Towards Human-like Software Testing

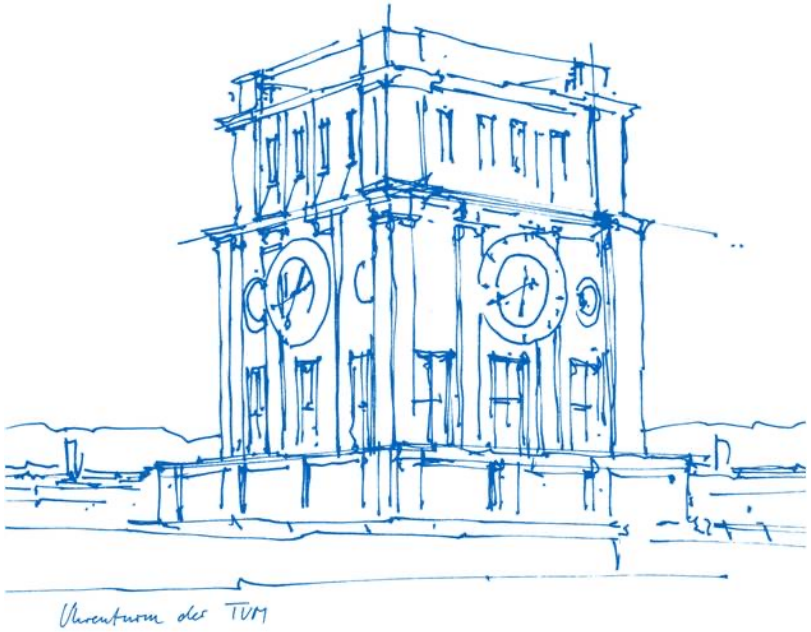
Chunyang Chen

Chair for Software Engineering & AI

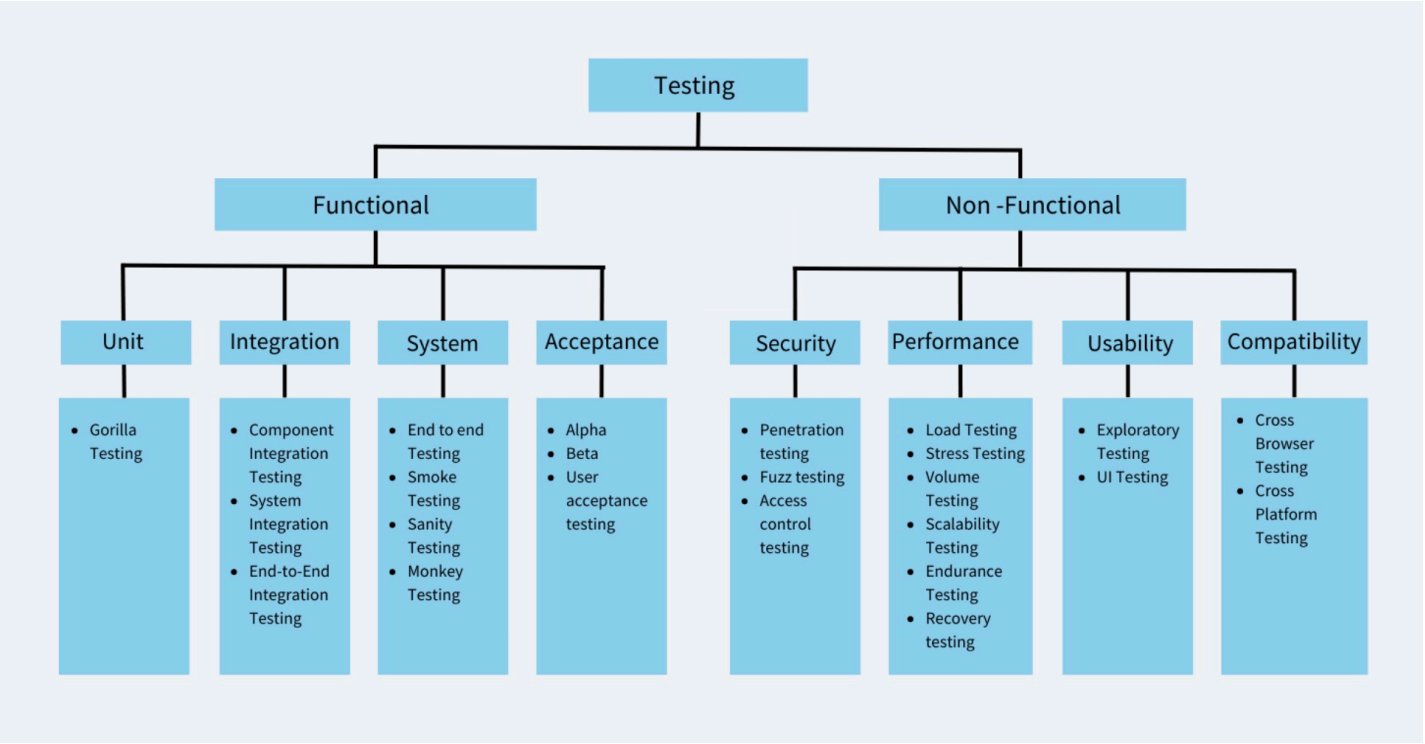
TUM School of Computation, Information and Technology

Technische Universität München

Germany



Software Testing Methods



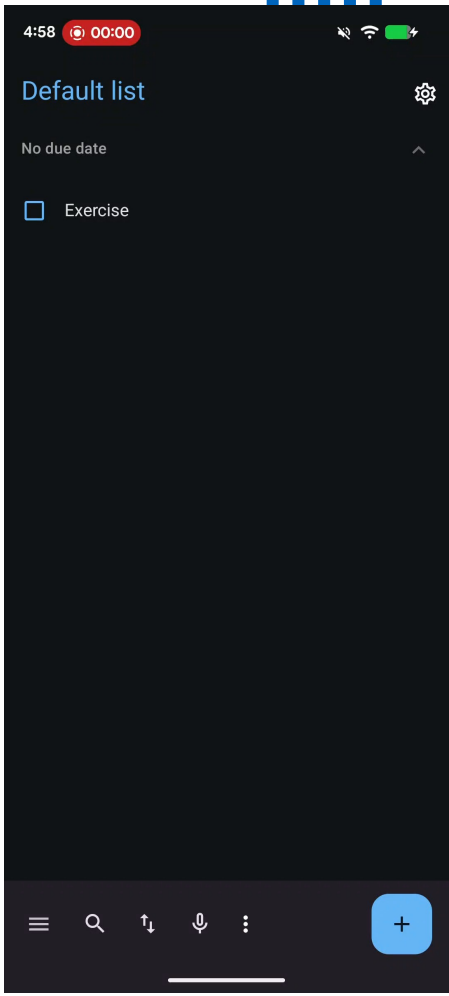
False positives, and low-value bugs

Impossible-to-perform-by-humans event sequences:

- Monkey rotates the screen 5–10 times per second while the app is initializing a Fragment

- Enter→back→enter in millisecond intervals

- Monkey fires BACK and MENU at the same time while touching two points outside the visible region.



Also apply to other testing methods

Understanding the
Monkey f

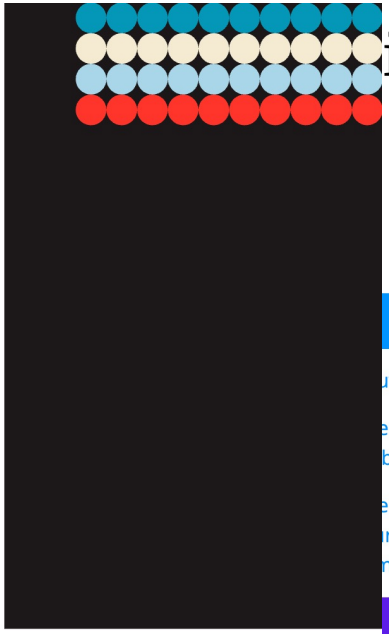
Huiyu Liu[1], Qichao Kong[1], Jue W

[1] Shanghai Key Laboratory of Ti
Uni
hysun@
[2] Nanjin

Abstract. Automated GUI te
Android apps. MONKEY is a wi
generation (AIG) tool to efficie
Android apps. However, it faces
it detects. To deeply understan
challenges, we conducted a co
ity issues of MONKEY with A
capability to reproduce crash
ity and explored the root caus
six popular open-source apps a
on them to monitor the invoca
Subsequently, we performed G
mented apps for 6,000 test case
each bug, we replayed it 200 ti
calculated the success rate. Th
ing files, log files of event han
we pinpointed five root causes
ity issues: Injection Failure, E
Loading, and Dynamic Conten
of the replays successfully repr
MONKEY's limitations in consi

Lessons from Building Static Analysis Tools at Google

SOFTWARE BUGS COST developers and software companies a great deal of time and money. For example, in 2014, a bug in a widely used SSL implementation ("goto fail") caused it to accept invalid SSL certificates,[36] and a bug related to date formatting caused a large-scale



Not integrated. The tool is not integrated into the developer's workflow or takes too long to run;

Not actionable. The warnings are not actionable;

Not trustworthy. Users do not trust the results due to, say, false positives;

ing a Large

SIMILAR ARTICLES

uveau: A 2.6.33 Surprise

en Source Collaboration is a
bal Endeavor

e Place to Manage Your Open
rce Projects and
mmunities

BROWSE CATEGORIES

en Source

Research

jects

Metric-driven Testing

- **Test Coverage:** requirement, test case, code coverage…
- **Defect Metrics:** Defect density, defect leakage, defect removal efficiency reopen rate…
- **Test Execution:** Test execution rate, pass/fail ratio, blocked test cases, test completion percentage...

