Week 11 Reviews

Zhiao Wei
University of Waterloo
Master of Computer Science in Crysp Lab
z23wei@uwaterloo.ca

1. Review of Code today, deadline tomorrow: Procrastination among software developers

Zeinabsadat Saghi, Thomas Zimmermann, Souti Chattopadhyay, Proc. of the 47th ACM/IEEE Intl. Conf. on Software Engineering (ICSE), Ottawa, April–May 2025.

Problem Being Solved

While procrastination is a well-studied phenomenon in psychology and academic settings (particularly among university students), remarkably little is known about how it manifests specifically within the software engineering industry. The authors identify a significant research gap: software development involves unique challenges—such as high cognitive load, complex abstract problem-solving, and tight task interdependence—that distinguish it from other fields. Existing research in SE focuses on burnout, happiness, or ADHD, but lacks a dedicated investigation into the unique triggers, effects, and mitigation strategies for procrastination among professional developers.

New Idea

The paper presents the first qualitative investigation (N=15 professional developers) into this topic, offering a nuanced taxonomy of procrastination in SE. The core "New Idea" is the identification of procrastination as a double-edged sword.

- **Negative Effects (14 identified):** Including technical debt, lowered code quality, strained team culture, and personal guilt/anxiety.
- **Positive Effects (8 identified):** Surprisingly, the study found benefits such as "Incubation" (creativity boost from stepping away), avoiding burnout through breaks, and "Near-deadline efficiency."
- **Triggers:** Major culprits were identified as "Task Vagueness," "Lack of Interest," and "Task Complexity."
- **Mitigation:** The authors proposed 19 mitigation techniques across four categories: Awareness, Task Focus (e.g., Fake Deadlines), Task Planning (e.g., Decomposition), and Team Support.

Positive Points

(1) **High Novelty & Relevance:** This is the first dedicated study to tackle "procrastination" directly in the SE domain, addressing a relatable but often stigmatized topic that affects developer productivity and well-being.
(2) **Nuanced Perspective:** Instead of simply framing procrastination as a vice, the authors provide a balanced view by acknowledging "Active Procrastination" and its potential benefits for creativity and incubation, which offers a fresh theoretical lens.
(3) **Actionable Outcomes:** The paper goes beyond observation to provide a concrete table of 19 mitigation strategies (Table V), making the findings immediately useful for practitioners and managers looking to improve their workflows.

Negative Points

(1) **Small Sample Size & Generalizability:** With only 15 participants recruited via snowball sampling, the findings may not be representative of the broader, global developer population (e.g., across different cultures or company sizes).
(2) **Reliance on Self-Reporting:** The study relies entirely on interviews and participant recall. This introduces bias, as participants may rationalize their procrastination (e.g., claiming it was "strategic incubation" when it was actually avoidance) or misremember their emotional states.
(3) **Lack of Quantitative Validation:** The study is purely qualitative. There is no quantitative data to prove that the suggested mitigation strategies (like "Fake Deadlines") statistically lead to reduced bug rates or faster delivery times.

Future Work

(1) **Quantitative Validation:** Conduct a large-scale survey (N > 500) to quantify the prevalence of the identified triggers and measure the statistical correlation between specific mitigation strategies and productivity metrics.
(2) **Tool Development (HCI):** Develop IDE plugins or AI-driven tools that can automatically detect "stuck" behavior (e.g., lack of typing for extended periods) and suggest interventions like Task Decomposition, moving beyond manual self-regulation.
(3) **Longitudinal Studies:** Track developers over a longer period (e.g., 6 months) to determine if "Active Procrastination" is sustainable or if it eventually leads to burnout and stress in the long run.

Rating

4.5/5. Despite the limitations of a small sample size, this paper establishes a foundational vocabulary for a new sub-field in developer productivity. The taxonomy of triggers and effects is robust and provides a necessary starting point for future quantitative research.

Discussion Summary

(1) **The Myth of "Good" Procrastination**
- *Class Perspective:* Several students questioned the validity of "active procrastination," arguing it often resembles prioritizing personal life rather than a deliberate engineering strategy. Another perspective was that procrastination might be inherent and not necessarily a "problem" as long as deadlines are met. Specific examples were shared, such

as ASE conference submissions where volume spikes 10x in the final three days, or peers who successfully start writing just 12 hours before a deadline. However, the professor noted that while common, this behavior creates significant logistical headaches for organizers (e.g., program chairs) who struggle with planning.

- *My Response:* I argued that the distinction lies in intention. If the delay is driven by fear, it is negative. However, if the brain is performing "background processing" (Incubation) on a complex algorithm while stepping away, it is a valid engineering process.

(2) **Privacy vs. Productivity Tools**

- *Class Perspective:* Students raised strong concerns about companies using IDE trackers, commit-count dashboards, and biometric data. A key argument was that these metrics are easily misinterpreted, especially by non-technical managers who might equate "productivity" with raw activity. For instance, a student noted that a large number of commits might reflect trivial changes, while deep, meaningful work might result in very few commits. The consensus was that monitoring is only helpful if interpreted by someone with technical understanding; otherwise, it feels like surveillance.
- *My Response:* I emphasized that Data Ownership is the dealbreaker. If tools are for "Self-Quantification" (private to the developer), they are acceptable. However, if data is shared with management for evaluation, it becomes surveillance, which ironically increases anxiety and procrastination.

(3) **Will AI Cure or Worsen Procrastination?**

- *Class Perspective:* Opinions were split. Some felt AI helps overcome the "cold start" problem, especially in unfamiliar domains or research tasks, by lowering the barrier to entry. Conversely, others argued AI encourages delay because developers might believe the model will "bail them out" at the last minute. A critical point raised was that relying on AI often leads to *underestimating* the effort required—AI code often requires significant debugging and verification, which ends up costing more time. Some also noted that in industry, the existence of AI is sometimes used to justify tighter deadlines, increasing pressure rather than relieving it.
- *My Response:* I proposed that AI solves the "Cold Start" problem (removing the fear of the blank page) but creates a new issue I call "Validation Procrastination." We might generate code instantly but then delay the difficult, tedious work of verifying and understanding it, meaning we start faster but don't necessarily finish faster.

Other Comments

I think this paper is particularly timely given the current shifts in the software industry, including the rise of remote work and AI-assisted coding. The distinction between "Active" and "Passive" procrastination is a valuable mental model that helps de-stigmatize natural work rhythms in creative professions like software engineering.