Rethinking OS Interfaces for LLM Agents

Yuan Wang[*◊†] , Mingyu Li[*◊] , Haibo Chen[*◊‡]
[*]Key Laboratory of System Software (Chinese Academy of Sciences)
[◊]Institute of Software, Chinese Academy of Sciences
[‡]Shanghai Jiao Tong University
[†]University of Chinese Academy of Sciences

This is a condensed version of “From Imperative to Declarative: Towards LLM-friendly OS Interfaces for Boosted Computer-Use Agents,” to appear in EuroSys 2026.

Computer-Use Agent (CUA)

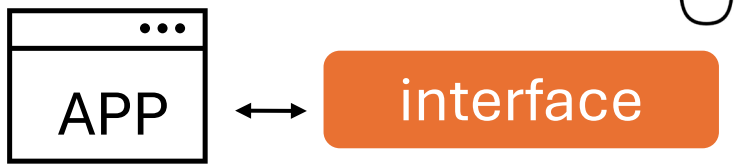
Vision:
complete tasks automatically
free human labor

OpenAI ANTHROP\C UI-TARS manus

OpenClaw

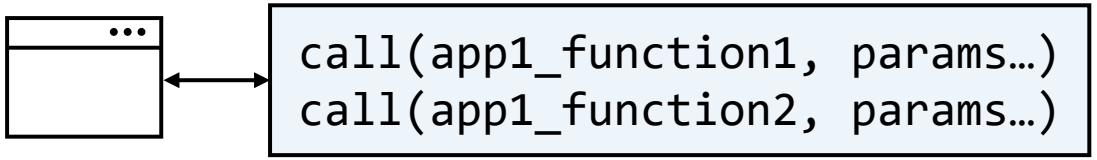
Computer-Use Agent (CUA)