- Not all software need expensive thorough testing
- Software are developed for human

Over testing?! Or insufficient testing?!

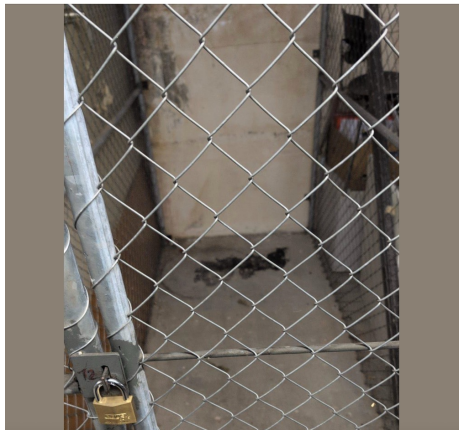
“*We don’t see things as they are; we see them as we are.* ”

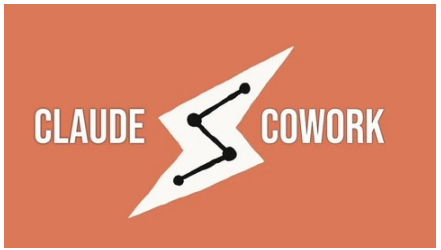
--Anaïs Nin

Beyond Software Behavior: Unexpected Human Behavior

I'm renting my cellar space in Winzererstrasse 86, 80797 Munich.

Cost 80€ per month.
If anyone is interested, DM me.



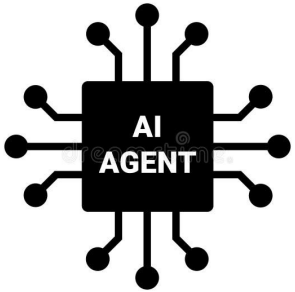


Human-centric Testing
On-demand Testing

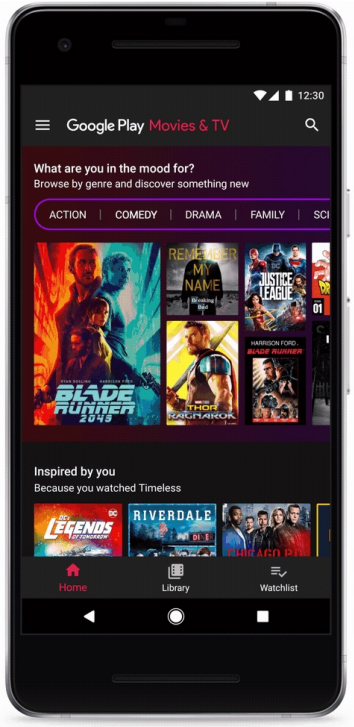
Human testing is still expensive

Automated Human-like Testing

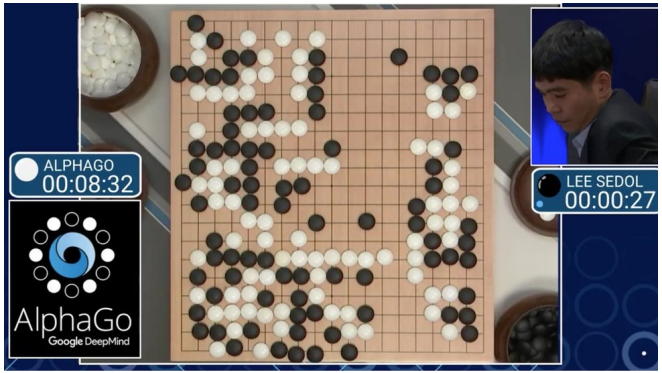
=



+



AI Agents Encoding Human Behavior

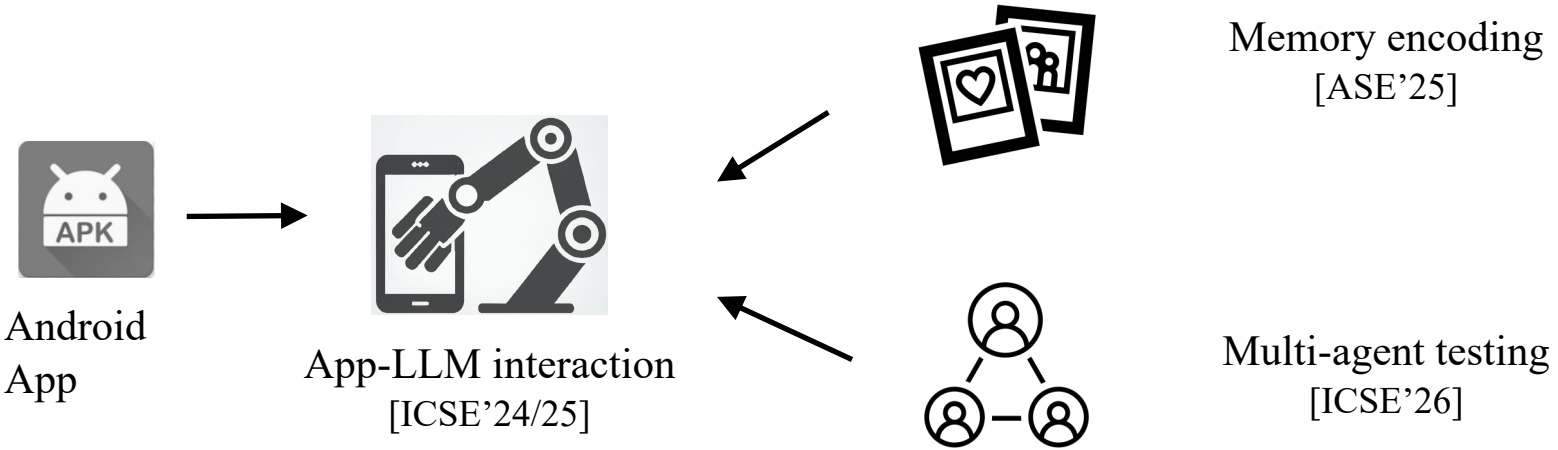


Deepmind AlphaGo

OpenAI's Dota play

Generative agents: Interactive simulacra of human behavior, UIST'24,

LLM-driven Automated Human-like App Testing



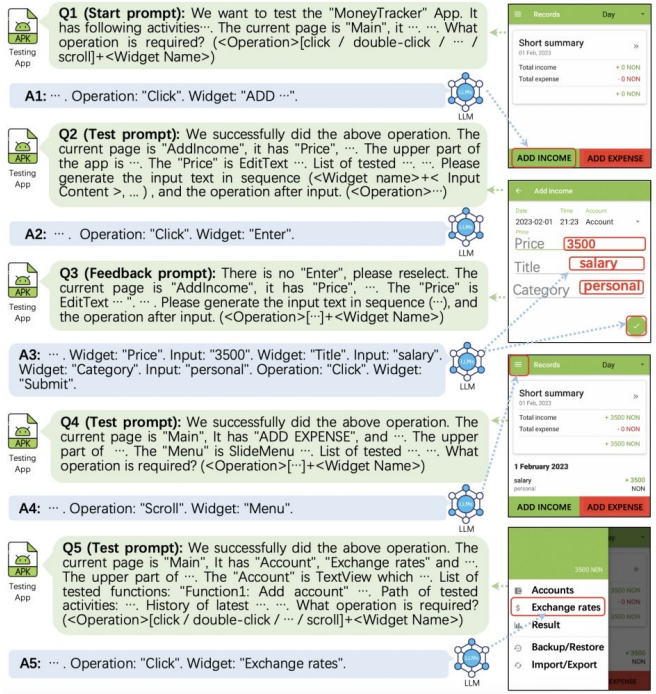
1. App-LLM interaction

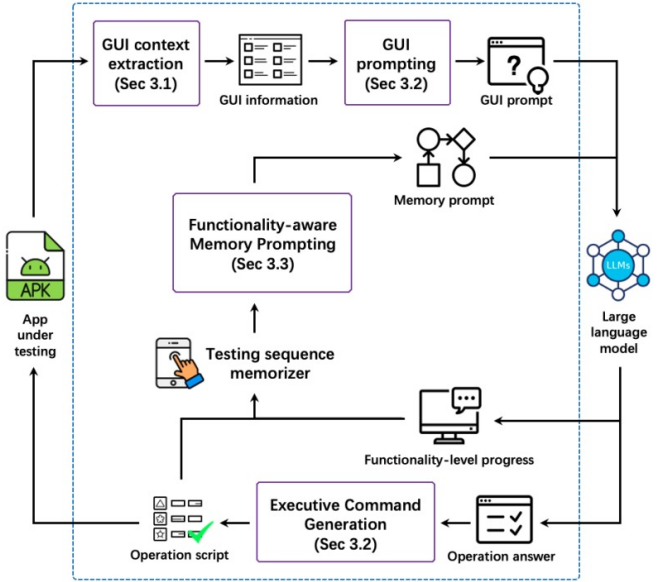
Inspired by ChatGPT, we formulate the GUI testing problem as a questions & answering (Q&A) task, i.e., asking the LLM to play the role as a human tester to test the target app.

