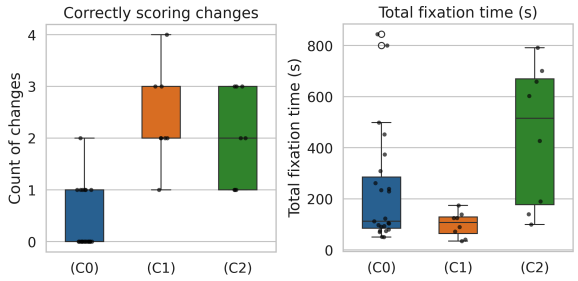


Figure 1: Distributions of correctly implemented scoring strategy changes and total fixation time across the three clusters. C_0 = Minimal Implementers, C_1 = Efficient Functional Implementers, C_2 = Elaborative Implementers.

feature space. We then concatenated the resulting eye and code behavior components to form the joint feature used for clustering.

4.2 Hierarchical Clustering

To identify behavioral patterns, we applied hierarchical clustering with Ward's linkage on standardized features. We varied the number of principal components for navigation and strategy features, as well as the number of clusters ($k \in \{2, 3, 4, 5\}$). The best silhouette score (0.48) was obtained with one eye and two code components at $k = 4$, closely followed by $k = 3$ (0.46).

Because the four-cluster solution included two small and highly similar groups, leading to unstable and underpowered comparisons, we merged them and retained a more interpretable three-cluster solution. We label the final clusters as C_0 (Minimal Implementers), C_1 (Efficient Functional Implementers), and C_2 (Elaborative Implementers).

4.3 RQ1: Emergent strategy profiles

Coding strategy features: We compared the three clusters on a set of coding-strategy features. These metrics capture (i) how participants modified the scoring logic, (ii) visual and auditory changes related to the game (*e.g.,* adding a timer or sound effects), and (iii) code-level properties such as code-level metrics like file changes, lines changed, and correctness of scoring and game elements. We applied the Kruskal–Wallis test for our small, unequal, non-normal clusters and reported effect sizes using ε^2.

As shown in Table 1, scoring behavior shows the strongest separation across clusters. Correct scoring edits differ markedly ($H = 24.69$, $p < .001$, $\varepsilon^2 = 0.63$): *"Minimal Implementers"*(C_0) make almost none ($M \approx 0.35$), while *"Efficient Functional"*(C_1) and *"Elaborative Implementers"*(C_2) make about two ($M \approx 2.38$, $M \approx 2.00$). Final scoring accuracy follows the same pattern: C_0 rarely achieve a correct score ($M \approx 0.22$), whereas C_1 and C_2 almost always do ($M = 1.00$). Figure 1 illustrates two representative metrics.

In our initial set of features, we included visual or auditory(VA) changes that are modifications that alter what the player sees or hears. Examples include updating the timer, adding sound effects, changing colors, or adding animations.

Results show that *"Minimal Implementers"* make very few VA changes overall ($M \approx 0.48$), *"Efficient Functional Implementers"* make the most ($M \approx 3.00$), and *"Elaborative Implementers"* fall in between, but still far above C_0 ($M \approx 2.50$). These results indicate that *"Minimal Implementers"* tend to stop at a partial or baseline implementation, while the other two actively refine and extend the scoring behavior. More fine-grained VA features align with this pattern. Clock measures (*e.g.,* total changes, accuracy, timer use), sound cues (*e.g.,* presence and correctness), and visual embellishments (*e.g.,* color changes and animations) all differ significantly across clusters with medium-to-large effect sizes.

Overall, the coding strategy features reveal three clear profiles. *"Minimal Implementers"*(C_0) produce only a partial game, with limited scoring and VA work and almost no timer, sound, or visual additions. *"Efficient Functional Implementers"*(C_1) focus on core mechanics, making many correct scoring and VA edits (*e.g.,* a working clock and key sounds) but little cosmetic polish. *"Elaborative Implementers"*(C_2) also achieve correct scoring while adding extra visual and structural enhancements beyond basic functionality.

