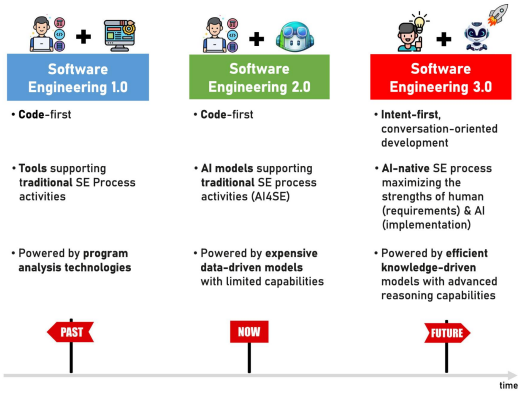
TOWARDS AI-NATIVE SOFTWARE ENGINEERING (SE 3.0): A VISION AND A CHALLENGE ROADMAP

Presenter: Haonan Zhang

January 16, 2025

UNIVERSITY OF WATERLOO

Different stages of SE



Significant cognitive overload on the human

Source: GitHub Copilot

Inefficient model training on domain-specific tasks

An example of the performance of FMs used for domain-specific tasks

Model	Cross-Task			Cross-Website			Cross-Domain		
	Ele. Acc	Op. F1	Step SR	Ele. Acc	Op. F1	Step SR	Ele. Acc	Op. F1	Step SR
Supervised Fine-Tuning									
FLAN-T5									
– Base	40.5	74.4	37.3	28.7	69.6	27.9	38.2	69.1	36.2
– Large	52.2	70.7	48.8	35.3	65.8	32.7	41.9	64.6	39.5
– XL	**56.8**	74.6	**52.5**	**42.6**	69.9	**39.5**	**43.8**	65.2	**40.7**
BLIP2-T5									
– Base	39.5	**74.9**	36.1	34.0	70.8	32.2	38.2	72.8	37.5
– Large	50.0	72.1	46.0	39.5	71.5	36.3	40.9	70.1	39.4
– XL	52.9	**74.9**	50.3	41.7	**74.1**	38.3	**43.8**	**73.4**	39.6
In-Context Learning									
GPT-3.5	19.4	59.8	16.8	14.9	56.5	14.1	25.5	57.9	24.2
GPT-4	40.2	63.4	31.7	27.4	61.0	27.0	36.2	61.9	29.7
SEEACT									
– Attributes	4.7	39.5	4.7	9.7	37.8	9.7	16.0	41.4	15.3
– Annotation	15.1	66.5	13.0	11.3	63.4	10.5	16.5	65.1	14.7
– Choice	**48.9**	**69.1**	**40.6**	**48.5**	**70.6**	**41.7**	**44.0**	**70.9**	**40.9**
– Oracle	72.9	80.9	65.7	74.4	83.7	70.0	72.8	73.6	62.1

"The most effective grounding strategies we explored in this paper still exhibit a 20-25% performance gap compared to oracle grounding"

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2025. GPT-4V(ision) is a generalist web agent, if grounded. In Proceedings of the 41st International Conference on Machine Learning (ICML'24)

Unaffordable resource usage

It is very expensive to fine-tune the models for improved domain-specific capability

“We predict that to achieve 95% accuracy for in-domain low-level tasks, 1M episodes would be required, while 2M episodes would be required to obtain 95% episode completion rates for 5-step high-level tasks”

Li, Wei, et al. On the Effects of Data Scale on Computer Control Agents. In Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS'24)

Suboptimal code quality produced by FMs

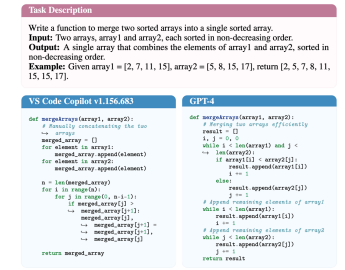
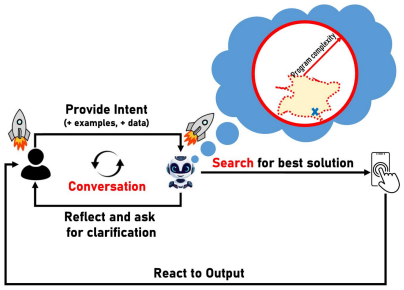


Figure 1: Example codes with distinct time complexity generated by Copilot and GPT-4, respectively. Code accessed on January 15, 2024.

