Editor: **Ciera Jaspan**
Google
ciera@google.com

Editor: **Collin Green**
Google
colling@google.com

Developer Productivity for Humans, Part 8: Creativity in Software Engineering

Sarah Inman, Sarah D'Angelo, and Bogdan Vasilescu

SOFTWARE ENGINEERING RESEARCH typically focuses on productivity, such as measuring the speed at which developers write code or evaluating how tooling can make workflows more efficient. But as we have mentioned in this column before, software engineering is complex and creative.

Introduction

Developer productivity is more than just efficiency, so a focus on speed can cause us to lose sight of other important aspects, such as creativity. The literature has shown that creative thinking is essential to software engineering and is a key quality of great software developers.[1,2] Creativity is central to our ability to solve complex problems, and software development is, at its core, a form of problem-solving that, in turn, reinforces the importance of creativity.[3] However, the word "creative" is rarely used to describe software. This is perhaps not because software isn't creative but because creativity in software engineering might look different than in other domains.

With the recent changes in the ways we work, from new hybrid models to the rise of artificial intelligence (AI), there is an increased attention on creativity in the workplace. As software engineering continues to evolve, it is essential that we consider creativity as a core aspect of the developer experience and ensure that we are building tooling and processes to support it.

To better understand how software development tools and processes are impacting creativity, we need a better understanding of how software developers see creativity in their work. When beginning to unpack an ambiguous and complex topic such as creativity, it's important to start with the human experience. Building a definition for creativity in software engineering based on the experiences of individual developers will serve as the foundation for future progress.

In this column, we describe our approach and insights into understanding how a sample of software developers at Google defines creativity in their work. We first discuss the relevant literature in the space that shaped our approach and then outline our qualitative approach to forming a definition. Our findings suggest that creativity in software engineering centers on the concept of clever reuse rather than pure novelty. Understanding this distinction enables us to rethink how to better support and measure creativity in software engineering more broadly.

Understanding Creativity

To understand creativity in the context of software engineering, we first need to understand how creativity has been defined in the past. Researchers have been studying creativity for decades, and definitions of creativity often describe ideation with an emphasis on novelty and usefulness.[4] We leverage these definitions in our research to gather

Digital Object Identifier 10.1109/MS.2023.3340831
Date of current version: 22 February 2024

examples of creative moments by using the words "useful," "adaptable," and "elegant" in addition to "creative" to elicit responses. While these attributes are helpful starting points, they lack the depth and specificity needed to better identify creativity in software engineering workflows.

In the context of creativity in software engineering, we mainly see a focus on creativity in products and in people. To measure creative products, researchers have used metrics such as the number of new features compared to bugs[5] or the novel reuse of existing libraries.[6] Researchers have also explored how the individual characteristics of software developers influence creativity.[7,8] However, the outcomes of these studies vary in terms of what individual characteristics might predict creativity. As researchers focused on measuring and understanding the developer experience, we are not seeking to understand what characteristics of a person make them more or less creative or what aspects of a product make it creative. Rather, we are interested in understanding what makes the software development process creative and what makes the underlying code that contributes to products creative.

To define creativity as it emerges throughout the software development lifecycle, we took a qualitative approach. Because we are interested in defining creativity from a developer's perspective, we used qualitative methods that can provide rich context that is useful when exploring complex concepts. Concretely, we used a feedback and photo elicitation methodology,[9,10] in the context of a diary study, followed by a structured interview.

Feedback studies ask participants to provide information in response to prompts immediately after an event occurs, while elicitation studies ask participants to provide information (in the form of an artifact, here a screenshot) as a memory cue in follow-up interviews.[10] Elicitation studies are particularly useful for asking questions about nebulous concepts due to the known challenges of self-report methods and people's ability to recall information accurately. The prompts for the week-long diary study focused on how developers expect to be creative and what they saw as creative upon reflection on their week. The photo elicitation portion asked developers to submit a screenshot of their work that they considered "creative."

In the follow-up interviews, we used photos to cue memory in the moment and acquire further context about what parts of daily life developers find creative. We asked engineers to describe what made these artifacts creative and what aspects of their work they consider to be creative. This approach also afforded a better understanding of the connection between context and the creative process.

Ultimately, clever reuse and recombination of existing code in useful ways are the primary attributes of creativity in software engineering.

For analysis, we used a grounded theory approach, applying "codes" or descriptive labels to the week-long feedback and elicitation portion of the study and then to the interview transcripts line by line. We then used this information to identify common themes that emerged from the data, such as reuse, infrastructure, knowledge sharing, learning, novelty, constraints that aid creativity, and session. Finally, we connected the reflection prompts to specific points in the developer workflow, which helped us gain a deeper understanding of the participants' experiences.

