A Click Is Not a Click: UIA vs CDP Under Measurement

Exzil Calanza

Independent Researcher, Iloilo City, Philippines
Independent research preprint. IEEE-style formatting only; this work has not been accepted, published, or peer reviewed by IEEE.

ABSTRACT A click is not a click. When a computer-control agent reports clicked=true , that single boolean can hide several different propositions: it controlled the intended window, identified the intended element, used the correct coordinate space, delivered the intended input semantics, acted on an unobstructed target, and caused the application to accept the action. Any one of those can be wrong while an automation API itself reports success. This Skynet field study started with a practical engineering problem: trusted Windows UI Automation (UIA) actions were reliable but much slower than the normal browser-control path. The tempting conclusion was to replace UIA with Chrome DevTools Protocol (CDP) input. Instead, we instrumented the layers separately, increased the sample sizes after an independent review rejected the first draft, and allowed the experiments to falsify our own initial interpretation. Correctly instrumented CDP pointer input passed 50/50 on the tested control and produced trusted pointer/mouse events. DOM activation also passed 50/50 but with different event fidelity. An explicit coordinate transform matched the 96-DPI environment closely. Most importantly, the study corrected itself: an early benchmark interpretation was falsified by a better-controlled experiment, and the process uncovered the instrumentation bug that had inverted the result.

INDEX TERMS — application acceptance; browser automation; Chrome DevTools Protocol; coordinate systems; input fidelity; UI Automation; Windows automation

I. INTRODUCTION

A click is not a click.

When a computer-control agent reports `clicked=true`, that single boolean can hide several different propositions: it controlled the intended window, identified the intended element, used the correct coordinate space, delivered the intended input semantics, acted on an unobstructed target, and caused the application to accept the action. Any one of those can be wrong while an automation API itself reports success.

This Skynet field study started with a practical engineering problem: trusted Windows UI Automation (UIA) actions were reliable but much slower than the normal browser-control path. The tempting conclusion was to replace UIA with Chrome DevTools Protocol (CDP) input. Instead, we instrumented the layers separately, increased the sample sizes after an independent review rejected the first draft, and allowed the experiments to falsify our own initial interpretation.

The result is not “UIA wins” or “CDP wins.” It is a six-layer model for accurate computer control, two production defects repaired during the investigation, and a stronger rule for evidence: a transport receipt is not an application receipt.

II. BACKGROUND AND RELATED WORK

The original article does not contain a dedicated related-work section; its cited platform documentation is retained in References and discussed where it directly supports the measurements.

III. METHODOLOGY

A. Experimental scope

The controlled environment was:

- Windows 10 Pro N build 19045
- Chrome 151.0.7922.138
- AMD Athlon X4 870K, 4 logical processors
- 7.9 GB RAM
- NVIDIA GeForce GTX 1660
- one 1920×1080 display
- Windows system and window DPI 96, or 100% scaling
- devicePixelRatio = 1
- normal signed-in Chrome Profile 1

This is a single-machine field study. It does not claim mixed-DPI correctness, multi-monitor correctness, cross-browser

equivalence, cross-machine latency equivalence, or universal superiority of one automation technology.

The reversible browser target was Gemini's `Upload & tools` control. A trial counted as successful only when an independent browser postcondition proved the expected menu state. API return status alone never counted.

The final evidence package is SHA-256 bound. Key artifacts include:

- HWND birth-binding study: 5b49878e7e04d631a5b2292f3200ebaf2b1957d3a92315f63b86035ff0318058
- combined UIA study: 3317261ce0245b1b5fcfdbdb61bde7c44ee58d49ad42c929695ffa148d668d42
- combined CDP study: ba4065180697a8037c6befbeddac9864b3c65fb00cb9041f8ba5ba969f18a974
- coordinate-mapping study: b4acc799e3f15ba7ee8fb3c4383b42e74d06aab285c5d4c7709dcfc56f033f40
- occlusion study: e9a972d548c606353aa4f093e42206f6235af6ca1ecd4704155611020e69a491

IV. RESULTS

B. Result 1: bind a connector-created native window at birth

Native-window ambiguity is an avoidable source of automation error. Rediscovering a browser later by title or by whichever controls happen to be visible couples identity to mutable UI state.