“While functionally correct, this approach suffers from sub-optimal time complexity and space complexity”

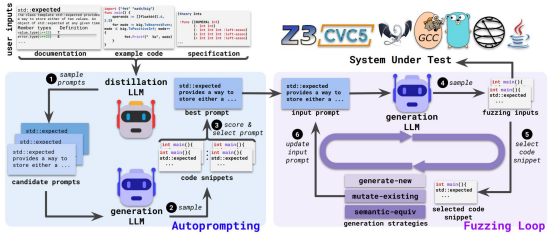
Dong HUANG, Yuhao QING, Weiyi Shang, Heming Cui, and Jie Zhang. EffiBench: Benchmarking the Efficiency of Automatically Generated Code. In Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS'24)

Vision of Software Engineering 3.0



Development is no longer driven by code but by intents expressed through back-and-forth conversations between human developers and their AI teammates.

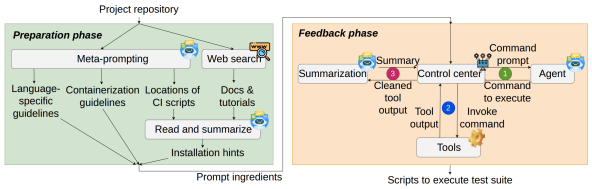
What fuzzing looks like in SE 3.0



What test execution looks like in SE 3.0

You Name It, I Run It: An LLM Agent to Execute Tests of Arbitrary Projects

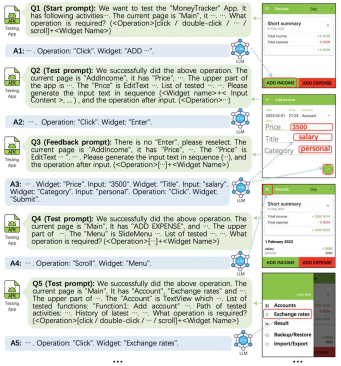
ISLEM BOUZENIA, University of Stuttgart, Germany
MICHAEL PRADEL, University of Stuttgart, Germany



What GUI testing looks like in SE 3.0

Make LLM a Testing Expert: Bringing Human-like Interaction to Mobile GUI Testing via Functionality-aware Decisions

Zhe Liu[1,2],Chunyang Chen[3], Junjie Wang[1,2,*], Mengzhuo Chen[1,2], Boyu Wu[2,4], Xing Che[1,2], Dandan Wang[1,2], Qing Wang[1,2,5,*]
[1]State Key Laboratory of Intelligent Game, Beijing, China
Institute of Software Chinese Academy of Sciences, Beijing, China;
[2]University of Chinese Academy of Sciences, Beijing, China; *Corresponding author;
[3]Technical University of Munich, Munich, Germany; [4]Ubix Global Inc;
[5]Science & Technology on Integrated Information System Laboratory
liuzhe181@mails.ucas.ac.cn,Chunyang.chen@monash.edu,junjie@ucas.ac.cn,wq@ucas.ac.cn



What GUI testing looks like in SE 3.0

Video Demo

Source: https://osu-nlp-group.github.io/SeeAct/

Balance between ask too many and not asking enough

Optimizing Instructions and Demonstrations for Multi-Stage Language Model Programs

Krista Opsahl-Ong[1], Michael J Ryan[1], Josh Purtell[2], David Broman[3], Christopher Potts[1], Matei Zaharia[4], Omar Khattab[1]

[1]Stanford University, [2]Basis, [3]KTH Royal Institute of Technology [4]UC Berkeley

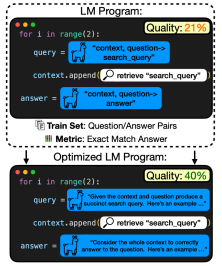


Figure 1: An example of the optimization problem we explore, shown for a multi-hop retrieval LM program. Given some question–answer pairs and a metric, the optimizer proposes new instructions and bootstraps new demonstrations (not pictured) for each stage.

