Creative Minds at the Keyboard: Eye-Tracking Insights into How Developers Think and Code

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

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

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

Abstract

Software development is a multifaceted process that blends problem-solving with coding to create novel solutions. While creativity is central to programming practice, little is known about how it influences programmers' cognitive and behavioral strategies.

To explore this understudied aspect of programming, we conducted a controlled eye-tracking study with 40 participants to examine how creativity shapes programmers' cognitive strategies. We assessed developers' creativity using established psychometric tests and related these measures to their performance, gaze behavior, and interaction with the programming environment. Our results show that creativity meaningfully influences how developers approach and execute programming tasks. Participants with higher divergent creativity scores were more exploratory, displaying longer visual engagement and more active IDE interactions. Conversely, participants with higher convergent creativity exhibited more analytical visual scanning and reported greater workload, suggesting a more deliberate and cognitively demanding strategy.

Our findings highlight creativity as a meaningful factor in programming behavior and suggest new ways to support diverse thinking styles in software engineering (SE) practice.

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:
Mahta Amini, Yahya Lafhal, and Zohreh Sharafi. 2026. Creative Minds at the Keyboard: Eye-Tracking Insights into How Developers Think and Code. In *19th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE'26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3794860.3794891

1 Introduction

"Creativity is not a mysterious spark but a process, grounded in expertise and collaboration." — Dr. R. Keith Sawyer

Software engineering, and programming in particular, exemplifies this notion: it is a human-centered practice that blends technical expertise with creative problem-solving to produce novel solutions. Developers constantly navigate uncertainty, make design trade-offs, and adapt to evolving constraints—all of which require creativity at multiple levels, from everyday problem-solving to innovative system design. Yet, despite creativity being central to programming, research in software engineering has rarely examined it systematically. Existing work has primarily focused on isolated phases such as requirements engineering or design ideation [2, 19], often overlooking how creativity manifests during the act of coding itself. Consequently, little is known about how developers' cognitive traits, strategies, and environmental interactions contribute to creative behavior in programming practice.

Creativity encompasses at least two primary psychological facets — divergent and convergent thinking — which jointly influence creative outcomes [33, 39]. Divergent creativity involves exploring multiple possible solutions through idea generation, whereas convergent creativity focuses on identifying the single best answer.

In this paper, we aim to shed light on creativity in programming by examining how divergent and convergent thinking shape developers' coding behavior. We position creativity as a relevant yet understudied facet of software engineering practice, extending its investigation beyond ideation or design into the act of coding itself. Specifically, we explore how individual creativity profiles manifest in developers' visual search strategies and interaction with the programming environment, revealing how different forms of creative thinking influence attention, exploration, and problem-solving during software development.

To situate our work within creativity research, we adopt the 4P framework (Person, Product, Process, Press) [43], which conceptualizes creativity as a multidimensional phenomenon shaped by the individual, their output, their approach, and their environment. Here, the *Person* refers to the developers themselves—their individual characteristics and divergent/convergent creativity profiles; the *Product* represents their programming outcomes (e.g., accuracy, efficiency, and task completion); the *Process* captures their problem-solving behavior, including navigation patterns, visual search strategies, and workload perception (NASA-TLX [16]); and the *Press* reflects the task environment and tools that influence behavior exhibited. We conducted an eye-tracking experiment involving 40 participants who worked on two Python programming tasks. To quantify creativity, in addition to Torrance Tests of Creative Thinking (TTCT) [50], we employed the Divergent Association Task (DAT) [39] and the Remote Associates Test (RAT) [33], capturing both divergent and convergent facets of creativity. Eye-tracking enabled us to investigate how these creativity dimensions manifest

CHASE'26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2484-8/26/04
https://doi.org/10.1145/3794860.3794891

in developers' *visual search behavior* and *interaction with the programming environment*, offering a fine-grained view of attention, exploration, and focus during coding.

The results show that while task performance did not differ significantly across creativity profiles, their behavioral patterns diverged markedly. Participants with higher divergent creativity (*DAT*) demonstrated more exploratory and sustained engagement with the code, as evidenced by longer visit durations and more active interaction(more mouse clicks and IDE actions) with the integrated development environment (IDE). In contrast, those with higher convergent creativity (*RAT*) exhibited more analytic and effortful behaviors, characterized by increased visual scanning (more fixations and saccades), more frequent environment revisits, and higher reported workload (*NASA–TLX*). Together, these results suggest that divergent and convergent creativity are reflected not in performance outcomes but in how developers visually and behaviorally navigate coding tasks.

Together, these findings highlight important links between creativity, performance, and strategies in software development, with implications for how creativity traits shape programming behavior. In this paper, we make the following contributions:

- **First eye-tracking study of creativity in programming:** We report results from a controlled human study with 40 participants performing two Python coding tasks, combining eye-tracking, IDE interaction logs, and post-task questionnaires. We examine how divergent (DAT) and convergent (RAT) creativity scores influence performance outcomes and development strategies.
- **Insights into visual, behavioral, and cognitive strategies:** We provide empirical evidence that higher creativity scores are associated with distinct visual search patterns, IDE interaction behaviors, and perceived workload (NASA-TLX). Our analysis shows how divergent and convergent thinkers differ in their engagement with code, navigation, and task demands.
- **Data and code availability:** We provide our data and analysis scripts in a public repository to support transparency, replication, and future research.

2 Related Work

As a fundamental aspect of human progress and innovation, creativity has been studied in various disciplines. In this section, we cover the most impactful prior work on creativity, including studies that are directly relevant to our research as well as areas that have recently become prominent topics of investigation. To the best of our knowledge, prior work has not jointly leveraged standardized tests for measuring creativity with eye-tracking and fine-grained IDE interaction logs during programming tasks. Accordingly, this paper presents the first integrated investigation in SE.

Creativity in Psychology : In the field of cognitive psychology, multiple prominent models have been proposed that establish creativity as a multifaceted concept. Amongst the most prominent ones are Mooney's 4P framework (Person, Product, Process, Press) [36]. In this paper, we base our approach on the 4P framework. Amabile's Componential Theory of Creativity [1] is an impactful model, which identifies domain-relevant skills, creativity-relevant processes, and task motivation as the three core components that combine to produce creative work. Domain-relevant skills include knowledge, expertise, and technical abilities in the domain where the individual is working. Creativity-relevant processes are cognitive and personality-related abilities that can influence how a person approaches a problem. Intrinsic task motivation is the motivation to engage in a task for personal interest or enjoyment without any external influence. The Guilford model of creativity [14] centers on the idea that creativity is a cognitive ability characterized by divergent thinking, which involves generating multiple unique ideas and solutions. Boden defines creativity as "the ability to come up with ideas or artifacts that are new, surprising, and valuable." These models have been extensively studied in the context of varying fields [3].