GPTDroid

Liu, Z., Chen, C., Wang, J., Chen, M., Wu, B., Che, X., ... & Wang, Q. (2023). Make LLM a Testing Expert: Bringing Human-like Interaction to Mobile GUI Testing via Functionality-aware Decisions. *ICSE'24, arXiv:2310.15780*.

App-LLM interaction





Liu, Z., Chen, C., Wang, J., Chen, M., Wu, B., Che, X., ... & Wang, Q. (2023). Make LLM a Testing Expert: Bringing Human-like Interaction to Mobile GUI Testing via Functionality-aware Decisions. *ICSE'24, arXiv:2310.15780*.

App-LLM interaction

Table 3: The example of linguistic patterns of functionality-aware memory prompts and generation rules.

Id	Type	Sample of linguistic patterns/rules	Instantiation
		Long-term functionality-aware memory patterns: *FunctionMemory*	
1	Explored functionalities	**List of tested functions:** <*Function Name*> + <*Visits Time*> + <*Status*>, <*Function Name*> + <*Visits Time*> + <*Status*> ...	**List of tested functions:** "Function1: Add your income. Visits: 3. Status: Finished", "Function2: Delete information. Visits: 2. Status: Finished", ...
2	Covered activities.	**Path of tested activities:** <*Activity Name*> + <*Visits Time*>, <*Activity Name*> + <*Visits Time*>, ...	**Path of tested activities:** "Activity: Main. Visits: 3", "Activity: Account. Visits: 4", "Activity: AddAccount. Visits: 3", ...
3	Recently tested operations	**History of latest tested pages and operations: Latest 5th step** tested the <*Activity Name*> page. The following widgets with visits time of this page have been tested: <*Widget Name*> + <*Visits Time*>, ... The following executive command achieve the page transition: <*Operation*> + <*Widget Name*>. **Latest 4th step** tested the <*Activity Name*> page. The following ... ... **Latest 1st step** tested the <*Activity Name*> page. The following ...	**History of latest tested pages and operations: Latest 5th step** tested the "Exchange" page. The following widgets with visits time of this page have been tested: "Widget: Add exchange, Visits:2", "Widget: Submit exchange, Visits:1", 'Widget: Cancel, Visits:1" ... . The following executive command achieve the page transition: "Click" the "Exchange". **Latest 4th step** tested the "main" page. The following widgets ... ... **Latest 1st step** tested the "Add Account" page. The following widgets ...
		Function question patterns: *FunctionQuestion*	
4	Functionality inquiry	Functionality inquiry + <*Output Template*>	What is the functions currently being tested? Are we testing a new function? (<*FunctionName*> + <*Status*>)
		Functionality-aware memory prompt generation rules	
1	**Functionality-aware memory Prompt:** *FunctionMemory*[1,2,3] + *FunctionQuestion*[4]		

Liu, Z., Chen, C., Wang, J., Chen, M., Wu, B., Che, X., ... & Wang, Q. (2023). Make LLM a Testing Expert: Bringing Human-like Interaction to Mobile GUI Testing via Functionality-aware Decisions. *ICSE'24, arXiv:2310.15780*.

App-LLM interaction for GUI Testing

- App activity coverage: 0.71
- 135 real-world bugs from 216 apps

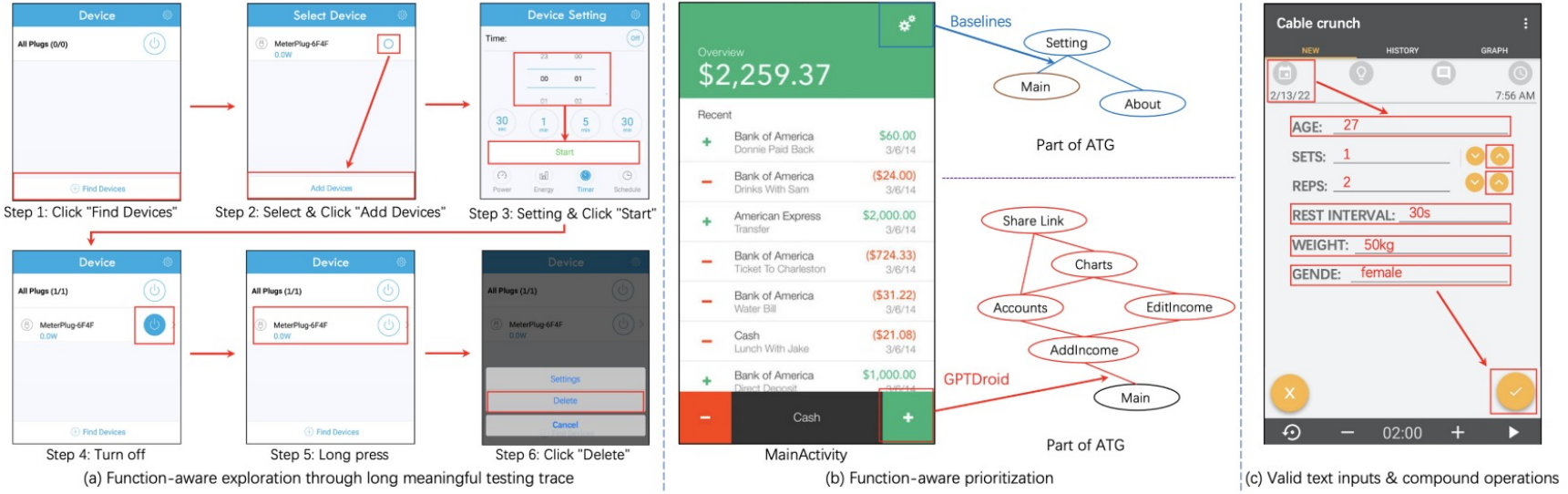
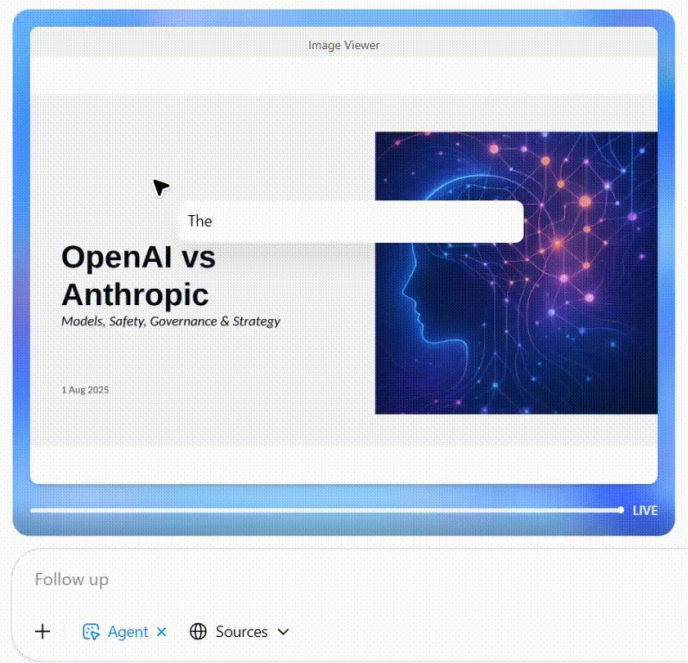
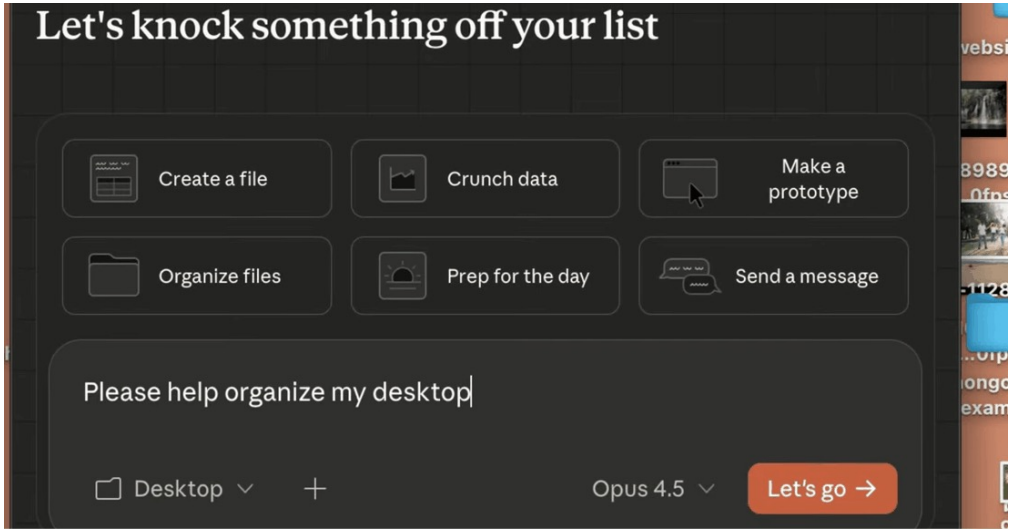


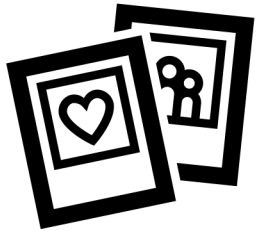
Figure 7: Examples of our insights from experiments.