Skynet instead enumerated visible top-level Chrome HWNDs immediately before and after opening a dedicated Gemini window and calculated the set difference.

Across 50 fresh connector-created windows, exactly one new visible Chrome HWND appeared in 50/50 trials.

- median open latency: 2,197.817 ms
- mean: 2,240.169 ms
- standard deviation: 248.691 ms
- p95: 2,534.892 ms
- p99: 2,654.542 ms

This is not a universal Windows guarantee. The production rule remains fail closed when the delta is zero or greater than one. The useful finding is that creation-time binding can establish native identity before later UI state becomes part of the problem.

C. Result 2: much of the old UIA latency was Skynet implementation overhead

The existing managed PowerShell/.NET trusted UIA path resolved the exact target and performed a foreground-verified clickable-point action.

Across 56 accepted trials:

- application successes: 56/56
- median action latency: 2,250.911 ms
- mean: 2,317.116 ms
- standard deviation: 195.566 ms
- p95: 2,774.774 ms
- p99: 2,928.005 ms

A cached in-process Python COM client then exercised the same basic safety contract: exact HWND, exact accessible name, one matching button, enabled/offscreen checks, `GetClickablePoint`, foreground verification, physical input, and an independent browser postcondition.

Across 59 attempts:

- accepted actions: 58/59
- one typed failure: no_clickable_point
- false-positive application successes: 0
- all-attempt median: 113.856 ms
- mean: 122.254 ms
- p95: 162.094 ms
- p99: 246.012 ms

The conclusion is deliberately narrow. This does not prove that Python is universally faster than PowerShell, or that one language is inherently better for UIA. Both paths use Windows UI Automation. The measured difference points to avoidable Skynet implementation costs such as process/runtime startup and repeated cross-process UIA work.

Microsoft's UIA documentation supports the architectural direction: property and pattern retrieval crosses process boundaries, and batching/caching requested properties and patterns can improve performance.

The engineering implication is therefore: keep the identity and fail-closed checks, but move frequently used UIA work into a persistent narrow client and cache only the properties and patterns needed for the exact action.

D. Result 3: the first CDP conclusion was wrong

The initial small CDP experiment appeared to show that `Input.dispatchMouseEvent` could not activate the Gemini control. That interpretation did not survive review.

Inspection found a concrete instrumentation defect in Skynet's own extension validator: it preserved `button='left'` but silently discarded the protocol's `buttons` bitfield. A supposedly complete press sequence therefore could not faithfully express `buttons=1` while the left button was down.

The production validator was repaired under the blast-radius guard. It now preserves `buttons=0` or `buttons=1` and rejects unsupported bitfields. The relevant extension security tests passed, and the later focused advisor/extension regression reached 130 passing tests.

After the repair, the corrected pointer experiment used:

- exact DOM target center
- elementFromPoint target hit-test
- mouseMoved
- mousePressed with left button and buttons=1
- mouseReleased with buttons=0
- independent browser menu postcondition
- target event logging

Across 50 corrected trials, the application postcondition succeeded in 50/50.

- median action latency: 1,324.998 ms
- mean: 1,358.405 ms
- p95: 1,550.237 ms
- p99: 1,648.853 ms
- trusted target-event trials: 50/50

The dominant event sequence in 46/50 trials was:

```
focus -> pointermove -> mousemove -> pointerdown -> mousedown -> pointerup -> mouseup -> click
```

Four trials additionally observed pointerover/mouseover before movement.

This is the most important scientific correction in the project. The earlier apparent CDP failure was not evidence that CDP pointer input was intrinsically inaccurate. It was evidence that the measurement path itself was incomplete.

E. Result 4: DOM activation and trusted pointer input are different semantics

For comparison, the exact target's `HTMLElement.click()` method was called through `Runtime.evaluate` in 50 trials.

The application postcondition also succeeded in 50/50.