Creativity in Software Engineering: Creativity is mostly an understudied topic in the field of software engineering. Most of this research consists of conducting studies with focus groups to gather qualitative data on the impact and role of creativity in SE [13, 35].

Other studies have investigated factors that could impact creativity in software development; for instance, Amin *et al.* examined the influence of programmers' personality traits on their creative performance [4]. Within SE, one discipline where it has received notable attention is requirement engineering(RE) [44]. Creative techniques are an essential tool for RE, where the client's requirements are vague or imprecise. Most of the work on creativity in RE falls into one of these categories: developing a framework to examine creativity in RE [7, 29, 37], providing tools and techniques, such as brainstorming, brainwriting, or brainsketching for creative requirement generation [11, 23, 31], and conducting empirical studies to research how these techniques work in real-world RE contexts, such as workshops and agile teams [8, 34, 48].

Beyond RE, two recent empirical studies [5, 38] involving 77 participants investigated how developers' creativity—measured through psychometric tests—relates to their programming behavior. Participants with higher creativity scores produced more diverse and expressive solutions, often incorporating visual-auditory features and additional functionalities. These findings demonstrate that creativity shapes developers' problem-solving strategies and expressive choices during programming, underscoring the need for a systematic study of creativity in software development practice.

Creativity and Eye tracking: Researchers have begun leveraging eye-tracking to shed light on the connection between visual attention and creative processes. Jankowska *et al.* combined the use of eye-tracking glasses with traditional creativity tests(The Test of Creative Thinking-Drawing Production [21, 51]). They concluded that visual gaze patterns can predict creative performance. The more time the participants spent looking at the main areas of interest (AOI), and the more fixations within the AOIs that were recorded, the higher their scores were. In another research, Maheshwari *et al.* examined the role of divergent and convergent thinking in eye movement patterns [30]. Their analysis revealed that during divergent-thinking tasks (an Alternate Uses Test [15]), participants exhibited more dispersed and detailed visual scanning than during convergent tasks (RAT).

Empirical evidence suggests that creative insight is often accompanied by noticeable shifts in visual attention. Findings by Kwon *et al.* [26] and Salvi *et al.* [45] indicate that, just before generating new ideas, individuals tend to reduce their visual focus by scanning

Table 1: Overview of participant demographics, including age, gender, and study level.

Characteristics	All (40)	Men (34)	Women (6)
Age (n, %)			
18–25	30 (75.0%)	28	2
25–30	6 (15.0%)	3	3
30–35	3 (7.5%)	2	1
35 or older	1 (2.5%)	1	0
Educational (n, %)			
Undergraduate	13 (32.5%)	13	0
Bachelor's Degree	13 (32.5%)	13	0
Graduate Degree	14 (35.0%)	8	6

more widely or briefly looking away, which appears to help the mind reorganize information and foster creative thinking.

Creativity and Artificial Intelligence Advancement: Recent advancements in Artificial Intelligence (AI), particularly in Generative AI (GenAI) powered by large language models (LLMs), have transformed how humans create, design, and interact with technology. With the emergence of accessible tools capable of generating text, images, code, and even ideas, GenAI has made creative and technical tasks more collaborative and scalable than ever before. These rapid developments have sparked growing research interest in understanding AI's influence on creativity and innovation [9, 17].

Using McLuhan's tetrad framework [32], Jackson *et al.* examined how GenAI reshapes software development and identified five key research themes: individuals, teams, product, unintended consequences, and society [20]. A related body of work focuses on human–AI co-creation, investigating how people collaborate with generative systems in creative domains such as musical composition [28], scientific ideation [41], and writing [24, 27].

3 Experimental Methodology

We conducted an in-person study to explore how creativity shapes developers' performance and problem-solving strategies during software programming tasks. The study was conducted under the approval of our institutional review board, and informed consent was obtained from all participants prior to their participation.

3.1 Participants and Recruitment

A total of 40 participants were recruited for this study, comprising both graduate and undergraduate students. Recruitment was conducted through university email lists, LinkedIn, and institutional communication channels such as Discord. Calendly was used to manage participant scheduling, messaging, and prescreening. Prior to participation, individuals answered two screening questions regarding their familiarity with the Python programming language and their programming experience (number of years). Only those with relevant Python experience were included in the study. Participant demographics are summarized in Table 1. Participants were compensated with $40 at the end of the experiment.

3.2 Software Systems and Tasks

The study included two primary programming tasks, both conducted in a controlled laboratory environment using the PyCharm IDE. Participants were given a maximum of 20 minutes to complete each programming task.

In the first task, participants were presented with the classic *Pong* game. Pong is a simple two-player arcade game where each player controls a paddle to keep a ball in play and prevent it from passing their side of the screen. Participants were provided with a partially implemented version of the game in which the scoring mechanism had been intentionally removed. Their task was to implement this missing functionality without external guidance or examples.

In the second task, participants worked with the *scikit-learn* [40] framework. We introduced 28 code smells [10] distributed across 15 files within the framework's `Examples` directory. Code smells are subtle indicators of potential design flaws that, while not breaking functionality, often make code harder to understand, maintain, or extend. These covered a range of code smells that varied in difficulty, from relatively easy to detect issues to those requiring deeper structural understanding. Examples included *Long Method*, *Duplicate Code*, *Data Clumps*, *Feature Envy*, and *Magic Numbers*. Participants were instructed to identify and refactor these code smells and anti-patterns to improve code quality.

3.3 Equipment and Setup

Our experiment was performed on a 27-inch monitor with a resolution of 1920 x 1080 pixels. We recorded the participant's eye movements using the Tobii Pro Fusion eye-tracker [18]. Tobii Pro Fusion is a screen-based, non-intrusive, and remote device that can capture gaze data at speeds up to 250 Hz. We used the Tobii Eye Tracker manager to adjust the participant's seating position using its visual indicator. Before each task, a standard calibration process was performed using Tobii Pro Lab. Tobii Pro Lab also handled screen recording throughout the experiment. During the programming tasks, the screen was divided into three distinct sections (Figure 2). The left side of the screen was dedicated to the PyCharm IDE, the upper right side to a Chrome web page and the lower right side displayed a timer. Given that our eye-tracking analysis relies on fixed screen coordinates, altering the screen layout could disrupt the mapping between gaze data and visual elements. Therefore, participants were instructed not to modify the layout or resize the windows.

To capture interactions within the IDE, such as code edits and navigation events, we used CodeGRITS [49] alongside Tobii Pro Lab. CodeGRITS is a JetBrains IDE plugin compatible with PyCharm and other IDEs and used for empirical software engineering research. It enabled us to track developers' interactions within the IDE as well as their eye gaze data.