Computer use agent

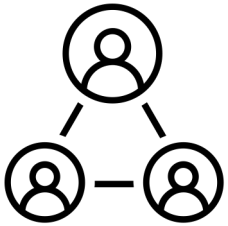




Better Model Human User Behavior



Experience

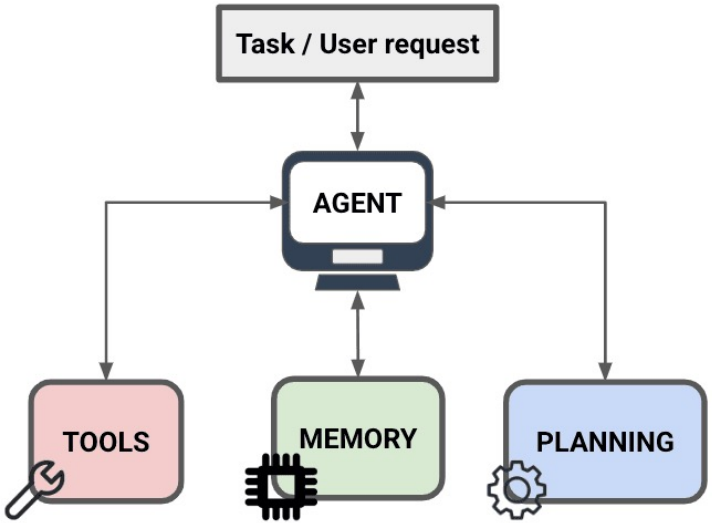


Interaction

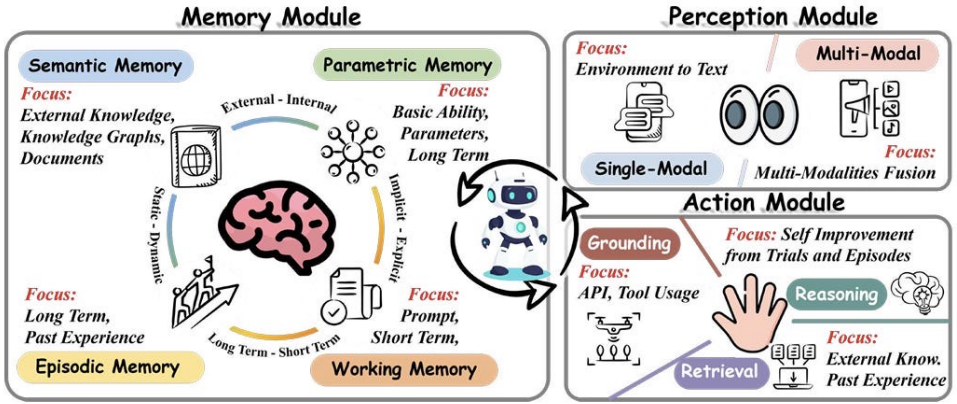
Humans grow up with diverse experiences

AI-generated

AI agents with memory



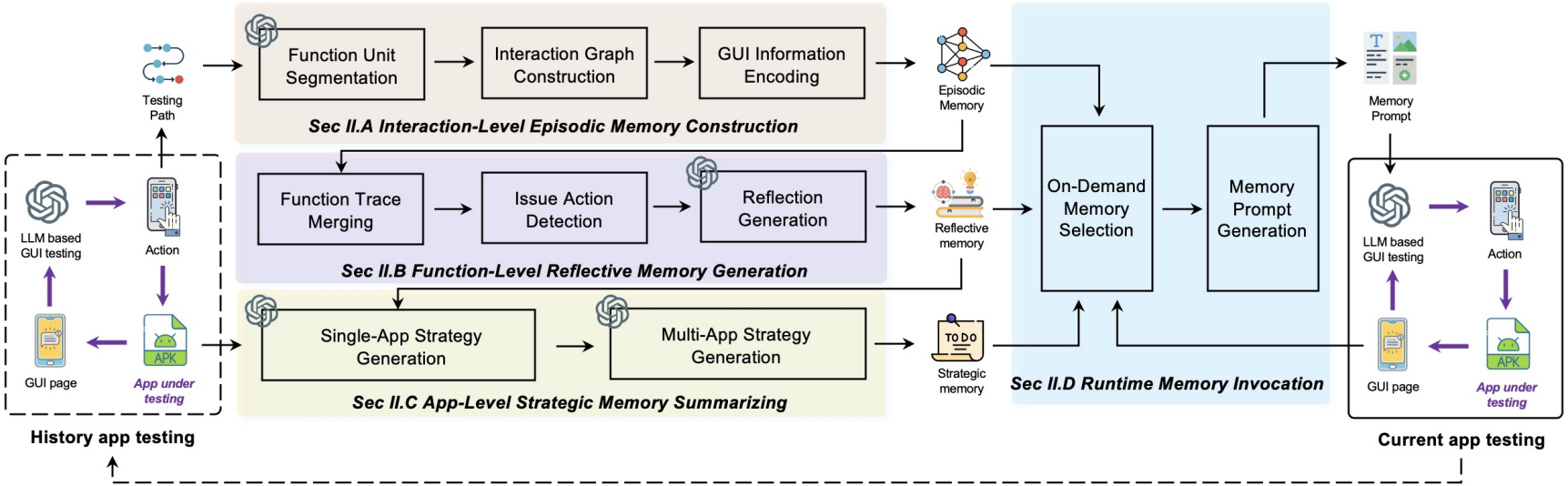
promptingguide.ai/agents/introduction



nb-data.com/p/advancing-ai-lifelong-learning-roadmap

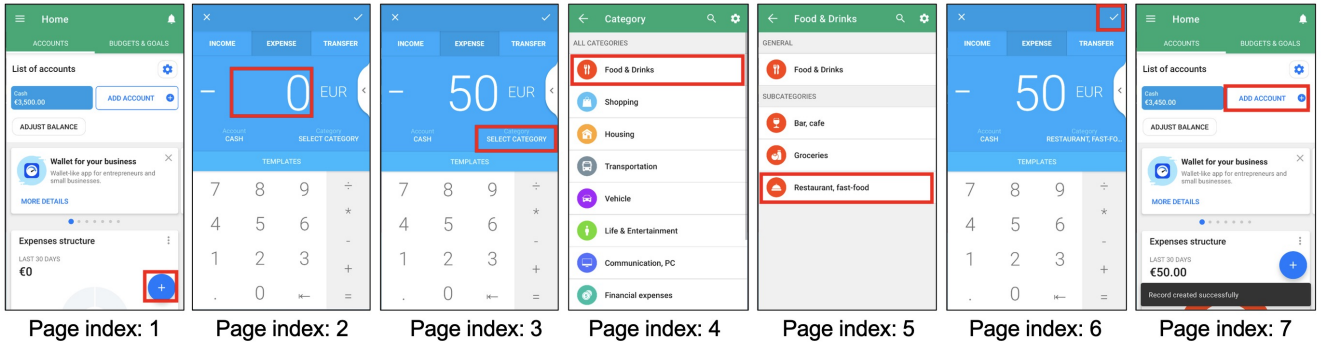
2. Memory construction for GUI testing agent

- Interaction-Level Episodic Memory
- Function-Level Reflective Memory
- App-Level Strategic Memory



Chen, M., Liu, Z., Chen, C., Wang, J., Wu, B., Hu, J., & Wang, Q. (2025). Beyond Static GUI Agent: Evolving LLM-based GUI Testing via Dynamic Memory. *ASE'25,*

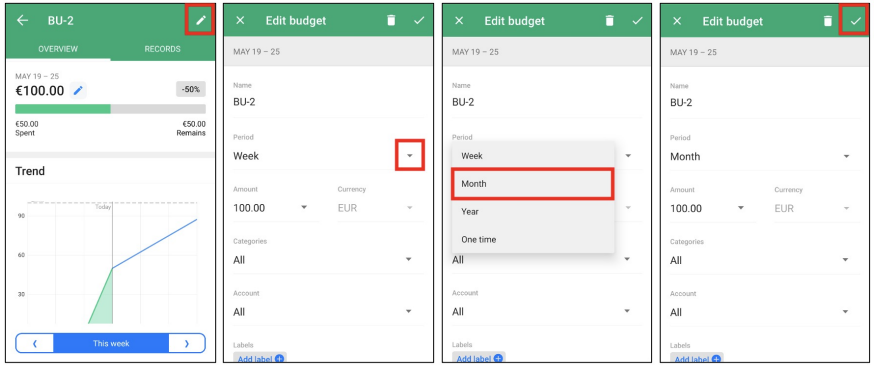
Memory construction for GUI testing agent



(a) Screenshot Annotation

1. Page index: "1"; Activity name: "Home"; Action type: "click"; label: "Add"
2. Page index: "2"; Activity name: "Add …"; Action type: "input"; Text: "50"
3. Page index: "2"; Activity name: "Add …"; Action type: "click"; Text: "SELLECT …"
4. …

(b) Textual Seq



(a) Function unit

```
"functional_description": "Editing … ",
 "execution_count": 4,
 "average_path_length": 4,
 "unique_screen_count": 3,
 "duplicate_screens": {
   "BudgetDetailActivity": 4,
   "EditBudgetFormActivity": 4
 },
 "screen_occurrence_distribution": {
   "BudgetDetailActivity": 4,
   "EditBudgetFormActivity": 4,
   "DropdownActivity": 4
 },
 …
```

(b) Testing Behavior Summary

Function: Edit budget period …
Observed Issue: Same edit operation repeated 4 times with no visible state change …
Behavior Summary:
1. Confirmed each time via the submit button.
2. Always returned to the same overview screen.
3. …

Cause: Period change not applied …
Strategy Suggestion: Detect repeated UI loops without …
…

(c) Reflective memory

Memory construction for GUI testing agent

App Name: DailyLedger

Category: Finance

App Description: DailyLedger is a personal finance app that … .

