GUIFuzz++: Unleashing Grey-box Fuzzing on Desktop Graphical User Interfacing Applications

Dillon Otto, Tanner Rowlett, Stefan Nagy
University of Utah, USA

Introduction & Overview

Software fuzzers are uniquely engineered to target specific software **interfaces** (e.g., on-disk files, in-memory buffers or environment-level resources). Yet, **one major interface remains universally under-tested** across today's ever-growing desktop software ecosystems: **the Graphical User Interface (GUI).**

We thus introduce **GUIFuzz++:** the world's first general-purpose fuzzer for desktop GUI applications. **Through the following enhancements,** GUIFuzz++ enables GUI-agnostic fuzzers like AFL++ to drive random, diverse GUI interactions that unveil complex, **GUI-induced software defects**:

- **GUI Interaction Interpreter:** a middle-layer for translating fuzzer-generated random bytes into distinct GUI events.
- **Window Interference Handling:** tracking and preventing unwanted window interference (e.g., update dialogues, third-party pop-ups) that otherwise derail GUI fuzzing.
- **Higher-precision GUI Interaction:** leveraging the AT-SPI accessibility interface to better pinpoint GUI elements.

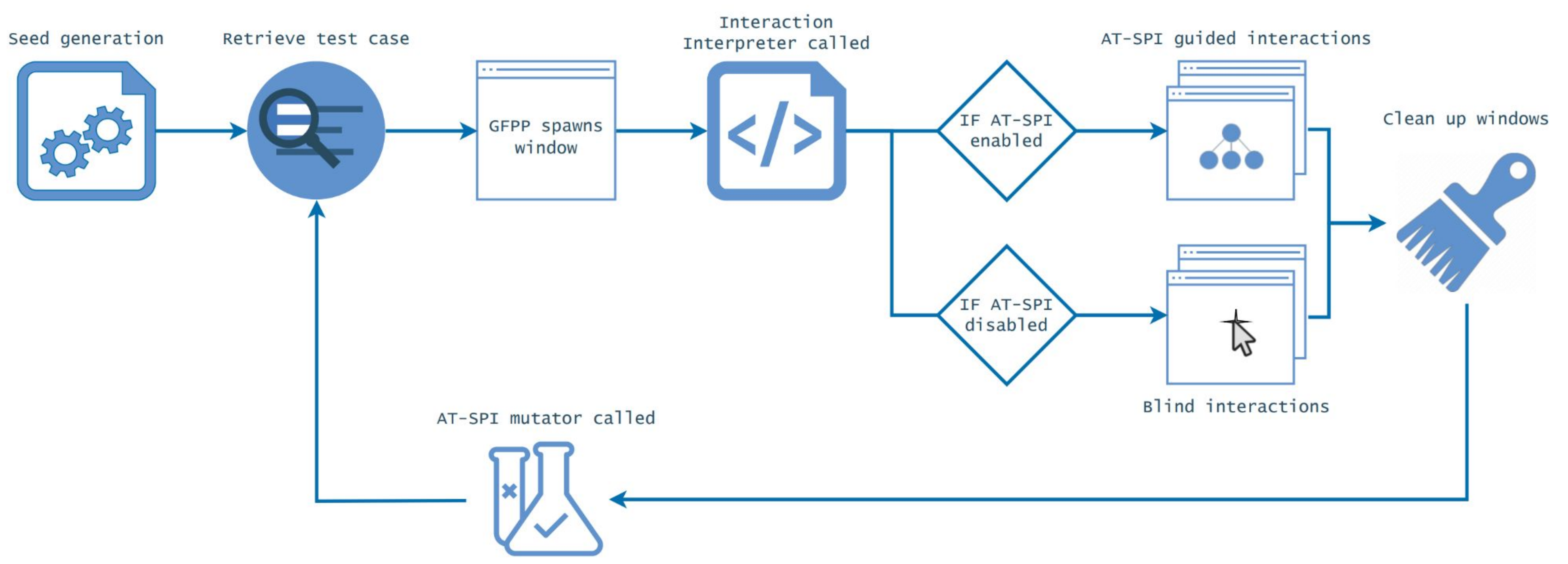
Evaluated across **12** popular real-world Linux GUI applications, GUIFuzz++ finds **23 previously-unknown GUI-induced bugs**.