- median latency: 1,260.122 ms
- mean: 1,356.288 ms
- p95: 2,405.928 ms
- p99: 2,470.285 ms

The event semantics were different. The observed DOM click was synthetic (`isTrusted=false`), while the repaired CDP pointer trials produced trusted pointer/mouse events.

That distinction matters even when both transports happen to reach the same application state. A useful automation receipt should record what kind of activation occurred, not collapse every successful action into `clicked=true`.

F. Result 5: coordinate spaces must remain explicit

Windows UI Automation uses physical screen coordinates for clickable points and bounding rectangles. CDP mouse input uses main-frame viewport CSS pixels.

At 96 DPI / 100% scale, Skynet measured 10 moved-window positions for the target.

- valid positions: 10/10
- devicePixelRatio: 1
- raw DOM-to-UIA translation was not a constant offset
- an explicit viewport-CSS-to-screen transform predicted the UIA X coordinate with 0 px maximum absolute error
- Y error was within 8 px in the measured configuration

That result is intentionally bounded to this one 96-DPI, single-monitor environment. It is not evidence for mixed-DPI or multi-monitor correctness.

The design lesson is stronger than the specific numbers: coordinate spaces should be treated as typed values. A DOM coordinate must not silently become a physical Windows mouse coordinate, and a UIA physical coordinate must not silently become a CDP viewport coordinate.

G. Result 6: occlusion is a correctness problem, not a visual detail

A physical click can be wrong even when the semantic element and coordinates were correct moments earlier. Another top-level window can take ownership of the target point.

The final occlusion study first sanity-checked the occluder with `WindowFromPoint` and root-window ownership. An earlier helper implementation with an incorrect 64-bit ctypes HWND return declaration was invalidated rather than counted.

Under the corrected verified occlusion condition:

- guarded trials: 50
- topmost helper window proven over the target point: yes
- UIA clickable point available while occluded: 0/50
- actions blocked fail-closed: 50/50

This is exactly the behavior a physical-input path should prefer. A controller should not click because it remembers where a control used to be. It should prove who owns the point at action time.

H. Result 7: UIA readiness is a capability that must be proven

Attempts to assign a clean causal distribution to Chromium UIA-tree propagation did not produce defensible evidence. A Chromium UIA COM subscriber error appeared during repeated propagation measurement.

That negative result is important: no propagation-latency distribution is published, and no causal AXTree/UIA lag number is claimed.

The safer rule is operational: before timing or invoking a UIA action, prove that the provider surface needed for that action actually exists and is actionable.

I. Result 8: exact local file-input state can still be a false success

The most consequential production defect discovered during the project involved Gemini attachments.

Skynet's extension could construct a synthetic `FileList`, assign it to a hidden file input, and read the exact filename and byte size back. That proved local browser input state.

It did not prove provider acceptance.

The production advisor contained a fail-open path that could still credit exact local input readback after provider-readiness verification failed.

That contract was changed. Gemini attachment delivery now fails closed unless provider-side readiness is proven.

This lesson generalizes more safely than any benchmark number: local browser state, transport state, and remote/provider acceptance are different receipt classes.

V. DISCUSSION

J. The six-layer click truth model

These experiments suggest replacing generic `clicked=true` with six explicit proofs.

K. 1. Window identity

Which browser tab and which native HWND are controlled? Creation-time binding can be stronger than rediscovery from mutable titles or later UI state.

L. 2. Element identity

Which exact semantic element is intended? Use the identity mechanism owned by the layer: accessible name/role/control type, automation ID, DOM identity, and uniqueness as appropriate.

M. 3. Coordinate-space identity

Are coordinates viewport CSS pixels, logical desktop coordinates, or physical screen coordinates? Carry the coordinate system as part of the contract.

N. 4. Action semantics and fidelity

Was the action UIA Invoke/other semantic pattern, UIA clickable-point physical input, CDP pointer input, DOM activation, keyboard input, or something else?

O. 5. Foreground and occlusion proof

For physical input, which root window owns the point at the moment of action? If ownership is wrong or ambiguous, fail closed.