Functionalities: [Add Expense], [Edit Record], [View Summary], …

Reflective Summaries:

1. [Add Expense]: Input failed silently when leaving the … .
2. [Edit Record]: After editing a record and saving, the same record was duplicated … .
3. [View Summary]: …

Query: Based on previous testing experiences, summarize the testing process. Provide a testing strategy for future tests to improve the bug detection and avoid redundant interactions.

(a) Prompt

Global Testing Plan:

1. [Core functional priorities]: Prioritize coverage of the Add Record, Edit Record, … , which constitute the primary usage loop for finance app. …
2. [Suggested testing order]: …

…

Function-Level Strategy:

1. [Add Record]: Ensure non-zero amount and non-empty …
2. [Edit Expense]: When testing editing workflows, prioritize checking whether changes are saved once or multiple times …

…

Operations Trigger Bugs:

1. [Delete]: Execute Delete while scrolling through the list, …

(b) Strategic memory

On-Demand Memory Invocation:

- cold start: strategic memory
- mid-testing stagnation: call reflective memory when exploration stalls
- functional transitions: fetch episodic units matching the current page.

Memory as a plugin for GUI testing tools

TABLE I: Comparison between MemoDroid and baselines.

GUI Testing Method	Average Coverage Activity	Average Coverage Code	Average Bugs
GPTDroid	0.29	0.27	0.50
GP + MemoDroid	**0.52 ↑ 79%**	**0.49 ↑ 81%**	**1.47 ↑ 194%**
DroidAgent	0.28	0.27	0.57
DA + MemoDroid	**0.55 ↑ 96%**	**0.51 ↑ 89%**	**1.70 ↑ 198%**
AUITestAgent	0.37	0.36	1.10
ATA + MemoDroid	**0.69 ↑ 86%**	**0.67 ↑ 86%**	**1.73 ↑ 57%**
VisionDroid	0.41	0.39	1.23
VD + MemoDroid	**0.77 ↑ 88%**	**0.73 ↑ 87%**	**2.10 ↑ 71%**
Guardian	0.40	0.38	1.30
GU + MemoDroid	**0.77 ↑ 93%**	**0.75 ↑ 97%**	**2.23 ↑ 72%**

TABLE II: Comparison between MemoDroid and variants.

GUI Testing Method	Average Coverage Activity	Average Coverage Code	Average Bugs
GP + MemoDroid	**0.52**	**0.49**	**1.47**
w/o Episodic Memory	0.39 ↓ 25%	0.38 ↓ 22%	1.13 ↓ 23%
w/o Reflective Memory	0.36 ↓ 31%	0.34 ↓ 31%	1.00 ↓ 32%
w/o Strategic Memory	0.44 ↓ 15%	0.41 ↓ 16%	1.23 ↓ 16%
DA + MemoDroid	**0.55**	**0.51**	**1.70**
w/o Episodic Memory	0.41 ↓ 25%	0.39 ↓ 24%	1.30 ↓ 24%
w/o Reflective Memory	0.34 ↓ 38%	0.32 ↓ 37%	1.03 ↓ 39%
w/o Strategic Memory	0.43 ↓ 22%	0.41 ↓ 20%	1.37 ↓ 19%
ATA + MemoDroid	**0.69**	**0.67**	**1.73**
w/o Episodic Memory	0.52 ↓ 25%	0.51 ↓ 24%	1.40 ↓ 19%
w/o Reflective Memory	0.49 ↓ 29%	0.47 ↓ 30%	1.27 ↓ 27%
w/o Strategic Memory	0.55 ↓ 20%	0.53 ↓ 21%	1.50 ↓ 13%
VD + MemoDroid	**0.77**	**0.73**	**2.10**
w/o Episodic Memory	0.57 ↓ 26%	0.52 ↓ 29%	1.80 ↓ 14%
w/o Reflective Memory	0.53 ↓ 31%	0.49 ↓ 33%	1.70 ↓ 19%
w/o Strategic Memory	0.61 ↓ 21%	0.57 ↓ 22%	1.87 ↓ 11%
GU + MemoDroid	**0.77**	**0.75**	**2.23**
w/o Episodic Memory	0.59 ↓ 23%	0.57 ↓ 24%	1.77 ↓ 21%
w/o Reflective Memory	0.58 ↓ 25%	0.56 ↓ 25%	1.60 ↓ 28%
w/o Strategic Memory	0.62 ↓ 19%	0.59 ↓ 21%	1.83 ↓ 18%

More experience → better performance

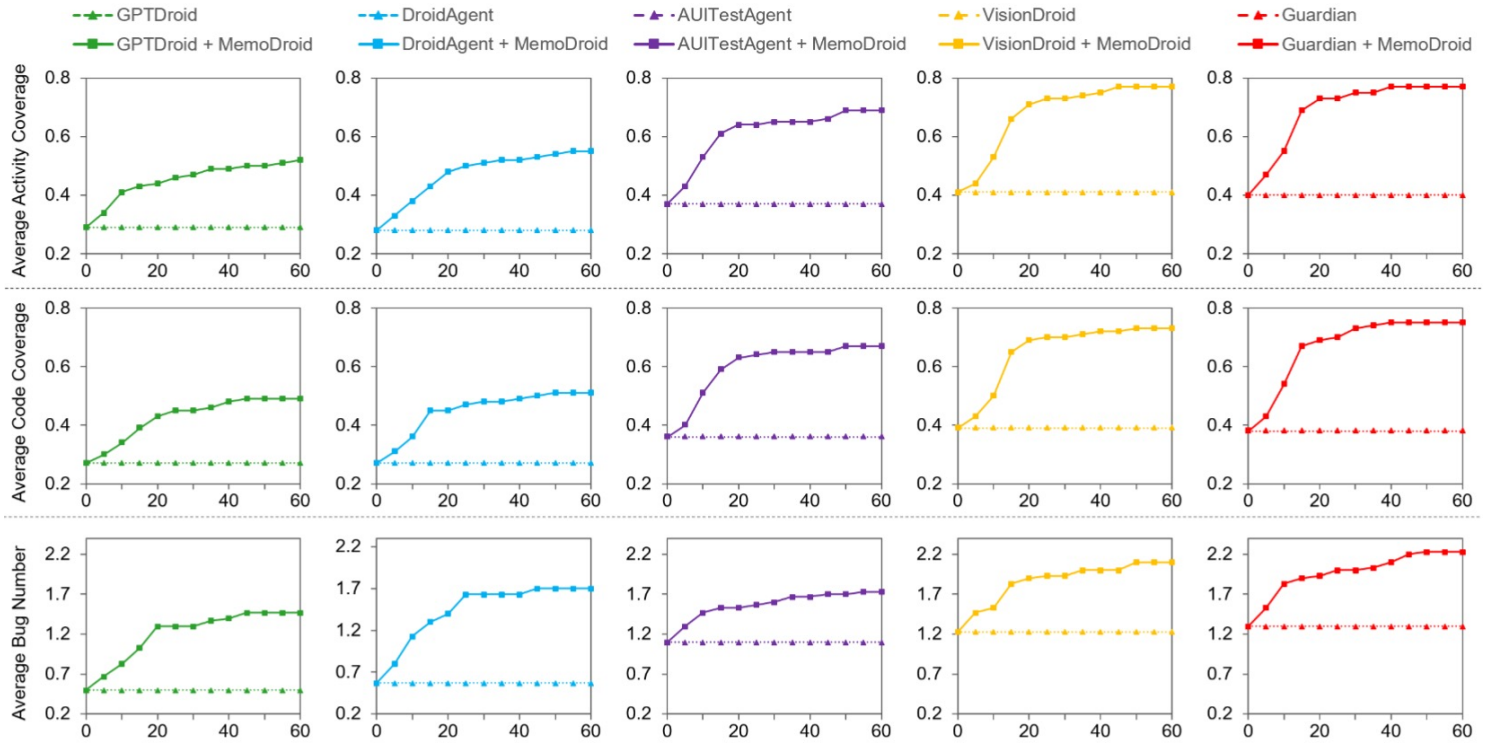


Fig. 5: Impact of memory pool evolving

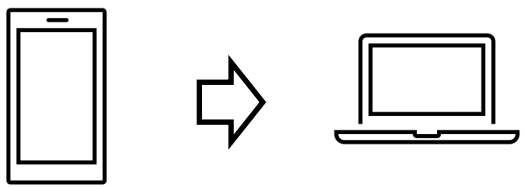
More discussions

Memory across versions

TABLE IV: Performance of different versions of an app.