Improving the efficiency of code synthesis

SynCode: LLM Generation with Grammar Augmentation

Shubham Ugare
University of Illinois Urbana-Champaign, USA

Tarun Suresh
University of Illinois Urbana-Champaign, USA

Hangoo Kang
University of Illinois Urbana-Champaign, USA

Sasa Misailovic
University of Illinois Urbana-Champaign, USA

Gagandeep Singh
University of Illinois Urbana-Champaign and VMware Research, USA

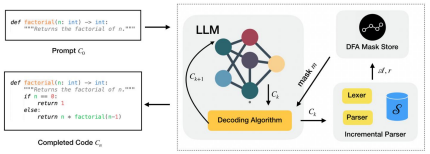


Figure 1: In the SynCode workflow, the LLM takes partial output C_k and generates a distribution for the next token t_{k+1}. The parser processes C_k to produce accept sequences $\mathcal{A}$ and remainder r. These values are used by the DFA mask store to create a token mask, eliminating syntactically invalid tokens. The LLM iteratively generates a token t_{k+1} using the distribution and the mask, appending it to C_k to create the updated code C_{k+1}. The process continues until the LLM returns the final code C_n based on the defined stop condition.

Improving runtime performance

Video

Source: https://osu-nlp-group.github.io/SeeAct/

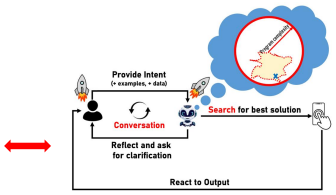
Positive points

- Redefine software engineering in the era of LLM
- Reveal the challenges we need to address to shift to SE 3.0.

Negative points

- Mostly from a theoretical perspective…
- Blurry boundary between SE 2.0 and SE 3.0?

the human developer. In our experience, a typical programming session looks as follows: create a class, write the constructor's signature and have the copilot autocomplete the implementation, create a new method, write a code comment inside the body of that method (e.g., "`## Clone a GitHub repository`") to induce copilot to generate code, rewrite the comment to force copilot to use some specific package, run tests, discover that tests failed because a key requirement was missed, reason about three alternative copilot suggestions while coding the fix, finish writing the fix, rerun the tests, and so on. As the example highlights, the process is inherently task-driven and puts a significant cognitive overload on the human.



Rating

Theoretical foundation: 5/5

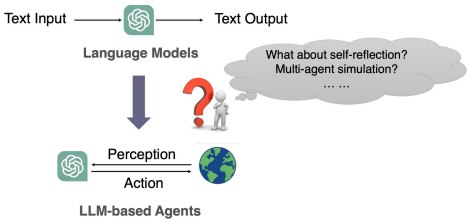
Practical experience: 4/5

Overall: 4.5/5

Discussion

- Is SE 3.0 = SE based on agent?

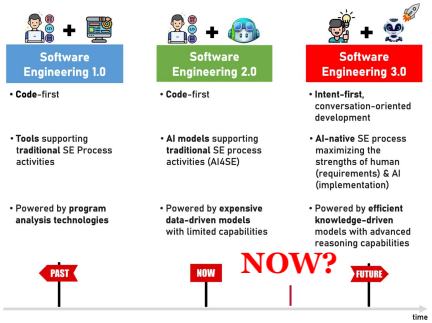
'Modern' agent = LLM + external environment?



Source: https://ysu1989.github.io/resources/language_agents_Yu0u_2024.pdf

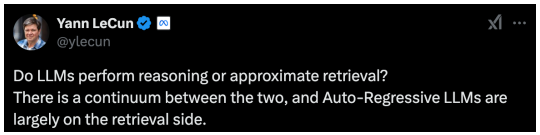
Discussion

- We are now at SE 2.0 or 2.5?



Discussion

- Do FMs really think or just content retrieval?

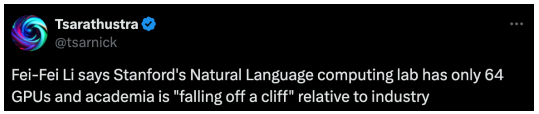


- If just content retrieval, then can we ever shift to SE 3.0?

Source: https://x.com/

Discussion

- Is Prompt Engineering the only thing we can do as a SE researcher?



Source: https://x.com/

UNIVERSITY OF WATERLOO