3.4 Procedure

The experiment was conducted in a quiet room with minimal distractions. Participants sat in a comfortable chair with armrests, signed a consent form, and received a detailed explanation of the procedure. To ensure privacy and confidentiality, each participant

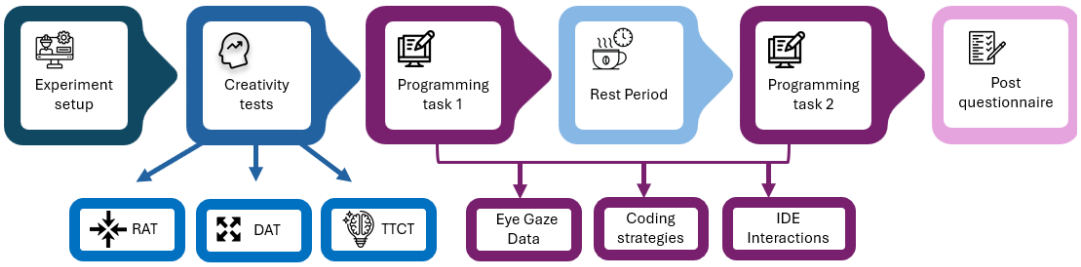


Figure 1: Overview of the experiment's procedure.

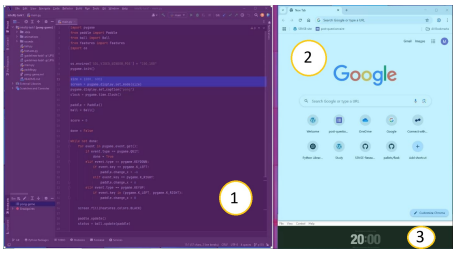


Figure 2: Experimental workspace layout showing its three main sections: the IDE (project explorer and code editor), the web browser, and the timer.

was assigned a randomized, anonymized ID used throughout the study. All collected data have been made accessible here [1].

The experiment consisted of three main sections. In the first section, participants completed three tests designed to measure different aspects of creativity. We developed a study-specific website with distinct pages for this purpose. Participants used our website to perform the DAT test and the RAT test, which assess divergent and convergent thinking, respectively. Participants also completed a standardized figural version of the TTCT test. In this task, participants were given a sheet of paper with 30 empty circles and instructed to fill in as many as they could with unique drawings within a time limit of two minutes. This test assesses dimensions of creativity that are not limited to linguistic or semantic processing, like DAT and RAT.

In the second session, participants completed the two programming tasks described in Section 3.2. To counterbalance potential order effects, the tasks were alternated between participants—half began with Task 1 and the other half with Task 2. Each participant worked on a dedicated source code branch identified by their anonymized ID. Eye-tracker calibration was performed at the start of each task and repeated as needed if accuracy or precision fell below acceptable thresholds. Before beginning the main task, participants reviewed the provided guidelines and procedural steps. Participants were instructed to refrain from using external AI-based assistance while solving the tasks to ensure performance reflected their own problem-solving processes. The researcher then launched CodeGRITS and initiated the timer. A five-minute rest period was provided between tasks to minimize visual fatigue and maintain data quality.

In the final section, the participants filled out a post-questionnaire form. In this questionnaire, they reported their cognitive workload using the NASA-TLX questionnaire and provided feedback about the experiment. Figure 1 provides an overview of the experimental procedure.

3.5 Data Processing and Preparation

We used Tobii Pro Lab software to analyze participants' eye movements. We segmented eye movements into fixations and saccades [42], and then computed standard fixation- and saccade-derived measures (*e.g.*, number of fixations, total and average fixation duration; number of saccades), which are commonly used as proxies for cognitive load and visual effort. *Fixations* are spatially stable eye gaze lasting for 200 to 300ms. During a fixation, the participant's visual attention is concentrated on a specific area of the stimulus, triggering cognitive processes [22]. *Saccades* are common, continuous, and rapid eye movements lasting 40–50ms. They occur between fixations, providing limited visual perception. We applied Tobii Pro Lab's I-VT (Velocity-Threshold Identification) filter to generate fixations from raw data. Participants' eye movements were analyzed by mapping fixation data to predefined areas of interest(AOIs). Following Goldberg and Helfman's AOI design guidelines [12], we divided the screen into three sections: the IDE, the web browser, and the timer, as shown in Figure 2.

We examined behavioral logs extracted from CodeGRITS that capture how participants interacted with the IDE while solving programming tasks. These logs reflect a range of activities, from fine-grained editing behavior to broader navigation and interaction patterns. To organize these data, we grouped all metrics into three main behavioral dimensions—*IDE Attention and Information Processing*, *IDE Navigation and Exploration*, and *Code Editing and Execution*—as summarized in Table 2.

To ensure comparability across participants and tasks, all metrics were normalized using z-scores and aggregated to compute an overall IDE interaction score for each participant per task. This process provided a detailed behavioral profile of how individuals edited, navigated, and managed their development environment.

4 Experiment Measures

In our experiment, we focus on two main measures: developers' performance and their problem-solving strategies. We then investigate how individual creativity levels influence these measures.

[1] https://doi.org/10.6084/m9.figshare.31198108

Table 2: Participants' behavioral metrics grouped by IDE activity dimension.

Category	CodeGRITS Metrics Included
IDE Attention and Information Processing	`n_fixations`, `revisits_total`, `reading_depth`, `entropy_files`
IDE Navigation & Exploration	`file_switch_count`, `line_switch_count`, `n_lines_viewed`, `n_files_viewed`, `entropy_files`
Code Editing and Execution	`n_typings`, `n_caret_pos_changes`, `n_actions`, `n_navigate_actions`, `n_undo_redo`, `n_runs`, `n_saves`, `n_debug_actions`, `n_files_opened_or_switched`, `n_mouse_moves`

Specifically, we examine whether participants with higher creativity scores demonstrate differences in performance outcomes or adopt distinct strategies compared to those with lower scores.

4.1 Psychometric Creativity Scores

Creativity is a multidimensional construct encompassing both divergent and convergent thinking, which we assessed alongside a broader measure of creative potential.

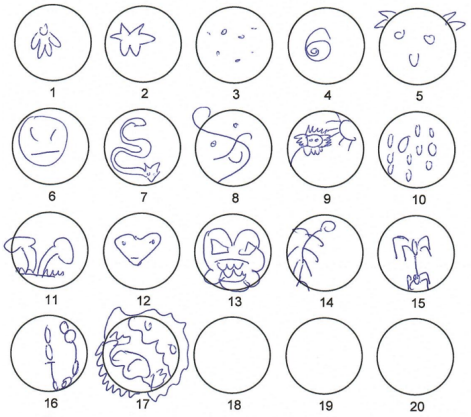
Divergent Association Task (DAT) [39]: Participants listed 10 words that were maximally distinct in meaning and usage. The semantic distance between these words provides an objective indicator of divergent thinking and has been shown to align with outcomes on other creativity assessments. DAT produces scores ranging from 1 to 100, with higher values indicating greater divergent creativity.