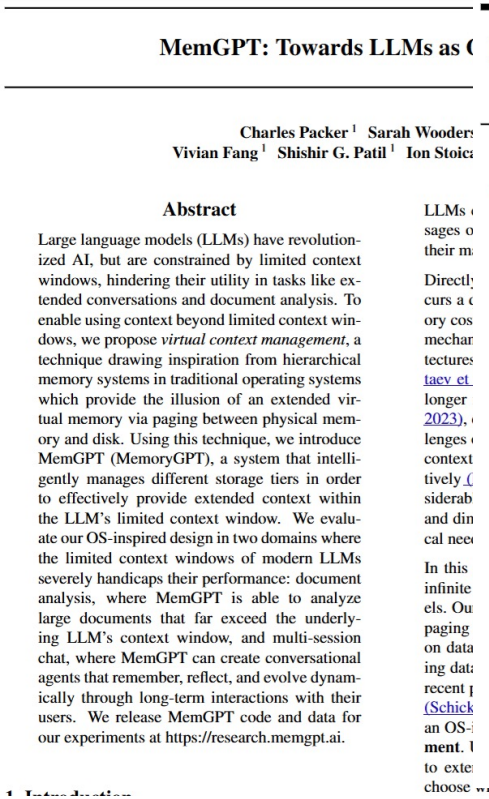
Testing Method	Activity Coverage (different versions)					All Bugs
	v 2.0	v 2.1	v 2.4	v 4.0	v 5.0	
GPTDroid	0.23	0.25	0.25	0.26	0.23	1
GP+MemoDroid	**0.23**	**0.33**	**0.39**	**0.41**	**0.43**	**2**
DroidAgent	0.27	0.26	0.28	0.26	0.29	1
DA+MemoDroid	**0.27**	**0.35**	**0.41**	**0.41**	**0.46**	**2**
AUITestAgent	0.29	0.28	0.30	0.29	0.31	1
ATA+MemoDroid	**0.29**	**0.37**	**0.38**	**0.44**	**0.49**	**3**
VisionDroid	0.33	0.32	0.35	0.29	0.31	2
VD+MemoDroid	**0.33**	**0.45**	**0.57**	**0.59**	**0.59**	**4**
Guardian	0.30	0.34	0.29	0.33	0.34	3
GU+MemoDroid	**0.30**	**0.47**	**0.53**	**0.61**	**0.64**	**5**

Memory across platforms



Chen, M., Liu, Z., Chen, C., Wang, J., Wu, B., Hu, J., & Wang, Q. (2025). Beyond Static GUI Agent: Evolving LLM-based GUI Testing via Dynamic Memory. *ASE'25,*

Agents with memory for software engineering

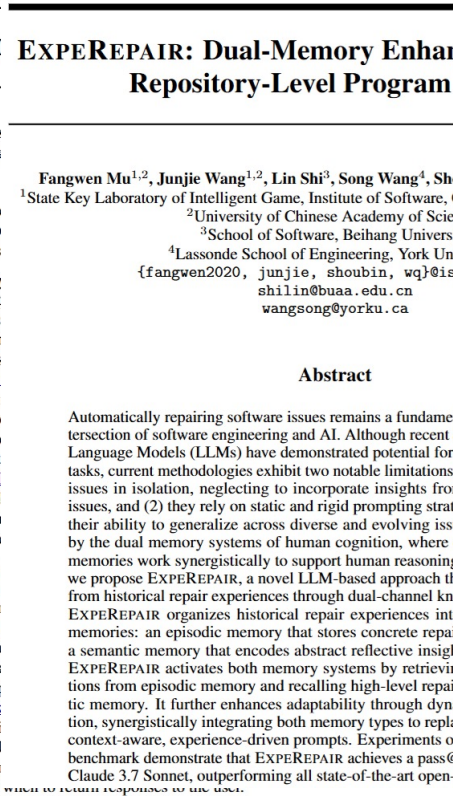


MemGPT: Towards LLMs as (

Charles Packer [1] Sarah Wooders
Vivian Fang [1] Shishir G. Patil [1] Ion Stoica

Abstract

Large language models (LLMs) have revolutionized AI, but are constrained by limited context windows, hindering their utility in tasks like extended conversations and document analysis. To enable using context beyond limited context windows, we propose *virtual context management*, a technique drawing inspiration from hierarchical memory systems in traditional operating systems which provide the illusion of an extended virtual memory via paging between physical memory and disk. Using this technique, we introduce MemGPT (MemoryGPT), a system that intelligently manages different storage tiers in order to effectively provide extended context within the LLM's limited context window. We evaluate our OS-inspired design in two domains where the limited context windows of modern LLMs severely handicaps their performance: document analysis, where MemGPT is able to analyze large documents that far exceed the underlying LLM's context window, and multi-session chat, where MemGPT can create conversational agents that remember, reflect, and evolve dynamically through long-term interactions with their users. We release MemGPT code and data for our experiments at https://research.memgpt.ai.

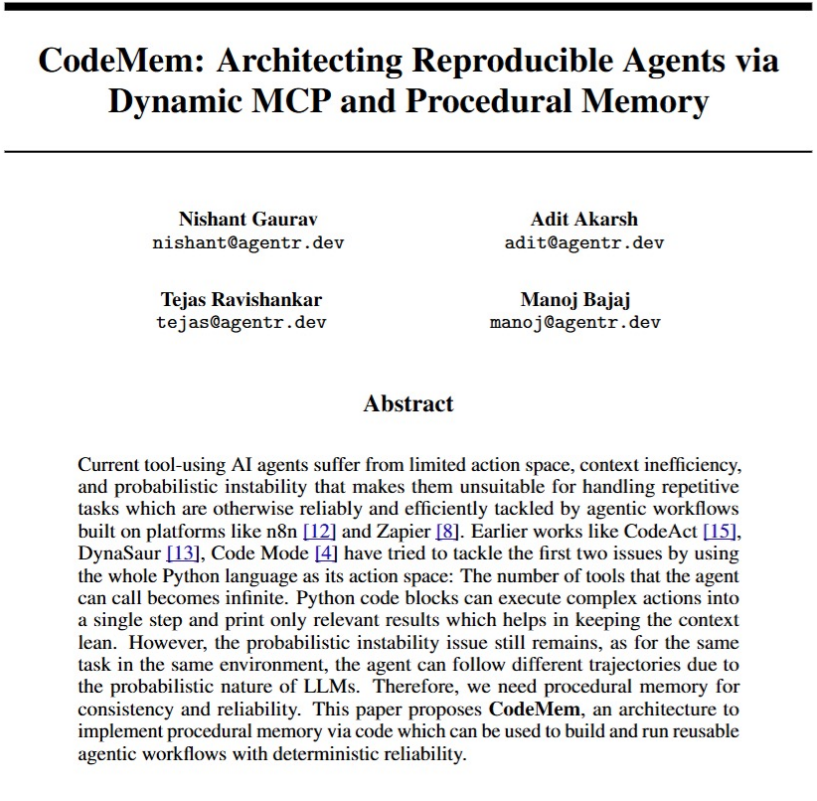


EXPEREPAIR: Dual-Memory Enhan
Repository-Level Program

Fangwen Mu[1,2], Junjie Wang[1,2], Lin Shi[3], Song Wang[4], Sh
[1]State Key Laboratory of Intelligent Game, Institute of Software,
[2]University of Chinese Academy of Scie
[3]School of Software, Beihang Univers
[4]Lassonde School of Engineering, York Un
{fangwen2020, junjie, shoubin, wq}@is
shilin@buaa.edu.cn
wangsong@yorku.ca

Abstract

Automatically repairing software issues remains a fundame
tersection of software engineering and AI. Although recent
Language Models (LLMs) have demonstrated potential for
tasks, current methodologies exhibit two notable limitations
issues in isolation, neglecting to incorporate insights fro
issues, and (2) they rely on static and rigid prompting strat
their ability to generalize across diverse and evolving iss
by the dual memory systems of human cognition, where
memories work synergistically to support human reasonin
we propose EXPEREPAIR, a novel LLM-based approach t
from historical repair experiences through dual-channel kn
EXPEREPAIR organizes historical repair experiences int
memories: an episodic memory that stores concrete repa
a semantic memory that encodes abstract reflective insigl
EXPEREPAIR activates both memory systems by retrievi
tions from episodic memory and recalling high-level repai
tic memory. It further enhances adaptability through dyn
tion, synergistically integrating both memory types to repl
context-aware, experience-driven prompts. Experiments o
benchmark demonstrate that EXPEREPAIR achieves a pass@
Claude 3.7 Sonnet, outperforming all state-of-the-art open-



CodeMem: Architecting Reproducible Agents via Dynamic MCP and Procedural Memory

Nishant Gaurav
nishant@agentr.dev

Adit Akarsh
adit@agentr.dev

Tejas Ravishankar
tejas@agentr.dev

Manoj Bajaj
manoj@agentr.dev

Abstract

Current tool-using AI agents suffer from limited action space, context inefficiency, and probabilistic instability that makes them unsuitable for handling repetitive tasks which are otherwise reliably and efficiently tackled by agentic workflows built on platforms like n8n [12] and Zapier [8]. Earlier works like CodeAct [15], DynaSaur [13], Code Mode [4] have tried to tackle the first two issues by using the whole Python language as its action space: The number of tools that the agent can call becomes infinite. Python code blocks can execute complex actions into a single step and print only relevant results which helps in keeping the context lean. However, the probabilistic instability issue still remains, as for the same task in the same environment, the agent can follow different trajectories due to the probabilistic nature of LLMs. Therefore, we need procedural memory for consistency and reliability. This paper proposes **CodeMem**, an architecture to implement procedural memory via code which can be used to build and run reusable agentic workflows with deterministic reliability.