Step-by-Step Visualization of GUIFuzz++

Overview: GUIFuzz++'s implementation **spans just minimal changes to the underlying GUI-agnostic fuzzer** of choice (e.g., AFL++).

- Following test case generation, GUIFuzz++ parses bytes into distinct GUI interactions and dispatches them accordingly.
- All of the fuzzer's core mechanisms (code coverage retrieval, crash detection, etc.) **remain completely unchanged**.

In total, our modifications to AFL++ encompassed **just eight lines of code**—making GUIFuzz++ easily added to most fuzzers today.



The GUI Interaction Interpreter

Key idea: translate fuzzers' randomly-generated inputs into actionable GUI events, **each following a three-byte structure:**

- **Op 1:** interaction **opcode** (e.g., closes, keypress, click, drag).
- **Ops 2-3:** interaction **semantics** (e.g., the pixel coords to click).

Outcome: easily reshape GUI-agnostic fuzzers **into capable GUI fuzzing tools**—with minimal changes to their core workflow.

Op Structure	Description of GUI Interaction
`00 FF FF`	**Close** currently-active window, ignoring the last two operands.
`01 CC FF`	**Input** the key press corresponding to the extended `ASCII` encoding of primary operand `CC`, ignoring the second operand. *Ex:* `01 7F FF` → input extended `ASCII` key press "`DEL`".
`02 XX YY`	**Click** the location (`X%`, `Y%`) relative to the current window's dimensions, offset from its bottom-left coordinate (`0,0`). *Ex:* `02 A0 1B` → click relative position (`62.5%`, `10.5%`).
`03 XX YY`	**Drag** the cursor from its *current* position to the *new* position (`X%`, `Y%`) relative to the current window's dimensions, offset from its bottom-left coordinate (`0,0`). *Ex:* `03 00 80` → drag to relative position (`0%`, `50%`).
`NN AA BB`	**All higher opcodes** (i.e., `04–FF`): normalize the opcode via (`NN % 4`), reinterpreting the transformed opcode accordingly. *Ex:* `B2 2C 9F` → reinterpret as *click* operation `02 2C 9F`.