Coding navigation features: We focused on eye-tracking metrics that summarize how participants visually engaged with the task. Only (`total_fixation_time_s`) shows a reliable cluster difference ($H \approx 8.43$, $p \approx .015$, $\varepsilon^2 \approx 0.18$; see the right panel of Figure 1). *"Efficient Functional Implementers"* spend the least time fixating on the screen ($M \approx 100$ s), *"Minimal Implementers"* are in the middle ($M \approx 233$ s), and *"Elaborative Implementers"* spend by far the most time ($M \approx 451$ s). Other time-based and eye metrics do not differ significantly between clusters. Along with the coding strategy results,

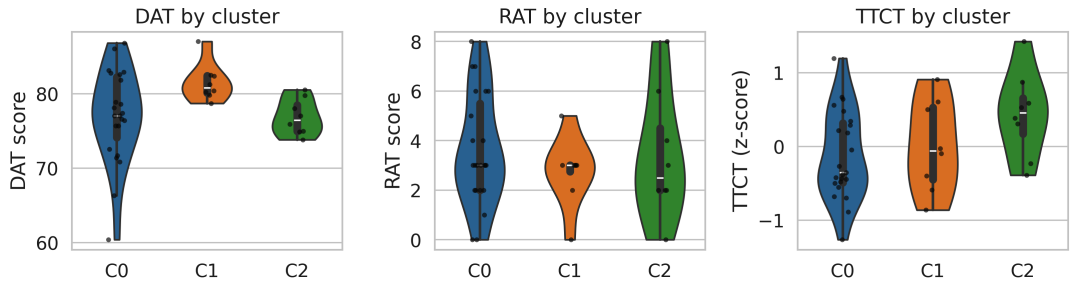


Figure 2: Distributions of creativity test scores by cluster. Each panel shows DAT, RAT, or TTCT score values for clusters C0–C2, with violin plots indicating the distribution of scores and overlaid points showing individual participants.

we found that the three profiles are defined primarily by *what* participants choose to implement and revise in the code, rather than by fine-grained differences in their moment-to-moment viewing patterns. The one robust navigation signal (`total_fixation_time_s`) aligns with the coding strategies: *"Efficient Functional Implementers"* appear to work in a relatively focused and economical way, while *"Elaborative Implementers"* invest substantially more visual time, consistent with their broader and more decorative editing behavior.

4.4 RQ2: Creativity Differences

To address RQ2, we examined how the three programming strategy profiles relate to psychometric creativity scores (DAT, RAT, TTCT) as summarized in Figure 2. For DAT, the Kruskal–Wallis test showed a significant cluster effect ($H = 6.60$, $p \approx .036$, $\varepsilon^2 \approx 0.13$). *"Efficient Functional Implementers"*(C_1) achieved the highest scores ($M \approx 81.52$, $SD \approx 2.56$, $n = 8$), whereas *"Minimal"*(C_0) and *"Elaborative Implementers"*(C_2) showed similar, lower scores ($M \approx 76.98$, $SD \approx 6.24$, $n = 23$; $M \approx 76.86$, $SD \approx 2.42$, $n = 8$). Dunn post-hoc tests suggested that C_1 differs from both C_0 and C_2 (uncorrected $p \approx .028$ and $p \approx .017$), while C_0 and C_2 do not differ ($p \approx .48$). The standardized difference between C_1 and C_0 is large (Cohen's $d \approx 0.81$), indicating that the cluster with accurate, focused implementation also has higher scores. By contrast, the RAT did not exhibit meaningful cluster differences, as we have very similar means across clusters. For TTCT scores, we observed only a trend-level effect. ($Kruskal-Wallis : p \approx .088$). *"Minimal Implementers"* showed lower TTCT scores ($M \approx -0.12$), *"Efficient Functional Implementers"* were near the mean ($M \approx 0.01$), and *"Elaborative Implementers"* scored higher ($M \approx 0.43$).

5 Threats to Validity

Several factors may affect the validity of our findings. To reduce hypothesis guessing, we withheld the specific research goals and asked participants not to discuss the study, while clearly explaining the procedure, tasks, and use of the eye tracker. To limit Hawthorne effects, we emphasized that only eye movements (not video) were recorded, and that the experimenter maintained a distance. We also administered a post-task coding–skill self-assessment to mitigate stereotype threat [33, 35]. We used a video-based eye tracker calibrated individually to ensure accurate capture with natural head movement. We also relied on standard statistical methods, established eye-tracking metrics, and validated creativity tests. Because creativity and cognitive processes are multifaceted, we triangulated across creativity scores, gaze patterns, and IDE interactions, focusing on convergent evidence rather than single indicators.

Our sample consisted of experienced student developers, appropriate for near-term professionals but less generalizable to experts. The controlled lab setup improves internal validity but limits ecological realism; replications across languages, tools, and professional populations are needed. Finally, although typical for eye-tracking studies, the sample size constrains the granularity and generalizability of the analyses.