Remote Associates Test (RAT) [33]: Participants solved 10 triads of words by generating a fourth word that linked all three through a compound or associative phrase. For example, given Square, Cardboard, and Open, the correct response is Box. The RAT assesses convergent thinking under a strict time limit (five minutes total, or 30 seconds per triad). RAT yields scores between 0 and 10, reflecting convergent creative ability.

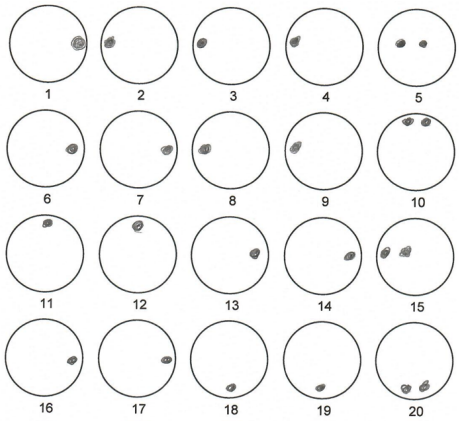
Torrance Tests of Creative Thinking (TTCT) [50]: Participants also completed selected items from the TTCT, a widely used standardized measure that evaluates creativity across multiple dimensions such as fluency, flexibility, originality, and elaboration. Unlike the DAT and RAT, which isolate divergent and convergent thought processes respectively, the TTCT offers a broader assessment of creative potential. While rooted in divergent thinking, it also captures how individuals generate, elaborate on, and adapt novel ideas across different contexts.

4.2 Performance Metrics

Participant performance was assessed using two metrics: (1) task completion time, which reflects efficiency by measuring how long participants took to finish the tasks, and (2) success rate, which reflects accuracy by verifying the correctness of responses, proper implementation of the scoring functionality, and error-free code compilation and execution.



(a) High TTCT score: fluency = 17, flexibility = 5, originality = 7, elaboration = 19.5, Total (std.) = 1.193



(b) Low TTCT score: fluency = 16, flexibility = 1, originality = 0, elaboration = 0; total (std.) = −1.263

Figure 3: Examples of responses from participants with high and low TTCT scores.

4.3 Problem-Solving Strategies

To assess participants' problem-solving strategies, we focused on the following three categories:

Visual Search we analyzed eye-movement metrics that show how participants allocate attention, process information, and navigate visually while solving programming tasks, including: `Avg Fixation Duration`, `Fixations No.`, and `Saccades No.`.

Programming Environment Interaction it consists of the two components: 1) Stimuli navigation: we calculated and utilized `Visits No.` and `Mouse Clicks No.`, which indicate how often participants revisited areas of interest (AOIs), namely the IDE and the web browser, and how frequently they interacted with these interfaces

Table 3: Representative RAT responses across high and low scoring participants

#	RAT Cues	Correct Answer	High Scorer	Low Scorer
1	Fountain, Baking, Pop	Soda	soda	Chocolate
2	Political, Surprise, Line	Party	news	Vote
3	Date, Alley, Fold	Blind	blind	Time
4	Safety, Cushion, Point	Pin	pin	Car
5	Basket, Eight, Snow	Ball	ball	Shovel
6	Coin, Quick, Spoon	Silver	silver	Steel
7	Square, Cardboard, Open	Box	box	Knife
8	Pure, Blue, Fall	Water	water	River
9	Notch, Flight, Spin	Top	top	Jet
10	Measure, Desk, Scotch	Tape	tape	Wood
		Score:	**9/10**	**0/10**

Table 4: Representative DAT responses across high and low scoring participants

Item	High Scorer	Low Scorer
word_1	Ocean	sky
word_2	Graphic	cloud
word_3	Human	heart
word_4	Flag	tree
word_5	Necklace	land
word_6	Assistant	sea
word_7	Opportunity	ocean
word_8	Anxiety	rock
word_9	Mirror	sand
word_10	Cable	wind
Score:	**87.006**	**60.387**

through mouse actions. 2) IDE interaction: The IDE interaction measure reflects the extent and diversity of participants' engagement with the development environment during programming. It captures their overall behavioral activity, encompassing editing, navigation, and task management within the IDE.

Workload Perception (NASA-TLX) Finally, we measured participants' subjective workload using the NASA-TLX questionnaire. This instrument captures six dimensions—mental demand, physical demand, temporal demand, performance, effort, and frustration—to provide an overall workload score. Incorporating TLX with visual search and environment interaction connects objective behaviors to subjective workload, offering a more integrated view of developers' problem-solving.

5 Analysis and Results

Our analysis addressed the following research questions, progressing from examining whether creativity matters in programming performance to understanding how it manifests in developers' behaviors and strategies:

RQ1. To what extent is programmers' task performance associated with their creativity scores?

RQ2. How do participants' levels of divergent and convergent thinking, as measured by DAT and RAT scores, influence their problem-solving strategies during development tasks?

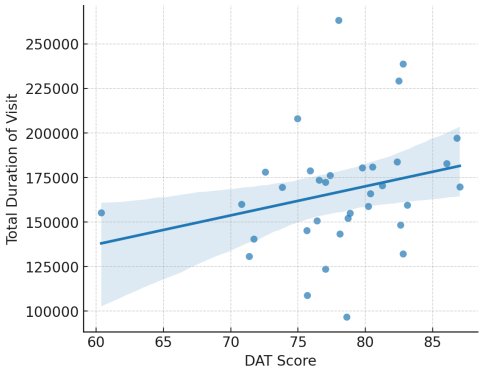


Figure 4: Scatter plot with regression line illustrating the association between DAT scores and the visit duration on the left and the number of mouse clicks on the right. Shaded region represents the 95% confidence interval. The moderate upward trend aligns with the statistically significant correlation for visit duration ($\rho = 0.346, p = 0.038$)

.

For group-based comparisons, we computed the sample mean for each creativity measure and categorized participants as either *high* (above the mean) or *low* (below the mean) in their respective creativity dimension, resulting in high-DAT/low-DAT and high-RAT/low-RAT groups. This approach enabled us to examine both the continuous relationships between creativity and behavioral measures, and categorical differences between participants with relatively higher and lower levels of divergent or convergent thinking. Figure 3 and tables 3 and 4 show creativity test outcomes for participants with low and high scores.

5.1 RQ1. Creativity and Performance

We compared participants' task performance across accuracy, completion success, and time.

For error-free compilation rates, both DAT and RAT high scorers showed slightly higher success compared to their low-scoring counterparts, with a somewhat stronger effect for RAT (High = 86% vs. Low = 73%) than for DAT (81% vs. 74%). In terms of score accuracy, high-DAT participants achieved notably higher accuracy rates (67% vs. 42%), whereas RAT groups showed minimal difference (57% vs. 54%). However, Fisher's Exact Test indicated that these contrasts were not statistically significant.