require interfaces to interact with applications



API

reliable and efficient



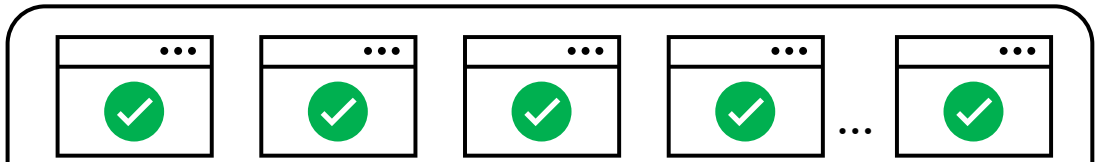
limited generality

We **cannot** use most applications through APIs

most apps w/o public APIs

GUI

dominant and universal



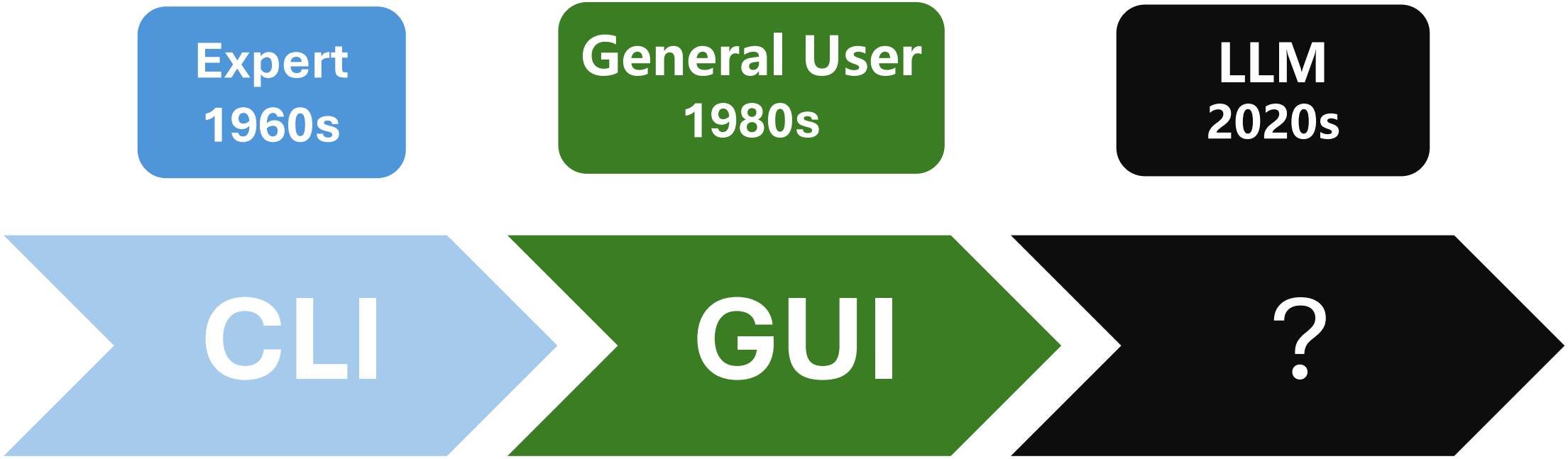
most apps w/ GUIs

long sequence of actions

visual perception

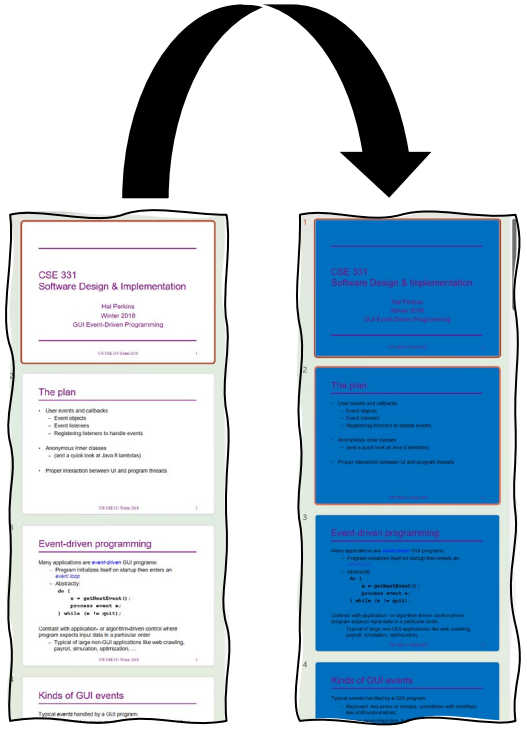
observe and **do action** 1 -> **observe** and **do action** 2 ->
observe and **do action** 3 -> **observe** and **do action** 4 ->
.....

What is a good OS interface for LLMs?