Defining Creativity in Software Engineering

At a high level, our analysis of the diary study and follow-up interviews identified three key themes.

1. Collaboration and brainstorming foster creativity among developers.
2. Regardless of working individually or in a group, problem-solving by exploring a solution space through learning and exploration sets the stage for creativity.
3. Ultimately, clever reuse and recombination of existing code in useful ways are the primary attributes of creativity in software engineering.

Based on these findings, we consider creativity in software engineering to include problem-solving that emphasizes reusability and usefulness over originality. Next, we give more details about each of the three emerging themes.

Fostering Creativity Through Collaboration and Brainstorming

Developers submitted screenshots of planning and design stages, including team communication, ideation,

and brainstorming, which aligns with prior literature[11] highlighting brainstorming as a key practice that influences software developer creativity.

As one interviewee put it

> "All of the discussion happens on chat now, rather than in meetings or just bantering in the office. This has caused a lot more cross-pollination in this brainstorming phase because if it's in chat, despite being an ocean apart, we're all talking about the same problem."

Participants identified creativity as occurring primarily after a problem has been defined but before a solution has been decided on. It is the utility to others that distinguishes the work as creative. Enacting creativity requires a creative process, which occurs during moments of deep focus or flow and in collaboration with others.

Facilitating Creative Thinking Through Exploration

Not knowing where to find the right information or expertise can lead to problems in organizations. Among our study participants, exploring a solution space emerged as a key component of developers' creative expression. Given that having to learn something new can often be perceived as a hindrance to getting work done efficiently, it is important to unpack what kinds of learning are considered to enable creativity.

One interviewee mentioned the following:

> "I think the creative part in the development stage is when you hit snags that you're maybe unaware of when you're first designing the project. And then you have to figure out ways to resolve those within the framework that you've already set."

Participants experienced creativity in moments of learning new concepts, tools, and languages but also when learning simple fixes from colleagues. This suggests that learning can be a hindrance to getting work done when the information is difficult to find but that this is mitigated by having a culture of knowledge sharing. Gaining a deeper knowledge of the system or learning a new tool or technique was a major area in which developers experienced creativity, as one participant exemplified.

> "For me, learning what the box diagram looks like, like being able to look at the system and then go back and get a whiteboard and draw a picture of how it fits together and what talks to what … that is always a super fascinating adventure for my brain."

In distinguishing what kinds of learning are central to creativity, the need for better documentation is clear. Problem-solving without proper documentation led to burdensome knowledge requirements disconnected from the bigger picture. However, when developers are able to access documentation and become immersed in learning, it is considered a central component of creativity. This study suggests that addressing major obstacles with respect to knowledge access and documentation could lead to the types of learning more associated with creativity and potentially relieve one of the primary hindrances to productivity.

Another participant defined creativity as a process or as something enjoyable, noting that mastery requires more learning.

> "It's easier to get creative [with] art even if you don't know much. You just let your mind wander until you spot something you like. You probably won't make anything 'genius' without a deep dive, but you will still create something new and enjoy the process. For work, though, you need to actually know as many techniques, technologies, approaches, or design solutions as possible in order to be creative with your solution."

Defining Creativity as Clever Reuse

Most screenshots developers submitted were in the implementation stage of software development. These excerpts were largely focused on reuse and developing improvised solutions. While many of the screenshots included artifacts, such as design documentation to better help future developers or brainstorming documents among teammates, in the implementation stage, developers submitted screenshots of actual code, either code they had written that was helpful, reusable, or novel or code that they had refactored. The following quote illustrates a common theme around refactoring code as a form of creativity:

> "I feel satisfied when refactoring code and making it cleaner. Especially if I can come up with a different data structure of the class that makes everything easier."

In follow-up interviews, participants discussed reuse as a primary attribute of creativity and contrasted their work to more classical creative fields such as art. For example, one participant noted the following:

> "Code shouldn't be creative because it should be reusable. I think art should be creative because it's something new. Sure, we are also trying

> to write something new, but it's not so new and unseen … Most novelty or creativity is just using constructs that are known."

Overall, developers emphasize the importance of future usage of their designs when asked about creative moments in their day. Thus, while novelty plays into developers' definitions of creativity, novel or clever reuse is more critical. This includes reuse by team members or envisioned downstream usage of their software products. This aspect of reuse in a developer community is a specific justification that developers give for not considering themselves to be "creative" according to more mainstream definitions that emphasize novelty. Many developers we interviewed differentiated their work from artists in that they—unlike artists—were not focused on being novel or original. On the contrary, overemphasizing clever designs or originality in code was seen as a negative, sometimes referred to as *snowflake code*.