Task completion times revealed similar patterns. For Task 1, high-DAT scorers ($M = 401.1$, $SD = 351.8$) spent less time than low-DAT scorers ($M = 479.0$, $SD = 408.8$), while high-RAT scorers ($M = 588.1$, $SD = 453.4$) spent more time than low-RAT scorers ($M = 371.7$, $SD = 325.4$). For Task 2, high-DAT participants again completed the task faster ($M = 226.2$, $SD = 124.7$) than low-DAT participants ($M = 274.5$, $SD = 328.3$), whereas high-RAT scorers took longer ($M = 296.3$, $SD = 287.4$) than low-RAT scorers ($M = 223.8$, $SD = 214.8$). Mann–Whitney U tests indicated non-significant trends for both comparisons.

To complement the DAT and RAT analyses, we also examined performance differences based on overall creative potential (TTCT).

Table 5: Mean (std) of visual search metrics for DAT, RAT, and TTCT groups for Task 1 and Task 2. Statistically significant differences are highlighted in the table. High RAT scorers exhibited a greater number of fixations and saccades across the entire stimulus (IDE and browser), reflecting increased visual scanning and cognitive effort.

		Task 1			Task 2		
		Avg Fix. Duration	Fixations No.	Saccades No.	Avg Fix. Duration	Fixations No.	Saccades No.
DAT	High	298 (63)	748 (626)	575 (520)	293 (111)	254 (471)	197 (341)
	Low	334 (68)	882 (756)	720 (640)	319 (155)	187 (374)	149 (320)
RAT	High	300 (75)	1067 (783)	802 (643)	300 (117)	258 (472)	202 (343)
	Low	324 (63)	689 (615)	570 (543)	315 (153)	184 (373)	146 (319)
TTCT	High	323 (58)	847 (790)	691 (652)	326.5 (66)	635 (723)	214 (263)
	Low	303 (72)	833 (632)	645 (548)	301 (65)	543 (364)	226 (299)

High-TTCT participants showed slightly higher task accuracy (68.4% vs. 59.1%) than low scorers, though these differences were not statistically significant.

Participants with high TTCT scores ($M = 287.83$, $SD = 260.16$) spent more time fixating on the code than those with *low TTCT scores* ($M = 223.51$, $SD = 227.30$), though this difference was not statistically significant ($U = 207.0$, $p = .44$).

RQ1. Creativity and Performance: High DAT, RAT, and TTCT scorers showed slightly better accuracy and completion rates and tended to spend more time on tasks or code, suggesting deeper engagement, though none of these differences were statistically significant.

5.2 RQ2. Creativity and Problem-solving Strategy

To assess participants' problem-solving strategies, we considered three complementary aspects. The first was visual search performance, which captures how participants locate and process information while engaging with the task. The second was development environment interaction, which reflects how participants translated ideas into code, implemented potential solutions, and adapted their approaches in response to challenges. The third dimension was workload perception, measured using the NASA-TLX questionnaire, which provides insights into participants' subjective experience of task difficulty. Taken together, these measures provide a comprehensive understanding of the strategies participants employed when solving problems.

5.2.1 Creativity and Visual Search. We examined whether participants with elevated creativity scores surpassed those with lower scores in visual search performance.

Table 5 presents various visual performance metrics across the RAT, DAT, and TTCT score groups for task 1 and task 2. We used the Mann–Whitney U test to assess differences between high and low creativity groups and complemented this with Spearman's rank correlation to examine relationships between continuous scores and visual metrics.

For task 1, participants with higher RAT scores exhibited significantly greater numbers of fixations ($Ustatistic : 229.5, p - value : 0.03$) and saccades ($Ustatistic : 224.0, p - value : 0.04$). For number of fixations, Spearman's analysis also showed a moderate positive correlation ($\rho = .346$, $p = .038$), significant at the .05 level.

While examining DAT scores, Spearman's rank correlation analysis revealed a moderate positive relationship between DAT score and total visit duration on the code ($\rho = .340$, $p = .030$), statistically significant at the .05 level (Figure 4).

Also, the high TTCT group exhibited slightly longer fixation durations and a higher number of saccades, although these differences were not statistically significant.

Together, these findings suggest that while convergent thinkers (*high RAT*) demonstrated more frequent but shorter eye movements—reflecting analytical scanning—divergent thinkers (*high DAT*) exhibited longer and more sustained fixations, indicative of deeper and more exploratory processing of the code.

Table 6: Mean (std) of programming environment interaction metrics for DAT, RAT, and TTCT groups for Task 1. Statistically significant differences are highlighted.

	Stimuli Navigation		IDE Interaction (z-score)		
	Visits	M Clicks	Attention	Navigation	Editing
DAT H	13 (14)	226 (132)	-0.04 (0.7)	-0.07 (0.7)	0.22 (0.6)
DAT L	14 (15)	145 (83)	0.04 (0.8)	0.07 (0.9)	-0.22 (0.5)
RAT H	18 (15)	208 (128)	0.14 (0.8)	0.11 (0.8)	-0.007 (0.5)
RAT L	14 (15)	145 (83)	-0.07 (0.7)	-0.05 (0.8)	-0.22 (0.5)
TTCT H	12 (14)	202 (81)	0.01 (0.9)	-0.005 (0.9)	0.21 (0.6)
TTCT L	16 (16)	186 (137)	0.01 (0.7)	0.004 (0.7)	-0.19 (0.5)

5.2.2 Creativity and Programming Environment interaction. Our analysis examined potential differences in programming environment interaction between participants with higher and lower creativity scores. As shown in Tables 6 and 7, interaction metrics were computed for task1 and task2 across DAT and RAT groups, and Mann–Whitney U tests were applied to evaluate group differences.

For stimulus navigation, in task 1, participants with higher RAT scores made significantly more visits ($Ustatistic : 286.0, p - value : 0.01$). Also, participants with higher DAT scores performed significantly more mouse clicks ($Ustatistic : 297, p - value : 0.045$) while interacting with the stimulus (IDE and the browser).