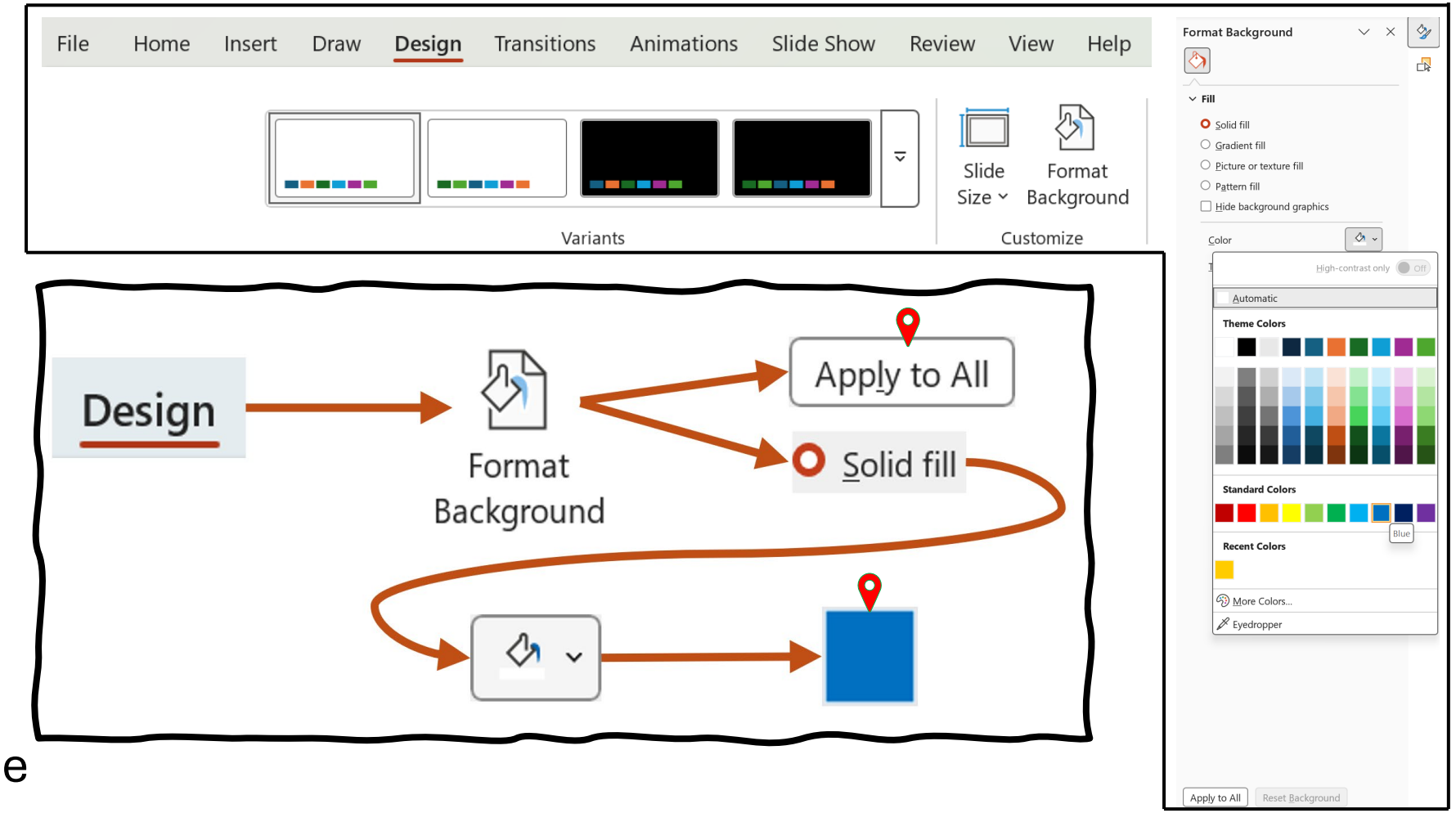


Imperative GUI design： Navigation

To navigate through menus, dropdowns, dialogs to make target control visible



make the background blue on all slides.