Creativity in Software Engineering

Reflecting on the themes emerging from our study, our main finding is that software practitioners at Google view creativity as a form of problem-solving that emphasizes reusability and usefulness over originality and novelty. Indeed, many of our informants distinguish their work from that of artists or "creatives" in that software engineering creativity includes being able to know when to reuse code, creating code or knowledge that can contribute to future use, and producing output that is useful to their team or future developers. This view tends to differ from the mainstream definitions of creativity, which typically highlight originality and novelty, and instead aligns with Boden's concept of combinational creativity, that is, "the combination of familiar ideas in unfamiliar ways."[12]

So what does this mean for the future of software engineering? The clearer understanding of developers' view of creativity that we provide suggests several new research directions at the intersection of software engineering, human–computer interaction, and machine learning. While surely not exhaustive, we outline a list of open research questions around measurement, AI/large language models (LLMs), and productivity that are informed by the work we described previously.

Measuring Creativity in Software

Going beyond developers' creative processes and perceptions of creativity, it becomes natural to ask how creativity might be operationalized in software artifacts. What are the characteristics of creativity and innovation in software? How can we measure the degree of innovation in software products and processes? The emphasis on clever reuse over absolute novelty revealed by our study offers several possible ways forward given prior work outside of software engineering. For example, researchers operationalize innovation as novel combinations of existing concepts,[13] novel combinations of citations of prior work[14] first co-occurring in a research article, or the "disruption"[15] of citations of prior work after the article deemed innovative gets published. To what extent do these operationalizations have valid analogs in software?

Creativity and LLMs

There has been an influx of AI-based developer tooling aimed at improving productivity. The introduction of LLMs into developer tooling is starting to change the way developers think about writing code, with a focus on writing code faster and automating tasks.[16] As we are at the beginning stages of AI in developer workflows, it is critical that we do not lose sight of creativity in our efforts to optimize for productivity. Indeed, recent work outside of software engineering[17] examined how experienced authors interacted with AI for creative writing tasks and found that the authors enjoyed brainstorming and adding details while aided by AI but did not want to "offload the creative process" to it. Do software developers feel similarly? If so, how might we better support creative software engineering while interacting with AI-based tools?

Creativity Versus Productivity

Finally, we argue that creativity could be at risk if we overoptimize for productivity and efficiency. Some factors that might be a hindrance to productivity in the short term could enable creativity in the long term. In our interviews, when developers moved past the expectation that creativity meant novelty, they noted that their own work was most creative when implementing code, refactoring, and collaborating with colleagues to solve a problem. Not all of these activities are when developers are typically considered at their most productive. For example, refactoring is not without controversy,[18,19] and the coordination challenges that come with collaboration are well known.[20] This suggests perhaps a different temporality to creativity than there is for productivity. To what extent does optimizing for creativity hinder productivity and vice versa? What are the tradeoffs involved?

There remains a need to better understand what creativity means in the context of software engineering, how it affects the whole software development process, and how factors like work location or AI tools might influence it. We're taking a step toward filling this knowledge gap by exploring how professional developers at Google think about creativity in their work. Leveraging qualitative methods to investigate this ambiguous topic, our findings showed that their definitions differ somewhat from the usual ones. Software developers think of creativity as making things reusable and useful rather than simply being original or new. Based on what we've learned, we can start to imagine how future software engineering research might focus more on creativity. While we've posed more questions than answers, we hope this discussion sparks new ideas and discussions about how to make software engineering better with creativity in mind.

ABOUT THE AUTHORS

SARAH INMAN is a user experience researcher on Google Cloud's Storage team at Google, Seattle, WA 98103 USA. Please contact her at icsarah@google.com.

SARAH D'ANGELO is a user experience researcher on the Engineering Productivity Research team at Google, Canterbury 7999, New Zealand. Contact her at sdangelo@google.com.

BOGDAN VASILESCU is a visiting researcher at Google and an Associate Professor at Carnegie Mellon University, Pittsburgh, PA 15213 USA. Please contact him at empirical@google.com.