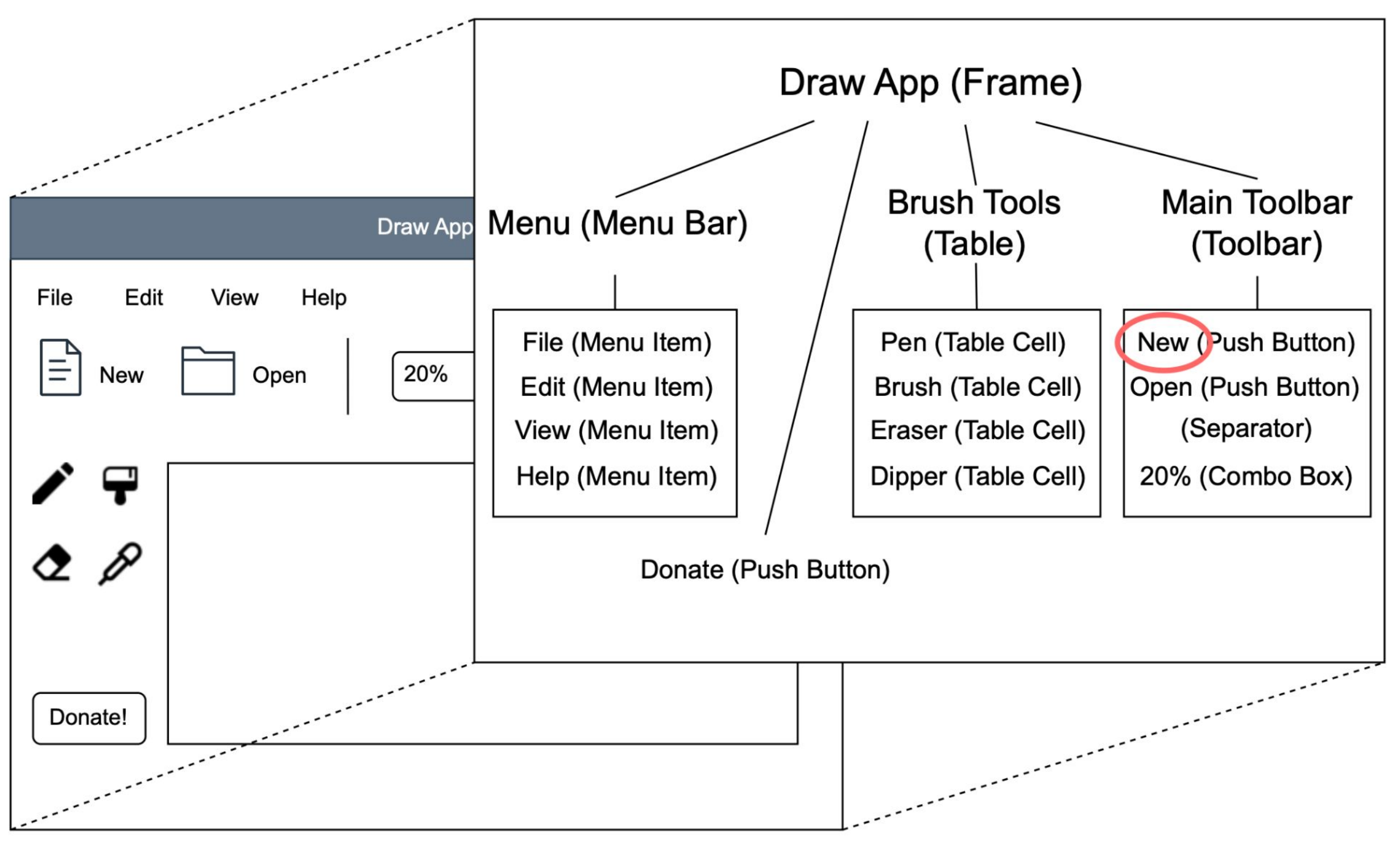
Attaining Higher-precision GUI Fuzzing via AT-SPI

Category	Op Structure	Element Type	Visual Example
General	`04 AA BB`	Pushable Button	Submit
	`05 AA BB`	Text Entry Field	Find ...
Toggleables	`06 AA BB`	Checkbox Button	✓ Apply?
	`07 AA BB`	On/Off Button	On Off
Selections	`08 AA BB`	Radio Button	● Use Option 1 ○ Use Option 2
	`09 AA BB`	Spinner Button	Width: 4px + −
	`10 AA BB`	Table Cell Button	+/− % ÷
	`11 AA BB`	Drop-down Item	Option 1 ▼ Option 2 Option 3
	`12 AA BB`	Combination Box	*Search: "width"* ▼ Option 1: Height Option 2: Width
Movable	`13 AA BB`	Scrollable Field	Field 2: value 2 Field 3: value 3 Field 4: value 4
	`14 AA BB`	Sliding Selection	0 1 2 3 4 5 6 7

Key idea: extend GUI Interaction Interpreter with the ability to pinpoint **distinct GUI elements**, enabling higher-precision fuzzing:

- **Leverage AT-SPI accessibility API** to parse the GUI tree on-the-fly.
- **Bucket GUI elements by type,** and randomly select which one to interact with during each dispatched GUI interaction.

Outcome: enhance GUI fuzzing's potential with **much deeper GUI exploration** compared to random pixel-coordinate clicking alone.