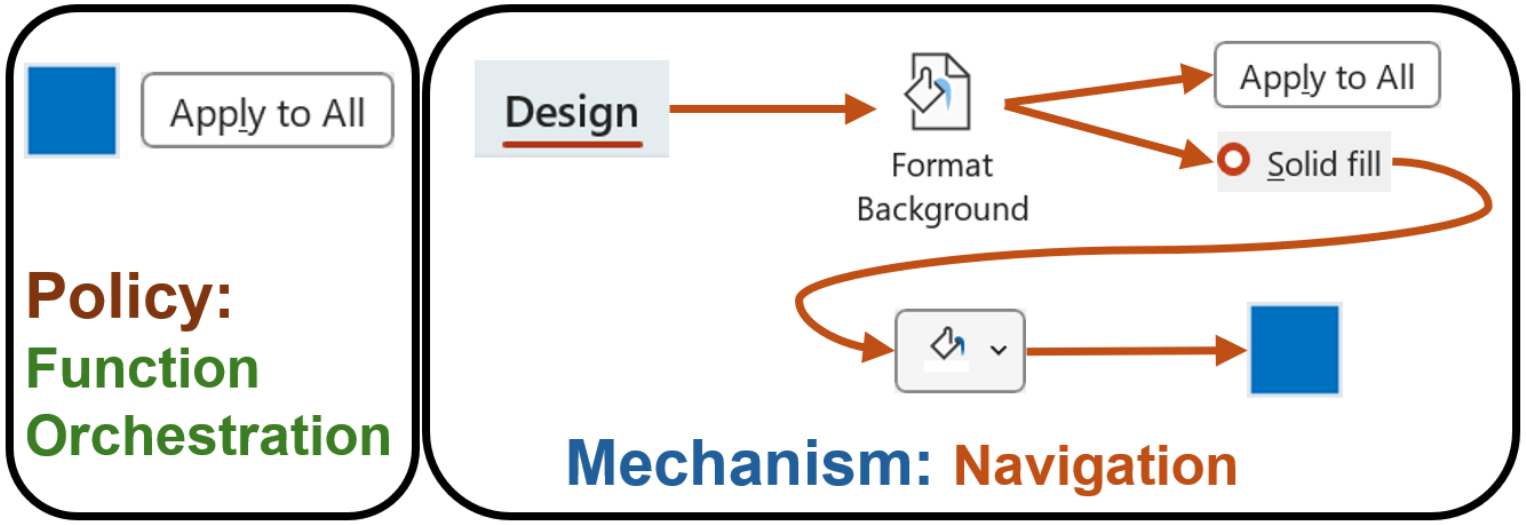


Imperative GUI design： Interaction

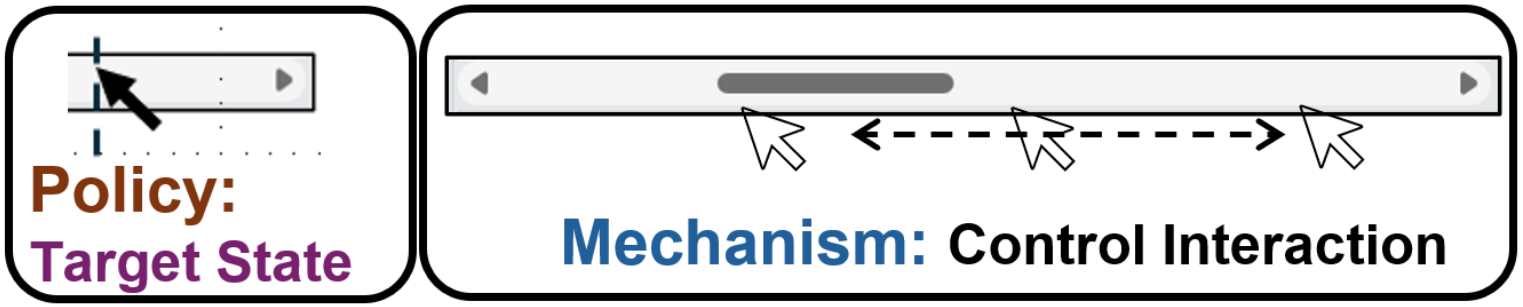
To trigger a function

a simple click, and complex interactions

Imperative GUI problem: Policy-Mechanism Coupling



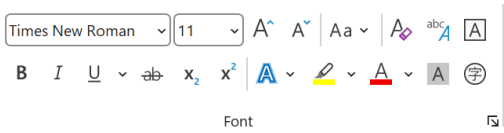
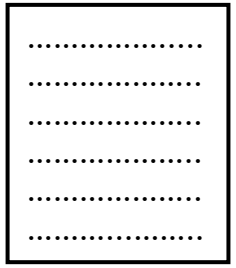
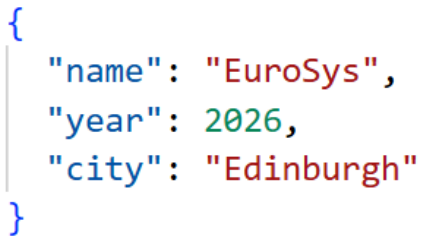
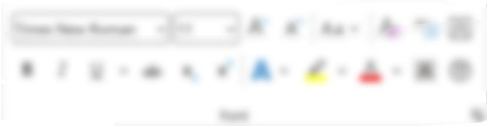
make the background blue on all slides.



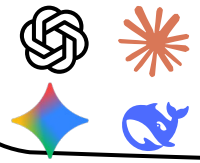
show the area close to the end.

Mismatch #1: Procedural *navigation*

hierarchical **navigation**, **limited** set of choices for each step

Memory Size	Syntax Recall	Visual Recognition
... 5 – 20 options	command names? structured output? programming?	
 1M tokens		