P. 6. Independent application acceptance

What changed on a different surface after the action? Examples include visible DOM state, URL transition, provider-visible attachment, application counter, accessibility state, stored server state, or another durable receipt.

Q. What changed in Skynet

This study produced production changes, not just measurements.

- The CDP transaction validator now preserves the mouse buttons bitfield instead of silently stripping it.
- Unsupported mouse bitfields fail closed.
- Gemini attachment delivery no longer treats exact local input.files readback as sufficient proof of provider acceptance.
- Relevant focused regressions reached 130 passing tests after the final corrections.

The strongest next implementation change is to move frequently used trusted UIA work into a persistent in-process client while preserving exact-window binding, semantic uniqueness, foreground checks, occlusion checks, and independent postconditions.

VI. THREATS TO VALIDITY

R. Limitations

This field study does not establish:

- mixed-DPI correctness
- multi-monitor correctness
- cross-browser equivalence
- cross-machine latency equivalence
- raw CDP API latency independent of Skynet's authenticated extension/lease path
- universal UIA or CDP superiority
- a valid UIA accessibility-tree propagation latency distribution

Only 96 DPI / 100% scaling and one monitor were physically available. Those unavailable conditions are left as replication targets instead of being simulated and reported as if they had been directly measured.

VII. CONCLUSION

The useful engineering question is not "UIA or CDP?"

It is: **which layer owns the truth for this action, and what independent evidence proves that the action landed?**

On this machine, the older managed UIA route was reliable but paid large avoidable implementation overhead. A cached in-process UIA client reduced action latency substantially while still preserving typed failure. Correctly instrumented CDP pointer input passed 50/50 on the tested control and produced trusted pointer/mouse events. DOM activation also passed 50/50 but with different event fidelity. An explicit coordinate transform matched the 96-DPI environment closely. The occlusion study showed why physical input must verify current point ownership. And the attachment investigation found a production case where exact local state could still be a false success.

Most importantly, the study corrected itself. An early benchmark interpretation was falsified by a better-controlled experiment, and the process uncovered the instrumentation bug that had inverted the result.

That is the standard reliable computer control should meet: explicit identity, typed coordinates, fidelity-aware actions, immediate visibility/ownership checks, independent postconditions, raw evidence, and conclusions that are allowed to change when the experiment says they should.

KEY TAKEAWAYS

- Bind identity early. Creation-time HWND binding was exact in 50/50 controlled browser-window trials, while ambiguity still remains fail-closed.
- Optimize implementation, not safeguards. A cached in-process UIA path cut Skynet's action overhead substantially while preserving typed failures and postconditions.
- Audit protocol fidelity. A bug that stripped CDP mouse buttons inverted the first benchmark result; after repair, corrected pointer input succeeded 50/50.
- Separate proof layers. Coordinate space, foreground ownership, occlusion, transport semantics, and provider acceptance must not collapse into clicked=true .

REFERENCES

- [1] Microsoft Learn – UI Automation screen scaling: https://learn.microsoft.com/en-us/windows/win32/winauto/uiauto-screenscaling

- [2] Microsoft Learn – UI Automation CacheRequest: https://learn.microsoft.com/en-us/dotnet/api/system.windows.automation.cacherequest?view=windowsdesktop-10.0
- [3] Chrome DevTools Protocol – Input domain: https://chromedevtools.github.io/devtools-protocol/tot/Input/
- [4] WHATWG HTML – interaction and synthetic click behavior: https://html.spec.whatwg.org/multipage/interaction.html
- [5] Chromium – Windows UI Automation: https://chromium.googlesource.com/chromium/src.git/+/refs/heads/main/docs/accessibility/browser/uiautomation.md
- [6] W3C – WebDriver BiDi: https://www.w3.org/TR/webdriver-bidi/

AUTHOR BIOGRAPHY

EXZIL CALANZA is an independent researcher and software builder based in Iloilo City, Philippines. His work focuses on browser and desktop automation, machine learning, reproducible evaluation, workflow verification, and AI-assisted systems engineering.