3. Multi-user Interactive Feature

Multi-user interactive feature is a component of software that enables **multiple users** to **interact with each other** in **nearly real-time** within a software.

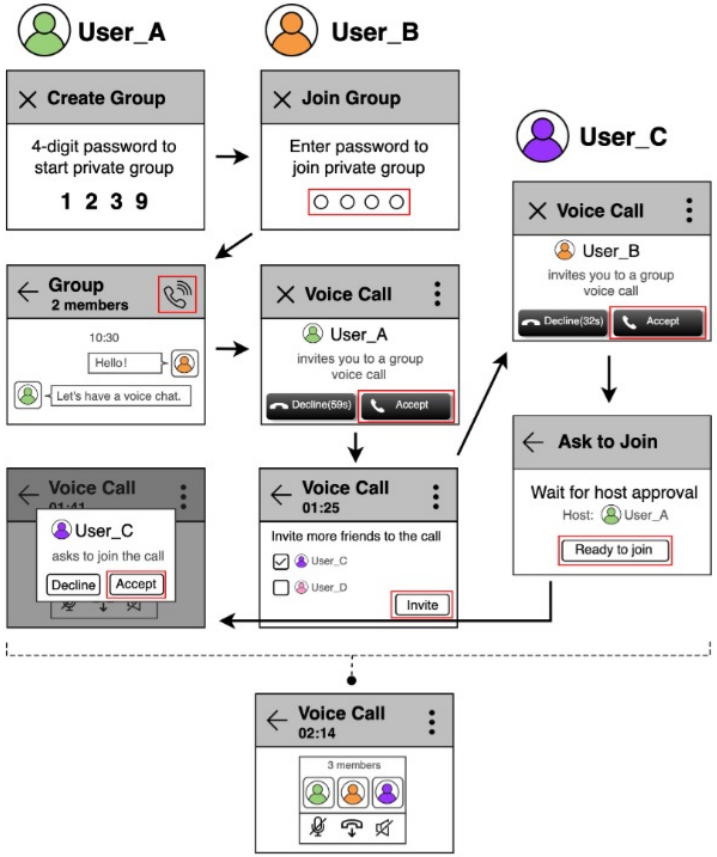


Figure 1: Illustration of a group voice call.

Motivation

40.31% (3,172 out of 7,870) of the feature testing tasks in TikTok involve multi-user interactive features.

Writing **record-and-replay** script for testing multi-user interactive feature is challenging.

- Device independent
- Action coordination

Task ID	$User_1$	$User_2$	$User_3$	#Actions
1	invite $User_2$ to a multi-guest LIVE session	accept the invitation	-	4
2	send $User_2$ an interactive card in LIVE session	trigger the interactive card	-	4
3	send a comment in LIVE session	reply the comment	-	4
4	create a face-to-face group chat	join the chat	join the chat	24
5	invite $User_2$ to join group chat	accept the invitation	-	4
6	send a real-time session invitation in group chat	enter the session via join card	accept the invitation	7
7	send a video call in group chat	accept the call	accept the call	6
8	send a animation in group chat	resume the animation	-	4
9	transfer group chat to $User_2$	confirm the group transfer	accept the transfer	9
10	invite $User_2$ with an animated link and $User_3$ without	join the animated link	join the link	8
11	send a video call to $User_2$ by Tool Bar	accept the call	-	4
12				2
13	send			4
14				6
15				7
16				6
17				7
18				5
19				3
20				4
21				8
22				8
23				14
24				9

TABLE I: Multi ... mpletion.

```python
import uiautomator2 as u2

# Device and App configurations
device_1 = u2.connect("{Xiaomi_Mix2S}")
device_2 = u2.connect("{Google_Pixel4a}")
...

# Device 1
action_1 = device_1(className=BUTTON, text="LIVE together").click()

action_2 = device_1(className=TEXTVIEW, text="{user_2}")\
        .parent()\
        .child_selector(text="Invite")
1
while not action_2.exists:
    device_1(className=LISTVIEW, resourceId="user_list")
        .scroll(direction="down")
    action_2 = device_1(className=TEXTVIEW, text="{user_2}")\
        .parent()\
        .child_selector(text="Invite")
    wait()
action_2.click()

# Device 2
action_3 = device_2(className=TEXTVIEW, text="{user_2} invites you to join LIVE")\
        .parent()\
        .child_selector(text="Accept")
action_3.click()
# more actions...
2
# Device 1
action_6 = device_1(className=TEXTVIEW, text="{user_2} requests permission")\
        .parent()\
        .child_selector(text="grant permissions")
action_6.click()
```

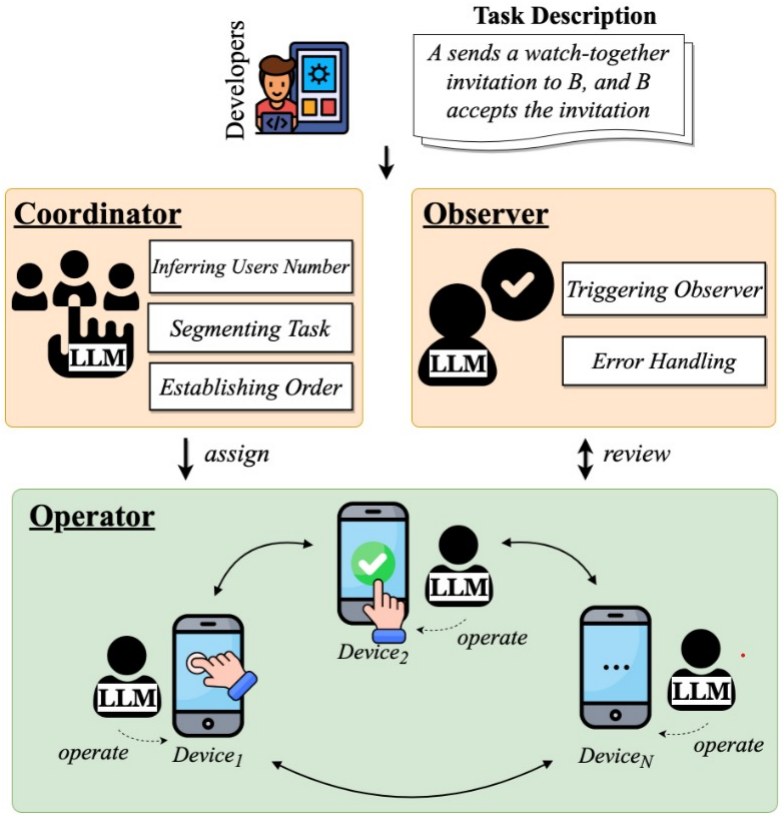
device agnostic

action coordination

Fig. 2: Example of the script for multi-user feature testing.