Evaluating GUIFuzz++

Evaluation: we evaluate GUIFuzz++ on **12 real-world Linux desktop GUI applications**, spanning three GUI frameworks.

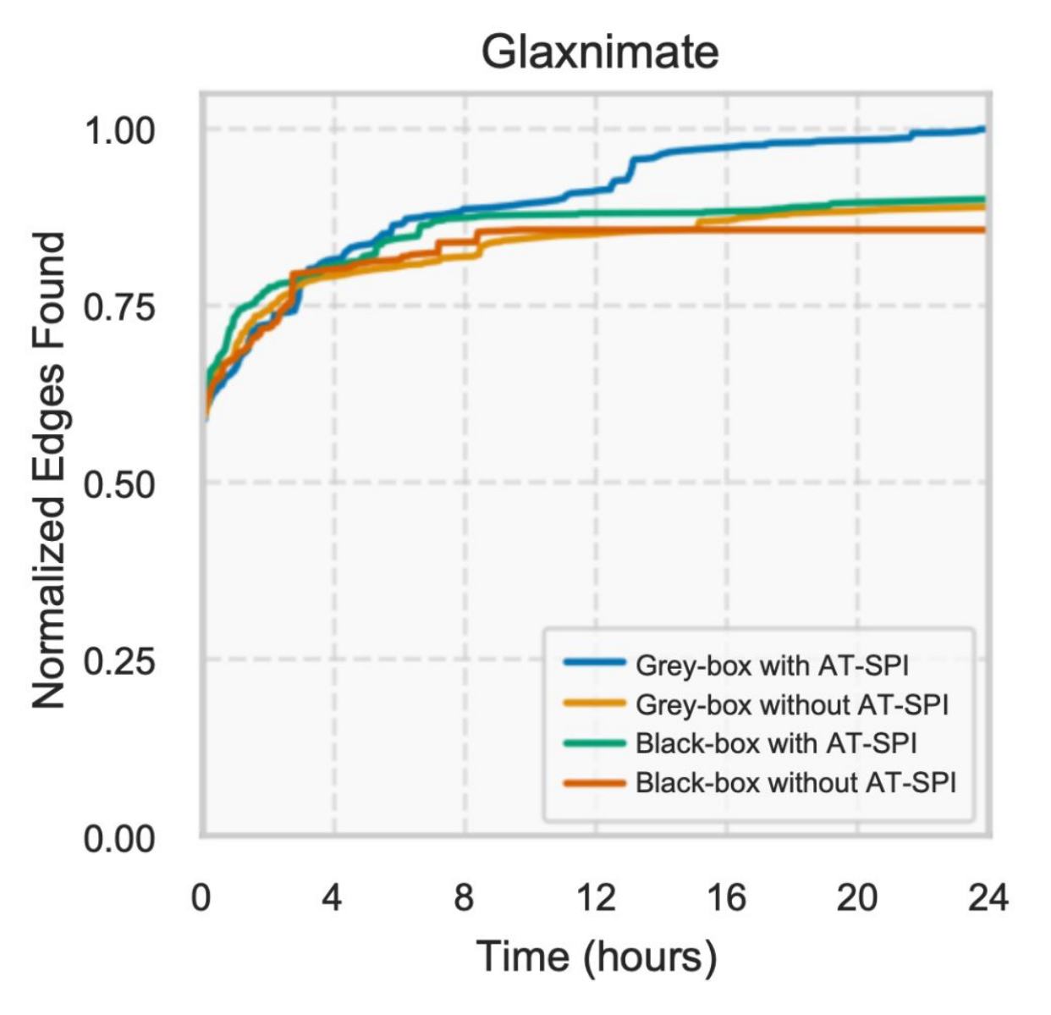
As no other comparable GUI fuzzers exist, we perform an **ablation study of GUIFuzz++'s four supported fuzzing modes**:

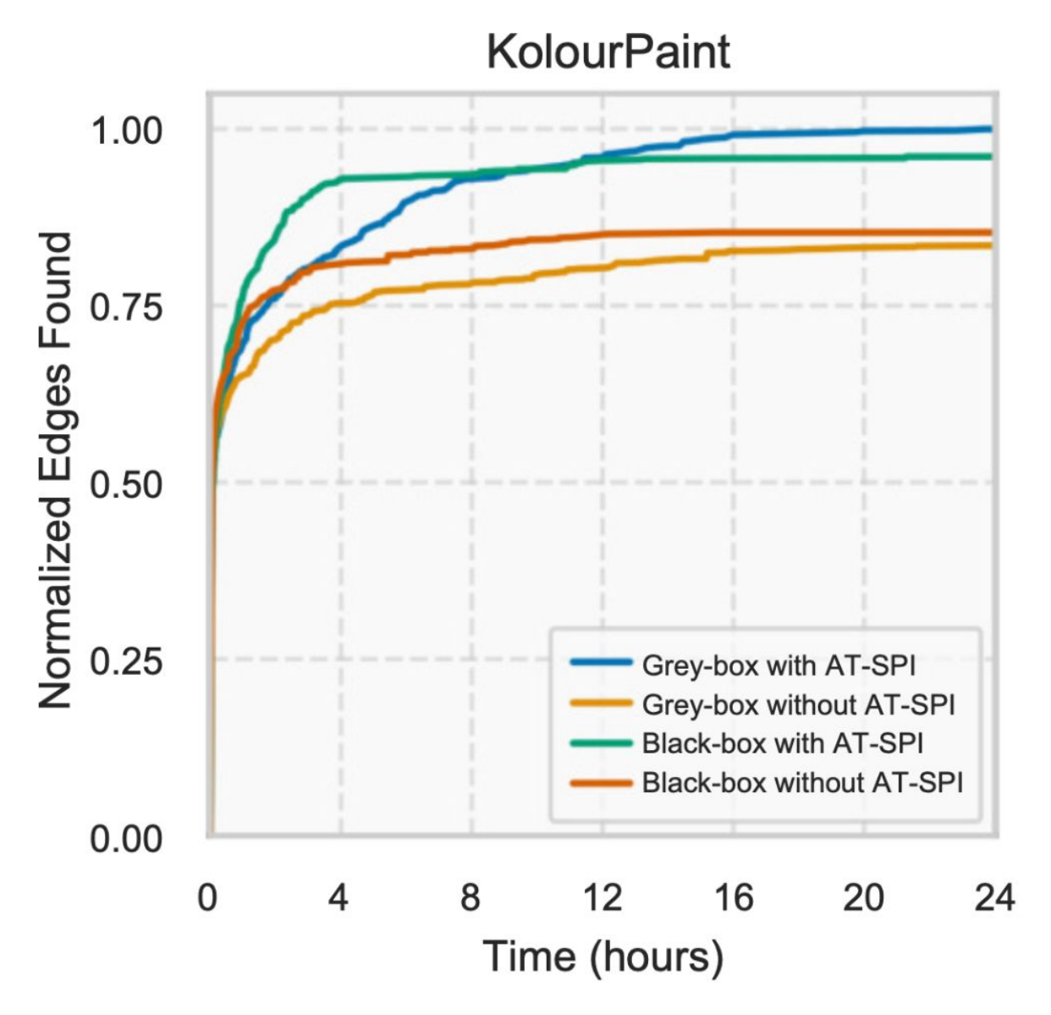
- (**1**) Grey-box fuzzing with and (**2**) without AT-SPI, and
- (**3**) Black-box fuzzing with and (**4**) without AT-SPI.

All experiments span five 24-hour fuzzing trials per benchmark, measuring **code coverage**, **throughput**, and **GUI bugs found**.