6 Conclusion and Future Work

Our study positions creativity as more than an abstract trait measured on a separate test: it manifests in how individuals navigate through code, determine what changes to make, and decide when to stop. By combining psychometric creativity scores with fine-grained gaze data and editing traces, we find that higher divergent-association creativity aligns with a distinct programming style. Participants with stronger DAT scores tend to adopt a lean, targeted implementation style: they focus on getting the critical mechanics correct, make a small number of well-directed edits, and spend relatively less time visually dwelling on the code than peers who either stop early or invest heavily in embellishments. Thus, creative behavior does not simply mean touching more files or adding more features; it looks like selectively investing effort where it most directly advances a working solution.

Future work should broaden this creativity-centered view of programming by expanding both the populations and tasks studied. Larger and more diverse samples, richer activities (e.g., debugging, refactoring, AI-assisted development), and multi-session designs would help test the stability of creativity-linked behaviors across contexts and time. Beyond behavioral and psychometric data, future studies could combine creativity scores with neural measures (*e.g.,* fMRI, fNIRS) to link divergent thinking to concrete patterns of attention and implementation. Ultimately, this line of research can inform tools and educational practices that recognize and support different creative styles in programming rather than assuming a single "best" way to code.

Acknowledgments

We thank the participants for their contribution. This work was supported by NSERC (Grant No. RGPIN-2024-04819).

References

[1] Teresa M Amabile et al. 1988. A model of creativity and innovation in organizations. *Research in organizational behavior* 10, 1 (1988), 123–167.

[2] Aamir Amin, Shuib Basri, Mobashar Rehman, Luiz Fernando Capretz, Rehan Akbar, Abdul Rehman Gilal, and Muhammad Farooq Shabbir. 2020. The impact of personality traits and knowledge collection behavior on programmer creativity. *Information and Software Technology* 128 (2020), 106405. doi:10.1016/j.infsof.2020.106405

[3] Mahta Amini, Jay Olson, and Zohreh Sharafi. 2024. Coding with a Creative Twist: Investigating the Link Between Creativity Scores and problem-solving Strategies. In *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results* (Lisbon, Portugal) *(ICSE-NIER'24)*. Association for Computing Machinery, New York, NY, USA, 21–25. doi:10.1145/3639476.3639766

[4] Jinghui Cheng and Jin LC Guo. 2019. Activity-based analysis of open source software contributors: Roles and dynamics. In *2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 11–18.

[5] Jacob Cybulski, Lemai Nguyen, Theerasak Thanasankit, and Sharman Lichtenstein. 2003. *Understanding problem solving in requirements engineering: Debating creativity with is practitioners*. Deakin University.

[6] Enrico Di Bella, Alberto Sillitti, and Giancarlo Succi. 2013. A multivariate classification of open source developers. *Information Sciences* 221 (2013), 72–83.

[7] Anil R Doshi and Oliver P Hauser. 2023. Generative artificial intelligence enhances creativity but reduces the diversity of novel content. *arXiv preprint arXiv:2312.00506* (2023).

[8] Anders Drachen, Rafet Sifa, Christian Bauckhage, and Christian Thurau. 2012. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *2012 IEEE conference on Computational Intelligence and Games (CIG)*. IEEE, 163–170.

[9] Berys Gaut. 2010. The philosophy of creativity. *Philosophy Compass* 5, 12 (2010), 1034–1046.

[10] Benito Giunta, Corentin Burnay, Neil Maiden, and Stephane Faulkner. 2022. Creativity triggers: extension and empirical evaluation of their effectiveness during requirements elicitation. *Journal of Systems and Software* 191 (2022), 111365.

[11] Cristina Aguilera González, Laia Albors Zumel, Jesús Antonanzas Acero, Valentina Lenarduzzi, Silverio Martínez-Fernández, and Sonia Rabanaque Rodríguez. 2021. A preliminary investigation of developer profiles based on their activities and code quality: Who does what?. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 938–945.

[12] Wouter Groeneveld, Laurens Luyten, Joost Vennekens, and Kris Aerts. 2021. Exploring the role of creativity in software engineering. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 1–9.

[13] Joy Paul Guilford. 1950. Creativity. *American psychologist* 5, 9 (1950), 444.

[14] Niklas Holzner, Sebastian Maier, and Stefan Feuerriegel. 2025. Generative AI and Creativity: A Systematic Literature Review and Meta-Analysis. *arXiv preprint arXiv:2505.17241* (2025).

[15] Verena Honsel, Steffen Herbold, and Jens Grabowski. 2016. Hidden markov models for the prediction of developer involvement dynamics and workload. In *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*. 1–10.

[16] VICTORIA JACKSON, BOGDAN VASILESCU, DANIEL RUSSO, PAUL RALPH, RAFAEL PRIKLADNICKI, MALIHEH IZADI, SARAH D'ANGELO, SARAH INMAN, ANIELLE ANDRADE, and ANDRÉ VAN DER HOEK. 2025. The Impact of Generative AI on Creativity in Software Development: A Research Agenda. (2025).