References

1. P. L. Li, A. J. Ko, and A. Begel, "What distinguishes great software engineers?" *Empirical Softw. Eng.*, vol. 25, no. 1, pp. 322–352, Jan. 2020, doi: 10.1007/s10664-019-09773-y.
2. P. L. Li, A. J. Ko, and J. Zhu, "What makes a great software engineer?" in *Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng.*, 2015, pp. 700–710, doi: 10.1109/ICSE.2015.335.
3. R. L. Glass, *Software Creativity*. Upper Saddle River, NJ, USA: Prentice-Hall, 1994.
4. T. M. Amabile et al., "A model of creativity and innovation in organizations," *Res. Org. Behav.*, vol. 10, no. 1, pp. 123–167, 1988.
5. J. W. Paulson, G. Succi, and A. Eberlein, "An empirical study of open-source and closed-source software products," *IEEE Trans. Softw. Eng.*, vol. 30, no. 4, pp. 246–256, Apr. 2004, doi: 10.1109/TSE.2004.1274044.
6. H. Fang, J. Herbsleb, and B. Vasilescu, "Novelty begets popularity, but curbs participation – A macroscopic view of the Python open-source ecosystem," in *Proc. IEEE/ACM 46th Int. Conf. Softw. Eng. (ICSE)*, New York, NY, USA: ACM, 2024, pp. 643–653.
7. A. Amin et al., "The impact of personality traits and knowledge collection behavior on programmer creativity," *Inf. Softw. Technol.*, vol. 128, Dec. 2020, Art. no. 106405, doi: 10.1016/j.infsof.2020.106405.
8. W. Groeneveld, L. Luyten, J. Vennekens, and K. Aerts, "Exploring the role of creativity in software engineering," in *Proc. Int. Conf. Softw. Eng., Softw. Eng. Soc. (ICSE-SEIS)*, Piscataway, NJ, USA: IEEE Press, 2021, pp. 1–9.
9. E. Bignante, "The use of photo-elicitation in field research. Exploring Maasai representations and use of natural resources," *EchoGéo*, vol. 11, no. 11, 2010, Art. no. 11622, doi: 10.4000/echogeo.11622.
10. S. Carter and J. Mankoff, "When participants do the capturing: The role of media in diary studies," in *Proc. SIGCHI Conf. Human Factors Comput. Syst. (CHI)*, Apr. 2005, pp. 899–908, doi: 10.1145/1054972.1055098.
11. R. Mohanani, P. Ram, A. Lasisi, P. Ralph, and B. Turhan, "Perceptions of creativity in software engineering research and practice," in *Proc. 43rd Euromicro Conf. Softw. Eng. Adv.*

Appl. (SEAA), Piscataway, NJ, USA: IEEE Press, 2017, pp. 210–217.
12. M. A. Boden, *The Creative Mind: Myths and Mechanisms*. Routledge, London, U.K.: Psychology Press, 2004.
13. B. Hofstra, V. V. Kulkarni, S. Munoz-Najar Galvez, B. He, D. Jurafsky, and D. A. McFarland, "The diversity–innovation paradox in science," *Proc. Nat. Acad. Sci.*, vol. 117, no. 17, pp. 9284–9291, 2020, doi: 10.1073/pnas.1915378117.
14. B. Uzzi, S. Mukherjee, M. Stringer, and B. Jones, "Atypical combinations and scientific impact," *Science*, vol. 342, no. 6157, pp. 468–472, 2013, doi: 10.1126/science.1240474.
15. R. J. Funk and J. Owen-Smith, "A dynamic network measure of technological change," *Manage. Sci.*, vol. 63, no. 3, pp. 791–817, 2017, doi: 10.1287/mnsc.2015.2366.
16. S. Barke, M. B. James, and N. Polikarpova, "Grounded copilot: How programmers interact with code-generating models," 2022, *arXiv:2206.15000*.
17. D. Ippolito, A. Yuan, A. Coenen, and S. Burnam, "Creative writing with an AI-powered writing assistant: Perspectives from professional writers," 2022, *arXiv:2211.05030*.
18. E. Ammerlaan, W. Veninga, and A. Zaidman, "Old habits die hard: Why refactoring for understandability does not give immediate benefits," in *Proc. Int. Conf. Softw. Anal., Evol., Reeng. (SANER)*, Piscataway, NJ, USA: IEEE Press, 2015, pp. 504–507.
19. M. Kim, T. Zimmermann, and N. Nagappan, "A field study of refactoring challenges and benefits," in *Proc. Int. Symp. Found. Softw. Eng. (FSE)*, 2012, pp. 1–11.
20. M. Cataldo and J. D. Herbsleb, "Coordination breakdowns and their impact on development productivity and software failures," *IEEE Trans. Softw. Eng.*, vol. 39, no. 3, pp. 343–360, Mar. 2013, doi: 10.1109/TSE.2012.32.

Digital Object Identifier 10.1109/MS.2024.3358034