LLM



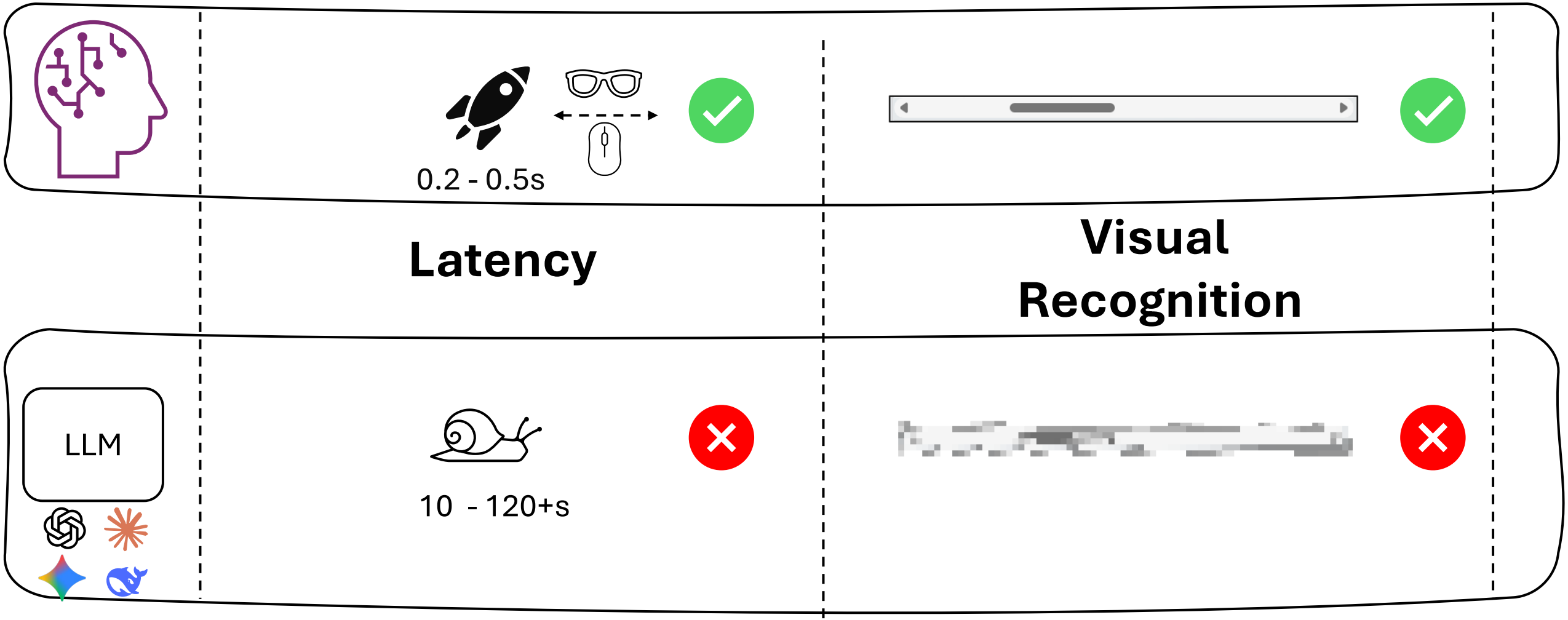
Mismatch #1: Procedural *navigation*

LLM's view:

GUI navigation requires long, fine-grained sequence of actions.
Success Rate: 0.7 x 0.7 x 0.7 x 0.7 x 0.7 x 0.7 x … 😭

Mismatch #2: Iterative *interaction*

“observe-act” loop to interact with UI controls



Mismatch #2: Iterative *interaction*

LLM's view:

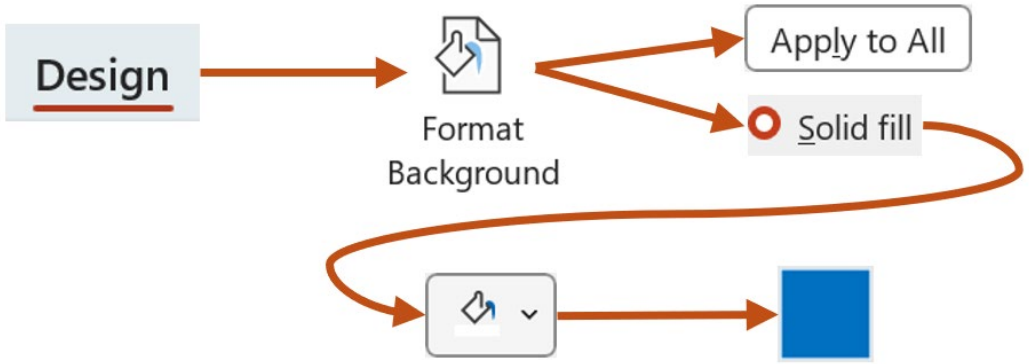
GUI interaction incurs substantial latency/token cost overhead & low success rate. 😭

How should OSes evolve interfaces for LLM agents?

Insight

much of the mechanism (navigation & interaction) is deterministic and can be resolved algorithmically without LLM involvement