- *For brevity, our full experimental data is shown in the paper.*

Program	Description	Base GUI	Program	Description	Base GUI
Dia [27]	Graphic Design	GTK	MATE-calc [38]	Calculator	GTK
Glaxnimate [3]	Animation	Qt	PlotJuggler [8]	Data Plotting	Qt
KCalc [29]	Calculator	Qt	QCAD [41]	3-D Modeling	Qt
KolourPaint [30]	Image Editor	Qt	Skrooge [32]	Finance	Qt
LabPlot [31]	Data Plotting	Qt	Umbrello [33]	UML Editor	Qt
LibreCAD [36]	3-D Modeling	Qt	XCalc [52]	Calculator	Xorg





GUI Bug Discovery

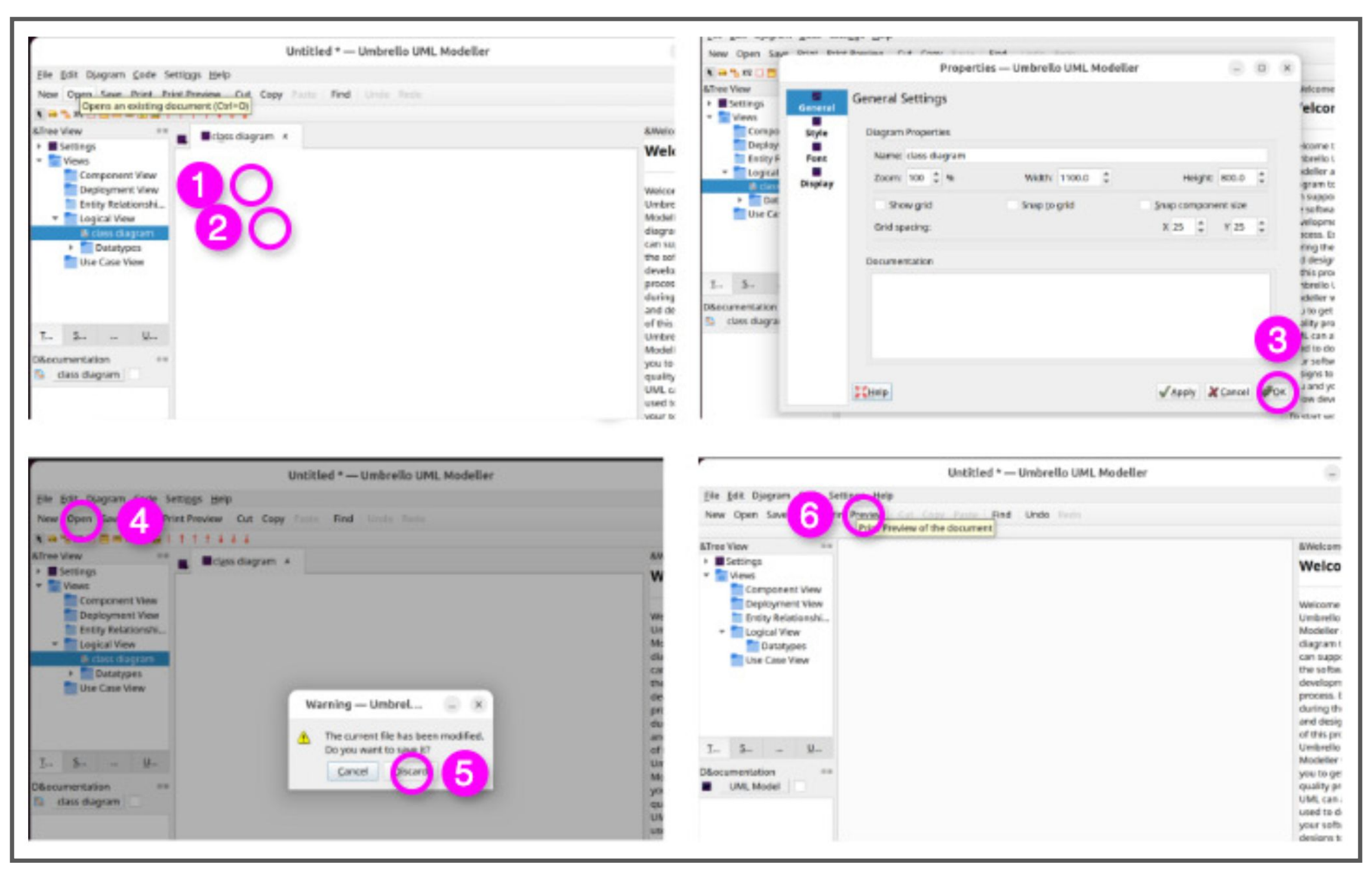
(🔒 = fixed by developers, 👍 = confirmed and waiting fixing, ⌛ = pending).