The analysis of IDE interaction revealed that participants with elevated DAT scores performed a larger number of code editing

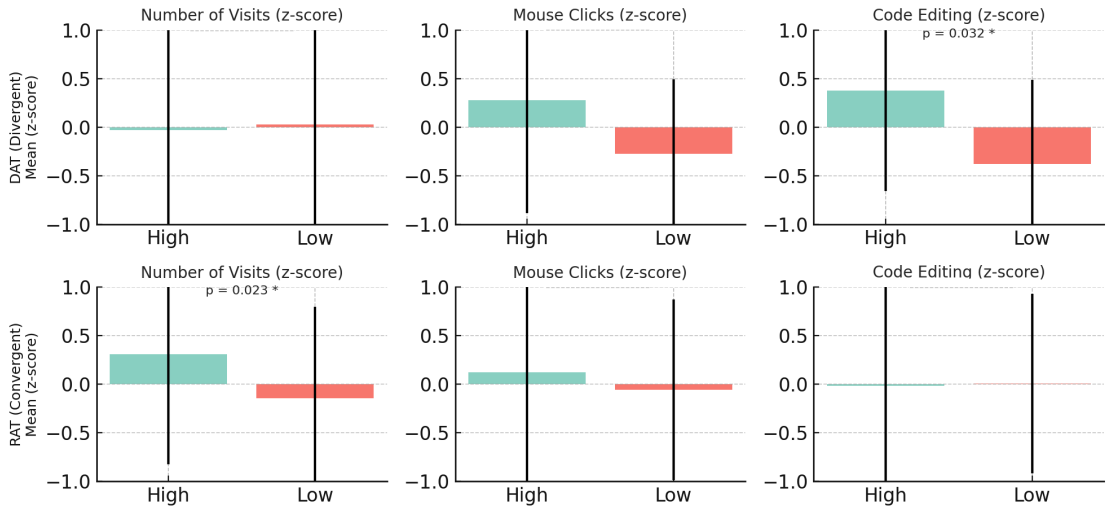


Figure 5: Mean standardized scores (z-scores) for Number of Visits, Mouse Clicks, and Code Editing by creativity profile. Bars represent mean values for High (teal) and Low (coral) groups within each creativity dimension — DAT (Divergent) and RAT (Convergent). Asterisks (*) mark statistically significant group differences (p < .05). High DAT participants showed greater engagement in code editing, whereas High RAT participants made more visits to the development environment.

Table 7: Mean (std) of programming environment interaction metrics for DAT, RAT, and TTCT groups for Task 2. No statistically significant differences were observed, except that high TTCT scorers performed more code editing actions.

	Stimuli Navigation		IDE Interaction (z-score)		
	Visits	Clicks	Attention	Navigation	Editing
DAT H	6 (6)	231 (144)	-0.10 (0.58)	-0.17 (0.5)	0.04 (0.5)
DAT L	5 (4)	212 (107)	0.11 (0.92)	0.19 (1.1)	-0.05 (0.4)
RAT H	6 (6)	242 (116)	0.10 (0.77)	0.20 (1.0)	-0.04 (0.3)
RAT L	5 (4)	212 (133)	-0.05 (0.77)	-0.11 (0.84)	0.022 (0.5)
TTCT H	8 (5)	251 (118)	0.07 (0.9)	0.12 (1.2)	0.15 (0.5)
TTCT L	8 (7)	196 (130)	0.06 (0.5)	-0.10 (0.5)	-0.12 (0.4)

actions and were more actively engaged with the code ($Ustatistic$: 254.5, $p - value$: 0.032).

Higher DAT scores were associated with more active engagement in the programming environment. Participants with stronger divergent thinking tended to perform a greater number of code editing actions—such as typing, executing runs, and debugging—which reflects increased exploration and interaction within the code itself.

High TTCT scorers made slightly fewer visits but more mouse clicks, although these differences were not statistically significant.

5.2.3 Creativity and Workload Perception. We also examined whether creativity levels influenced participants' perceived workload, as measured by the NASA-TLX questionnaire. While no differences emerged between high and low DAT groups (High: $M = 50.03$, $SD = 7.57$; Low: $M = 47.31$, $SD = 10.41$; $U = 222.5$, $p = .54$), high RAT scorers reported significantly higher workload compared to low RAT scorers (High: $M = 52.89$, $SD = 6.94$; Low: $M = 46.50$, $SD = 9.32$; $U = 256.5$, $p = .036$).

This suggests that convergent thinkers experienced the tasks as more demanding, possibly reflecting their stronger focus on correctness and higher self-imposed performance standards as illustrated in Figure 6.

RQ2. Creativity and Problem-solving Strategy Creativity shaped how participants approached programming tasks. Convergent thinkers (high RAT) showed more fixations, saccades, and visits, reflecting analytical scanning, whereas divergent thinkers (high DAT) had longer visits, more clicks, and greater code-editing activity, suggesting exploratory engagement. High TTCT scorers showed similar but non-significant trends. Convergent thinkers also reported higher workload, indicating a more intense, goal-oriented approach.

6 Discussion

Our findings reveal that creativity manifests not only in programming outcomes alone, but also in the distinct ways developers approach problem-solving, allocate attention, and interact with their environment. Although performance outcomes such as accuracy and task completion time did not differ significantly across creativity levels, eye-tracking and behavioral data provided a richer perspective on how creativity unfolds during coding.

Eye-movement and IDE interaction data revealed that divergent and convergent thinkers engage differently with source code. Divergent thinkers showed more active interaction with the environment, including higher numbers of mouse clicks, IDE actions, and navigations—behaviors consistent with exploratory programming and iterative refinement. In contrast, higher RAT scores were associated

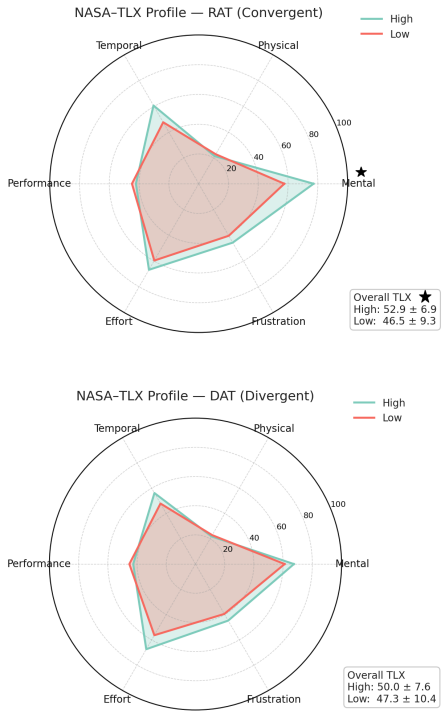


Figure 6: NASA–TLX workload profiles for High (teal) and Low (coral) scorers in (a) DAT (Divergent) and (b) RAT (Convergent) groups. Radar plots show mean self-reported workload across six dimensions: Mental, Physical, Temporal, Performance, Effort, and Frustration. Overall workload (mean ± SD) appears beside each plot. Divergent thinkers showed more balanced workloads, while convergent thinkers reported higher effort and mental demand.

with more frequent fixations and saccades, indicating analytical scanning and greater cognitive load. This pattern was further supported by the self-reported NASA–TLX results, where convergent thinkers reported higher perceived workload and effort.