1. Deterministic navigation

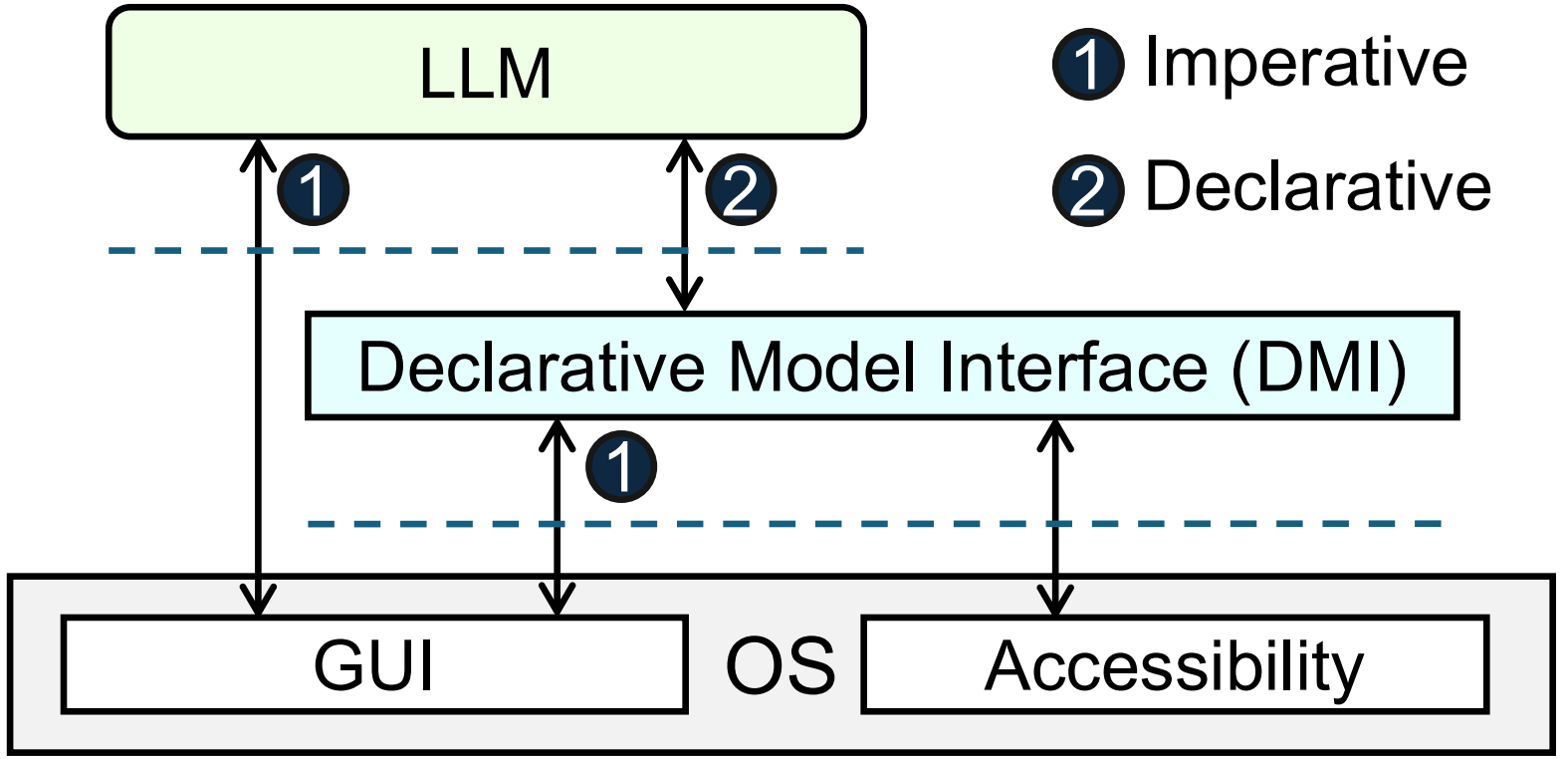


2. Finite interaction operations (UI Automation)

- 41 control types (e.g., Button, ListItem, Edit)
- 34 control patterns (e.g., TextPattern, ScrollPattern, SelectionPattern)

Solution : abstraction

Policy-Mechanism decoupling



Instead of forcing the LLM to operate the GUI imperatively, we introduce a higher-level abstraction that separates **policy** from **mechanism**.

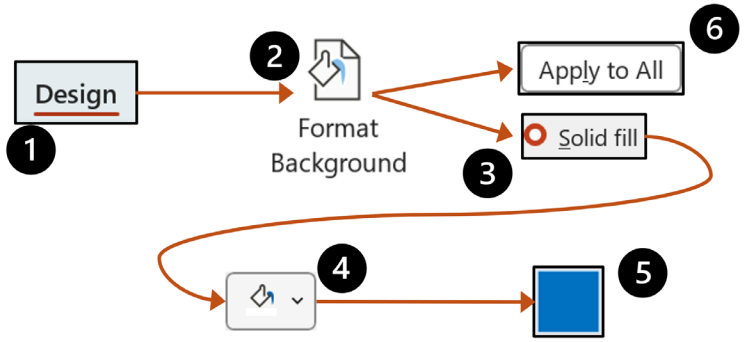
Primitive: Access

abstract away **navigation**

access | *visit(["leaf_node_id1",...)] ...*

make the background blue on all slides.

Task	GUI	DMI
1	click("Design") → click("Format Background") → click("Solid fill") → click("Fill Color") → click("Blue") → click("Apply to All")	visit(["Blue", "Apply to All"])



Just declare what to do:

Use "Blue", "Apply to All"