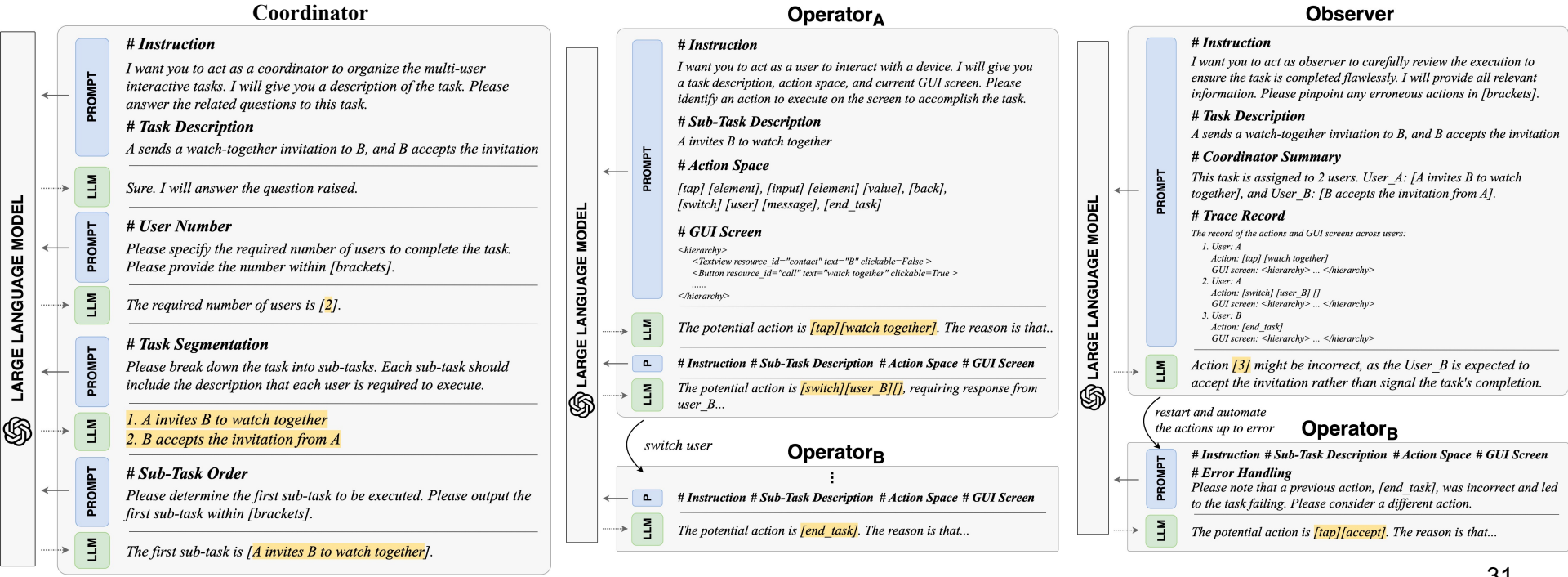
Approach

A **multi-agent** framework powered by the LLMs to emulate users interacting with GUI screens, collaboratively achieving GUI automation of multi-user interactive tasks.



S Feng, C Du, H Liu, Q Wang, Z Lv, G Huo, X Yang, C Chen. Breaking Single-Tester Limits: Multi-Agent LLMs for Multi-User Feature Testing. *ICSE'26, arXiv:2506.17539*

Device Allocation from Farm



Evaluation

Comparison with the State-of-the-Art

- 41 multi-user interactive tasks from 16 apps collected manually
- An average of 2.87 actions per device to complete the task

Method	Success Rate	Action Similarity
Monkey [14]	7.3%	-
Sapienz [52]	9.6%	-
Stoat [56]	12.2%	-
Humanoid [47]	9.6%	-
AdbGPT [30]	53.7%	79.1%
AutoDroid [66]	63.4%	85.4%
AppAgent [73]	60.9%	86.8%
Mobile-agent [63]	68.3%	89.6%
MAdroid	82.9%	96.8%

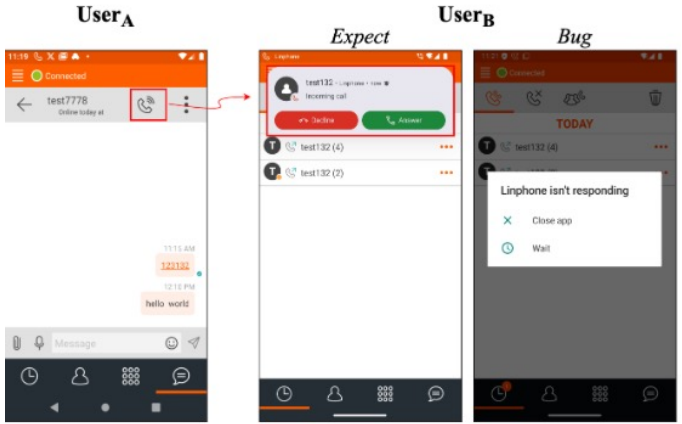


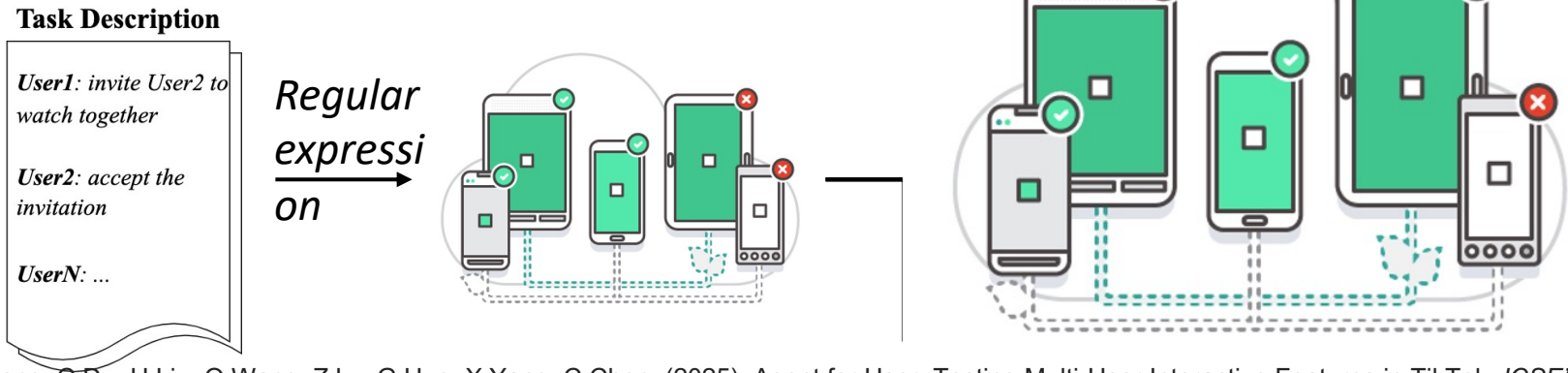
Figure 9: Illustration of interactive bug.

S Feng, C Du, H Liu, Q Wang, Z Lv, G Huo, X Yang, C Chen. (2026). Breaking Single-Tester Limits: Multi-Agent LLMs for Multi-User Feature Testing. *ICSE'26, arXiv:2506.17539*

Application to real-world industry

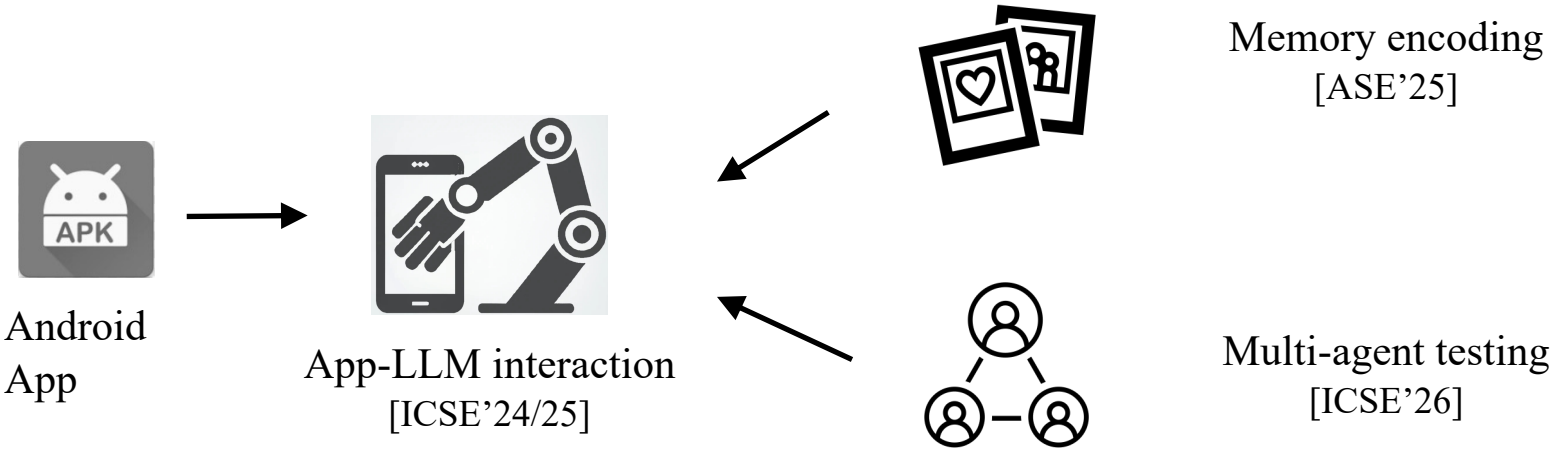
We build a virtual device farm for this feature testing

- 5,918 virtual devices using Android emulators distributed across 395 ARM-based commodity servers
- CPU, Storage, Bandwidth, etc.
- Smartphone vendors and services



S Feng, C Du, H Liu, Q Wang, Z Lv, G Huo, X Yang, C Chen. (2025). Agent for User: Testing Multi-User Interactive Features in TikTok. *ICSE'25, arXiv:2504.15474*

LLM-driven Automated Human-like App Testing



Limitations of Human-like GUI Testing

Scalability
& efficiency

Visual perception
of Multimodal-LLM

Oracle
missing

Testing
thoroughness

Beyond software testing: personalized AI