These patterns align with cognitive theories of creativity, where divergent thinking supports exploration and flexibility, while convergent thinking emphasizes focused evaluation. Our findings also echo recent research on creativity and programming behavior. Amini et al.[5] found that participants with higher creativity scores demonstrated richer problem-solving strategies and incorporated additional features beyond task requirements. Similarly, Njoku et al.[38] observed that creative developers adopted more innovative, synergistic coding behaviors, such as using multimedia elements and unconventional implementation patterns. Together with our findings, these studies reinforce that individual creativity profiles influence not only the outcomes of programming tasks but also the processes through which developers engage, explore, and innovate within their coding environments.

6.1 Implications for Research and Practice

Our findings highlight the importance of considering creativity as a central dimension of software development. Creativity-aware design of programming tools and learning environments could better support developers with diverse cognitive and creative profiles. Divergent thinkers may thrive in environments that foster rapid experimentation, feedback, and exploration, while convergent thinkers may benefit from interfaces that minimize cognitive load and promote structured reasoning.

From a pedagogical perspective, our results point to the need for creativity-informed teaching strategies in software engineering education. Instructors could design learning activities that explicitly alternate between divergent and convergent phases—encouraging idea generation, reflection, and structured refinement. For example, open-ended coding challenges or collaborative brainstorming can cultivate divergent thinking, while code reviews, debugging exercises, and algorithmic analysis foster convergent reasoning. Embedding this dual approach in curricula would help students not only produce correct solutions but also explore multiple pathways toward them, strengthening creative flexibility.

Methodologically for research, this study demonstrates the value of integrating psychometric creativity assessments with behavioral and eye-tracking analytics to capture the multifaceted nature of creativity in programming. This multi-method design bridges cognitive psychology and software engineering and paves the way for establishing a framework that moves beyond performance-based evaluation toward understanding how developers think, explore, and adapt while coding.

Such integration enables a richer characterization of programming as both a technical and creative act—where problem-solving is shaped not only by expertise but also by the balance between divergent and convergent cognitive processes. Eye-tracking and interaction analytics reveal aspects of creative behavior that remain invisible in outcome-based studies, such as shifts in focus, depth of engagement, and visual persistence during exploration. These insights highlight the potential of cognitive and behavioral triangulation for uncovering individual strategies and cognitive states as they unfold in real time. More broadly, this methodological perspective advances a more human-centered view of software development research. It highlights the importance of studying developers in context—capturing mental effort, attention, and exploration as core elements of creative cognition. It also encourages empirical approaches that see programming not merely as code production, but as a dynamic interplay of perception, reasoning, and imagination.

6.2 Integrating the 4P Framework

Viewed through the 4P framework [43], our results suggest that the *Person* (creative profile) influences the *Process* (visual search and interaction) more strongly than the *Product* (task outcomes), with the *Press* (coding environment) playing a mediating role. Extending this interpretation, the contrast between our two experimental tasks provides further insight into the interaction between *Person* and *Press*. Significant effects emerged primarily in Task1, which involved implementing a scoring mechanism—a more open-ended and generative activity that naturally encourages divergent

thinking and exploratory behavior. In contrast, Task2 focused on refactoring and anti-pattern identification, a more rule-constrained and expertise-dependent activity that likely limited opportunities for creative expression. It is also possible that participants' limited familiarity with design patterns and refactoring principles reduced their ability to apply creativity effectively in this context.

Complementing these findings, the 4C model of creativity [6] situates these behaviors within broader levels of creative expression. Developers' navigation and problem-solving reflect *Little-c creativity*, characteristic of everyday programming. Divergent and convergent differences may indicate pathways toward *Pro-c creativity*, where expertise and exploration converge. Prolonged gaze durations and revisits further suggest moments of *Mini-c creativity*, as developers construct insights during real-time coding.

While our study primarily focused on creativity profiles derived from DAT and RAT measures, it did not examine broader individual differences that may further shape creative behavior. Personality traits such as openness, tolerance for ambiguity, and cognitive flexibility are likely to influence how developers approach uncertainty, problem-solving, and exploration [4]. Considering these factors in future research could clarify how enduring dispositions interact with creative cognition, informing a more comprehensive understanding of the "Person" dimension in software creativity. Such an extension would help distinguish between creativity as a situational response and creativity as a stable personal tendency that guides how individuals engage with code, tools, and challenges.

7 Threads to Validity

Several factors potentially affect the validity of our study. We concealed the actual goal of the study from the participants. We asked them not to discuss the experiment with other students to minimize the risk of hypothesis guessing and demand characteristics. However, we did inform them about the study's procedure and tasks, as well as the fact that they would be connected to an eye tracker for the duration of the experiment. The Hawthorne effect is the alteration of behavior by participants due to their awareness of being watched and evaluated. To overcome this risk, we explained to the participants that the eye tracker does not record any video or image beyond the eye itself. In addition, our experimenter sat at a distance from the participants to minimize their sense of being watched. We administered a self-assessment questionnaire regarding the participants' coding skills at the end of the study to mitigate stereotype threat and avoid influencing their performance [46, 47]. To minimize the instrument bias, we employed a video–based eye–tracking system that does not require cumbersome goggles, allowing for natural head movement. Camera calibration was performed for each participant before each task set to ensure accuracy and consistency.

To strengthen conclusion validity, we used established eye tracking metrics, applied standard statistical methods, and measured creativity with validated standardized assessments. Creativity is multifaceted, and standardized tests, like the ones that we employed, only capture parts of divergent and convergent thinking. Scores may also be influenced by verbal fluency and test familiarity. Likewise, our eye-tracking indicators and IDE event logs are indirect proxies for cognitive strategy rather than direct measures. To limit these threats, we triangulated across independent sources, including creativity tests, gaze data, IDE interaction, and post-task workload, and based our conclusions on patterns that converge across all these measures, rather than relying on any single metric.

All the participants in our study were students with solid programming skills. This is appropriate when the goal of the study is not to evaluate the impact of expertise, and the study involves near-term professionals [25]. However, the findings may not fully generalize to seasoned developers or other domains, tools, or languages. We used a fixed IDE+browser setup and a small task set (two tasks), one of which involved a reasonably large, widely used open-source project in a popular language (Python). This setup improves control and realism but still constrains generalizability. Replication with varied systems, languages/IDEs, task types, and professional cohorts is needed. Finally, our sample size reflects the nature of controlled eye-tracking studies in SE/HCI and is typical for such studies. Larger and more diverse samples enable finer-grained analyses and stronger generalization.

8 Conclusion and Future Work

This study is among the first to empirically link developers' creativity profiles with their real-time visual and behavioral patterns. We examined how divergent and convergent creativity shape programmers' visual attention, problem-solving behavior, and interaction with the coding environment. Through a controlled eye-tracking experiment involving 40 participants, we integrated psychometric creativity assessments (DAT, RAT, and TTCT) with behavioral and perceptual data, revealing creativity as an active and measurable component of programming practice.