Primitive: State & Observation

abstract away **interaction**: **stateful representation**

`state` *set_scrollbar_pos(); select_controls(); select_lines(); …*

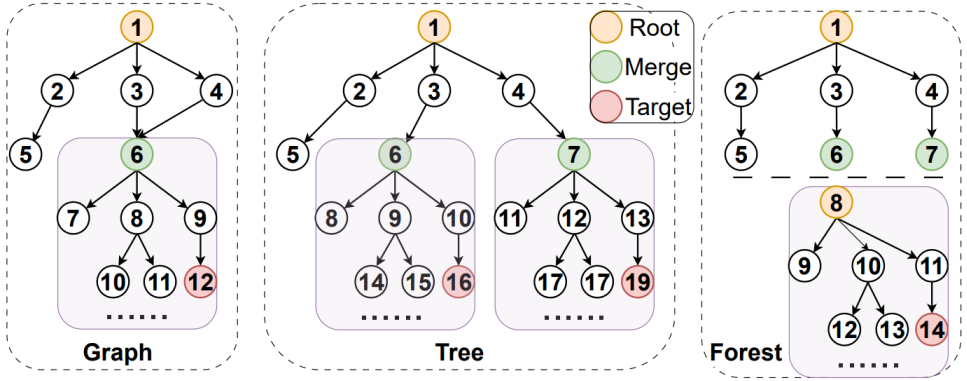
`observation` *get_texts(); …*

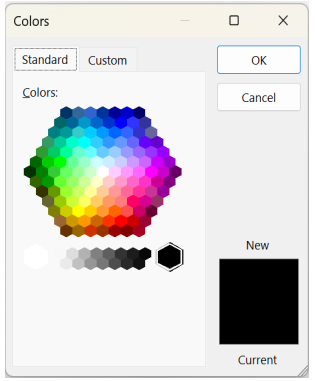
state declaration and observation declaration interfaces. UIA defines **34 control patterns** in total. These interfaces are extensible. For example, set_texts builds on TextPattern, set_toggle_state builds on TogglePattern, and set_expanded/set_collapsed builds on ExpandCollapsePattern.

Interface	Control Pattern	Description
set_scrollbar_pos	Scroll	Set scrollbar position to x%
select_lines	Text	Select one or contiguous lines
select_paragraphs	Text	Select one paragraph or a contiguous paragraph range
select_controls	Select	Support single or multiple selection
get_texts	Text & Value	Retrieve a control's text

Challenge

- navigation path ambiguity
- Limited LLM context windows
- Inaccurate long-horizon planning





Solution

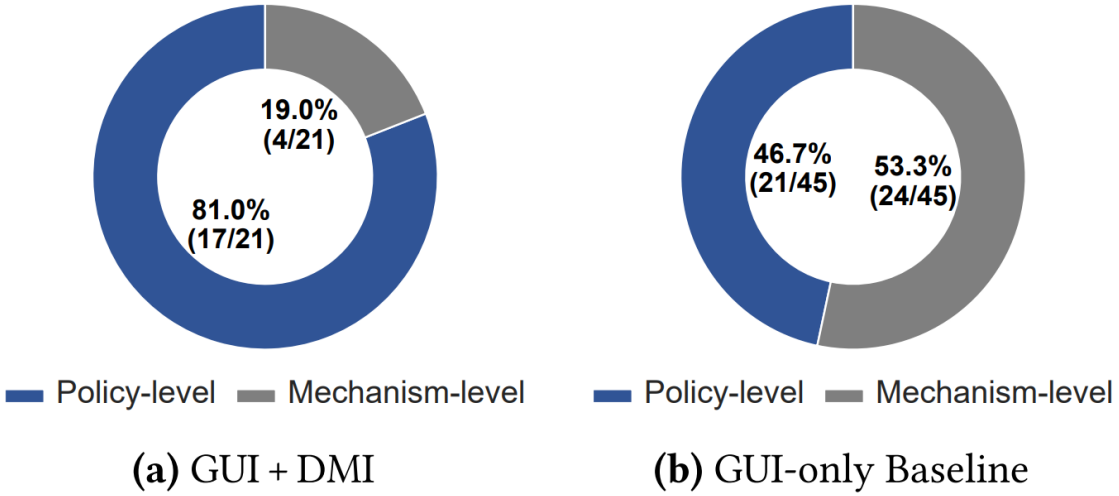
- path-unambiguous topology
- forest (a main tree and subtrees)
- query on-demand
- filtering, fuzzy match, fast-path/slow-path

Evaluation

Improvements over several model settings and tasks.