[17] Dorota M Jankowska, Marta Czerwonka, Izabela Lebuda, and Maciej Karwowski. 2018. Exploring the creative process: Integrating psychometric and eye-tracking approaches. *Frontiers in Psychology* 9 (2018), 1931.

[18] Ian Jolliffe. 2011. Principal component analysis. In *International encyclopedia of statistical science*. Springer, 1094–1096.

[19] Jeongyeon Kim, Sangho Suh, Lydia B Chilton, and Haijun Xia. 2023. Metaphorian: Leveraging large language models to support extended metaphor creation for science writing. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference*. 115–135.

[20] E Kwon, JD Ryan, A Bazylak, and LH Shu. 2020. Does visual fixation affect idea fixation? *Journal of Mechanical Design* 142, 3 (2020), 031118.

[21] Mina Lee, Percy Liang, and Qian Yang. 2022. Coauthor: Designing a human-ai collaborative writing dataset for exploring language model capabilities. In *Proceedings of the 2022 CHI conference on human factors in computing systems*. 1–19.

[22] Xiaozhou Li. 2019. Research on software project developer behaviors with k-means clustering analysis. *SSSME 2019: Joint Proceedings of the Summer School on Software Maintenance and Evolution* (2019).

[23] Ryan Louie, Andy Coenen, Cheng Zhi Huang, Michael Terry, and Carrie J Cai. 2020. Novice-AI music co-creation via AI-steering tools for deep generative models. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–13.

[24] Saurabh Maheshwari, Viplav Tuladhar, Tsering Thargay, Pallavi Sarmah, Palakshi Sarmah, and Kushal Rai. 2022. Do our eyes mirror our thought patterns? A study on the influence of convergent and divergent thinking on eye movement. *Psychological Research* 86, 3 (2022), 746–756.

[25] Sarnoff A Mednick. 1968. The remote associates test. *The Journal of Creative Behavior* (1968).

[26] Ross L Mooney. 1963. A conceptual model for integrating four approaches to the identification of creative talent. *Scientific creativity: Its recognition and development* (1963), 331–340.

[27] Anthonia Njoku, Mahta Amini, and Zohreh Sharafi. 2024. Innovating Coding: Evaluating the Impact of Innovative Thinking in Programming. In *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension* (Lisbon, Portugal) *(ICPC '24)*. Association for Computing Machinery, New York, NY, USA, 241–245. doi:10.1145/3643916.3644397

[28] Jay A Olson, Johnny Nahas, Denis Chmoulevitch, Simon J Cropper, and Margaret E Webb. 2021. Naming unrelated words predicts creativity. *Proceedings of the National Academy of Sciences* 118, 25 (2021), e2022340118.

[29] Norman Peitek, Annabelle Bergum, Maurice Rekrut, Jonas Mucke, Matthias Nadig, Chris Parnin, Janet Siegmund, and Sven Apel. 2022. Correlates of programmer efficacy and their link to experience: A combined EEG and eye-tracking study. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 120–131.

[30] ED Petkus Jr. 1996. The creative identity: Creative behavior from the symbolic interactionist perspective. *The Journal of Creative Behavior* 30, 3 (1996), 188–196.

[31] Marissa Radensky, Simra Shahid, Raymond Fok, Pao Siangliulue, Tom Hope, and Daniel S Weld. 2024. Scideator: Human-llm scientific idea generation grounded in research-paper facet recombination. *arXiv preprint arXiv:2409.14634* (2024).

[32] Carola Salvi, Emanuela Bricolo, Steven L Franconeri, John Kounios, and Mark Beeman. 2015. Sudden insight is associated with shutting out visual inputs. *Psychonomic bulletin & review* 22, 6 (2015), 1814–1819.

[33] Jenessa R Shapiro and Steven L Neuberg. 2007. From stereotype threat to stereotype threats: Implications of a multi-threat framework for causes, moderators, mediators, consequences, and interventions. *Personality and Social Psychology Review* 11, 2 (2007), 107–130.

[34] Dean Keith Simonton. 2000. Creativity: Cognitive, personal, developmental, and social aspects. *American psychologist* 55, 1 (2000), 151.

[35] Steven J Spencer, Claude M Steele, and Diane M Quinn. 1999. Stereotype threat and women's math performance. *Journal of experimental social psychology* 35, 1 (1999), 4–28.

[36] E Paul Torrance. 1966. Torrance tests of creative thinking. *Educational and psychological measurement* (1966).