Our findings indicate that while creativity did not directly affect task performance, it significantly shaped how programmers planned, explored, and executed their coding tasks, influencing their problem-solving strategies and engagement style. Divergent thinkers demonstrated more exploratory and sustained engagement with the environment, whereas convergent thinkers exhibited more analytical and effortful visual patterns, accompanied by a higher perceived workload. These results highlight that creativity in software development manifests not in output quality but in the processes through which developers engage with code. By situating our analysis within the 4P framework, we contribute to a growing understanding of programming as both a technical and creative act.

Future work could extend this line of inquiry through longitudinal studies to examine how creative behaviors evolve with experience, training, or sustained tool use, and whether individual creativity traits remain stable or adapt as developers gain expertise. While our focus centered on the person and process dimensions, it is equally important to investigate the product—the code or software itself—and evaluate its creative qualities. Creativity at the product level could be operationalized through dimensions such as originality, clarity of design, functional coherence, and aesthetic or expressive value. This broader perspective would help bridge cognitive, behavioral, and artefactual views of creativity in software engineering.

Acknowledgments

We thank the participants for their contribution. This work was supported by the Natural Sciences and Engineering Research Council of Canada (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, Mohd Fadzil Hassan, and Mobashar Rehman. 2018. A snapshot of 26 years of research on creativity in software engineering-A systematic literature review. *Mobile and Wireless Technologies 2017: ICMWT 2017 4* (2018), 430–438.

[3] Aamir Amin, Shuib Basri, Mohd Fadzil Hassan, and Mobashar Rehman. 2018. A snapshot of 26 years of research on creativity in software engineering-A systematic literature review. *Mobile and Wireless Technologies 2017: ICMWT 2017 4* (2018), 430–438.

[4] 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.

[5] 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

[6] Ronald A Beghetto and James C Kaufman. 2007. Toward a broader conception of creativity: A case for" mini-c" creativity. *Psychology of aesthetics, creativity, and the arts* 1, 2 (2007), 73.

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

[8] Shaun Dallman, Lemai Nguyen, John Lamp, and Jacob Cybulski. 2005. Contextual factors which influence creativity in requirements engineering. (2005).

[9] 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).

[10] Martin Fowler. 2018. *Refactoring: improving the design of existing code*. Addison-Wesley Professional.

[11] 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.

[12] Joseph H. Goldberg and Jonathan I. Helfman. 2010. Comparing Information Graphics: A Critical Look at Eye Tracking. In *Proceedings of the 3rd BEyond Time and Errors: Novel evaLuation Methods for Information Visualization Workshop* (Atlanta, Georgia) *(BELIV '10)*. ACM, New York, NY, USA, 71–78. doi:10.1145/2110192.2110203

[13] 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.

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

[15] Joy Paul Guilford. 1967. The nature of human intelligence. *New York: Macgraw Hill* (1967).

[16] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*. Vol. 52. Elsevier, 139–183.

[17] 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).

[18] https://www.tobiipro.com/. 2001. Online; Accessed 17-07-2020.

[19] 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. *ACM Trans. Softw. Eng. Methodol.* 34, 5, Article 133 (May 2025), 28 pages. doi:10.1145/3708523

[20] 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).

[21] 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.

[22] Marcel A Just and Patricia A Carpenter. 1980. A theory of reading: from eye fixations to comprehension. *Psychological review* 87, 4 (1980), 329.

[23] Inger Kristine Karlsen, Neil Maiden, and Andruid Kerne. 2009. Inventing requirements with creativity support tools. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 162–174.

[24] 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.

[25] Barbara A. Kitchenham, Shari Lawrence Pfleeger, Lesley M. Pickard, Peter W. Jones, David C. Hoaglin, Khaled El Emam, and Jarrett Rosenberg. 2002. Preliminary Guidelines for Empirical Research in Software Engineering. *IEEE Transactions on Software Engineering* 28, 8 (Aug. 2002), 721–734. doi:10.1109/TSE.2002.1027796

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

[27] 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.

[28] 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.

[29] Martin Mahaux, Alistair Mavin, and Patrick Heymans. 2012. Choose your creativity: Why and how creativity in requirements engineering means different things to different people. In *International working conference on requirements engineering: Foundation for software quality*. Springer, 101–116.

[30] 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.

[31] Neil Maiden, Sharon Manning, Suzanne Robertson, and John Greenwood. 2004. Integrating creativity workshops into structured requirements processes. In *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques*. 113–122.

[32] Marshall McLuhan. 1977. Laws of the Media. *ETC: A Review of General Semantics* (1977), 173–179.

[33] Sarnoff Mednick. 1962. The associative basis of the creative process. *Psychological review* 69, 3 (1962), 220.

[34] Rahul Mohanani, Paul Ralph, Burak Turhan, and Vladimir Mandić. 2021. How templated requirements specifications inhibit creativity in software engineering. *IEEE Transactions on Software Engineering* 48, 10 (2021), 4074–4086.

[35] Rahul Mohanani, Prabhat Ram, Ahmed Lasisi, Paul Ralph, and Burak Turhan. 2017. Perceptions of creativity in software engineering research and practice. In *2017 43rd euromicro conference on software engineering and advanced applications (seaa)*. IEEE, 210–217.

[36] 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.

[37] Lemai Nguyen and Graeme Shanks. 2009. A framework for understanding creativity in requirements engineering. *Information and software technology* 51, 3 (2009), 655–662.

[38] 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

[39] 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.

[40] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[41] 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).

[42] K. Rayner. 1978. Eye movements in reading and information processing. *Psychological Bulletin* 85, 3 (1978), 618–660.

[43] Mel Rhodes. 1961. An analysis of creativity. *The Phi delta kappan* 42, 7 (1961), 305–310.

[44] Shishir Kumar Saha, Mehmet Selvi, Güral Büyükcan, and Mirza Mohymen. 2012. A systematic review on creativity techniques for requirements engineering. In *2012 International Conference on Informatics, Electronics & Vision (ICIEV)*. IEEE, 34–39.

[45] 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.

[46] 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.

[47] 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.

[48] Richard Berntsson Svensson and Maryam Taghavianfar. 2015. Selecting creativity techniques for creative requirements: An evaluation of four techniques using creativity workshops. In *2015 IEEE 23rd International Requirements Engineering Conference (RE)*. IEEE, 66–75.

[49] Ningzhi Tang, Junwen An, Meng Chen, Aakash Bansal, Yu Huang, Collin McMillan, and Toby Jia-Jun Li. 2024. Codegrits: A research toolkit for developer behavior and eye tracking in ide. In *Proceedings of the 2024 ieee/acm 46th international conference on software engineering: Companion proceedings*. 119–123.

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

[51] Klaus K Urban. 2005. Assessing creativity: The Test for Creative Thinking-Drawing Production (TCT-DP). *International Education Journal* 6, 2 (2005), 272–280.