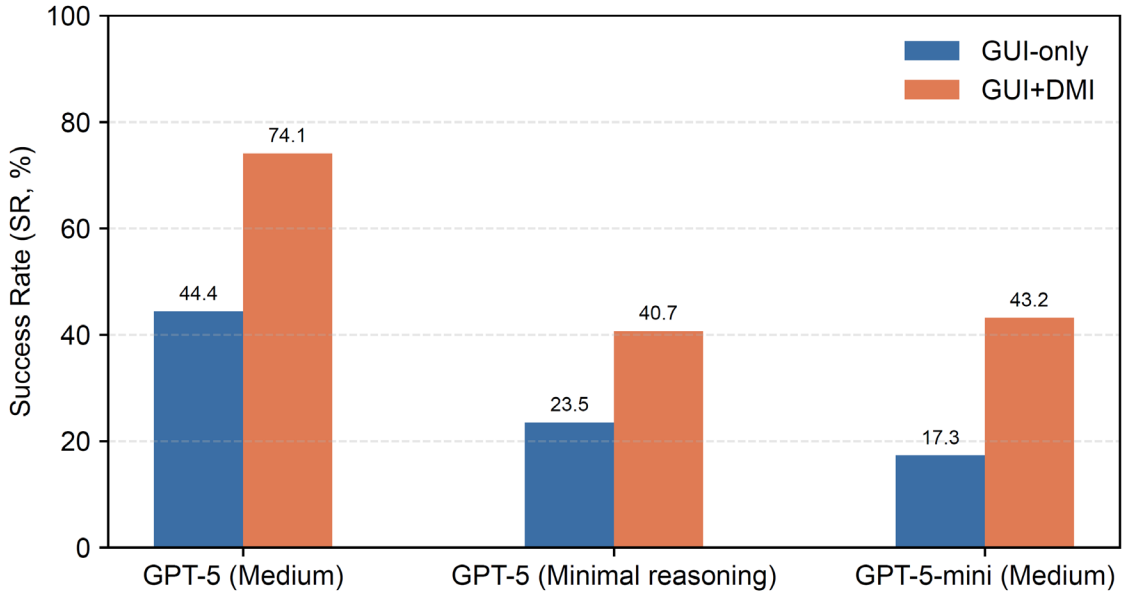
1. success rate: 44.4% to 74.1%.

2. 61% successful cases: only **one** LLM call.

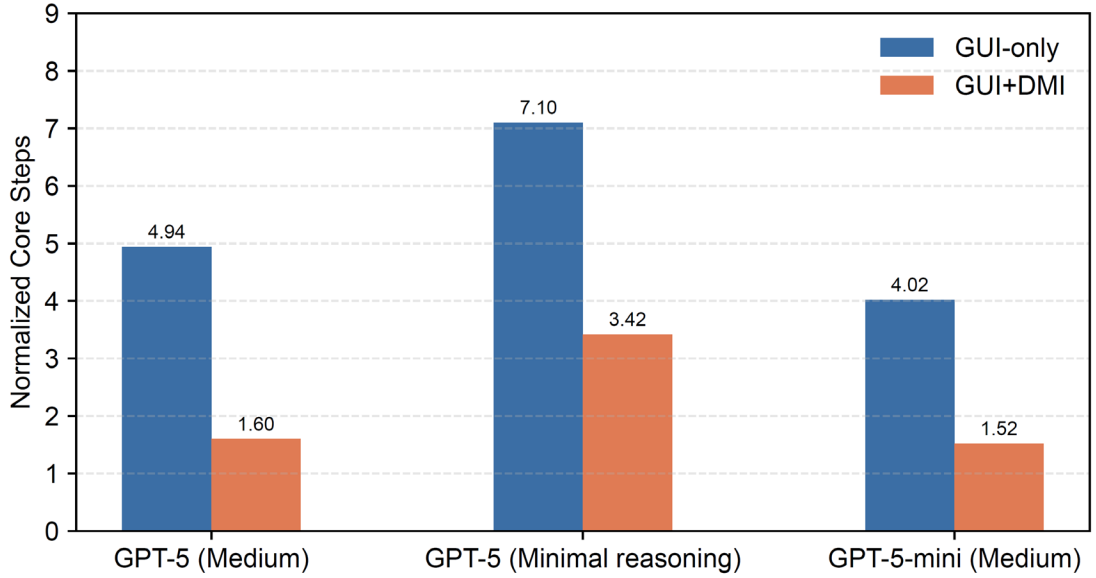


(a) GUI + DMI **(b)** GUI-only Baseline

DMI cuts out mechanism-related errors (navigation and composite interaction) and re-centers policy



(a) Success Rate



(b) Normalized Core steps

Benchmark: OSWorld-W(windows), Word, Excel, PowerPoint

Summary

We introduce DMI, an abstraction that decouples policy from mechanism by reducing complex GUI operations to declarative primitives.

DMI does **not** require modifying or accessing the application source code or relying on application programming interfaces (APIs).

DMI enables API-like declarative interactions for GUI applications.

Thanks!