ID	Program	Bug Type	Brief Desc.	New	Status
01	Dia	Bad Free	Color area (transient)	✓	🔒
02	Glaxnimate	Segfault	Improper closing	✓	⌛
03	Glaxnimate	Segfault	Invalid cut/pastes	✓	⌛
04	KCalc	Invalid Ptr	Inserting open parent	✓	👍
05	KCalc	Segfault	Left bit shift overflow		
06	KolourPaint	Heap UAF	Specific tools with undo	✓	👍
07	KolourPaint	Segfault	Buggy bug report menu	✓	⌛
08	KolourPaint	Segfault	Shortcut settings dropdowns	✓	⌛
09	KolourPaint	Segfault	Print preview zooming	✓	⌛
10	LabPlot	Invalid Ptr	Invalid column insert	✓	🔒
11	LabPlot	Heap UAF	Pinning spreadsheets	✓	🔒
12	LabPlot	Heap UAF	Pinning matrices	✓	🔒
13	LibreCAD	Heap UAF	Invalid plugin usage	✓	🔒
14	LibreCAD	Heap UAF	Consecutive points	✓	🔒
15	MATE-calc	Bad Free	Invalid square roots		
16	MATE-calc	Bad Free	Empty inverse trig functions	✓	🔒
17	PlotJuggler	Segfault	Quickly close button docker	✓	⌛
18	QCAD	Segfault	Tool use in multiple sheets	✓	⌛
19	Umbrello	Segfault	Birds eye after discard	✓	🔒
20	Umbrello	Heap UAF	Multiple sequence diagrams	✓	🔒
21	Umbrello	Heap UAF	Undo after discard	✓	🔒
22	Umbrello	Segfault	Print Preview after discard	✓	🔒
23	Umbrello	Segfault	Cut on empty diagram	✓	🔒
24	XCalc	FPE	Invalid modulus	✓	⌛
25	XCalc	FPE	Invalid modulus	✓	⌛

Results: GUIFuzz++ finds **23 previously-unknown GUI bugs**, of which **14 are thus far confirmed or fixed** by developers.

Many bugs spanned **complex GUI interaction sequences**, underscoring the power of GUIFuzz++'s fuzzing capabilities.

Conclusion: why GUIFuzz++?



In summary, GUIFuzz++'s unique contributions form **the first approach** to successfully extend general-purpose fuzzers to today's vast ecosystem of **desktop GUI software**:

- Systematically translating fuzzer-generated random byte sequences into actionable, logic-exercising GUI events.
- High-precision GUI fuzzing that avoids window hurdles.
- Discovered **23 previously-unknown GUI bugs** so far.

Try GUIFuzz++ out yourself!

- Full repository available [illegible]: github.com/FuturesLab/GUIFuzzPlusPlus.
- **Integration coming soon to AFL++!**

KAHLERT SCHOOL OF COMPUTING
THE UNIVERSITY OF UTAH

FUTURES³ LAB
FUTURE TECHNOLOGY FOR USABLE, RELIABLE, & EFFICIENT SECURITY OF SOFTWARE & SYSTEMS

U.S. National Science Foundation

DARPA