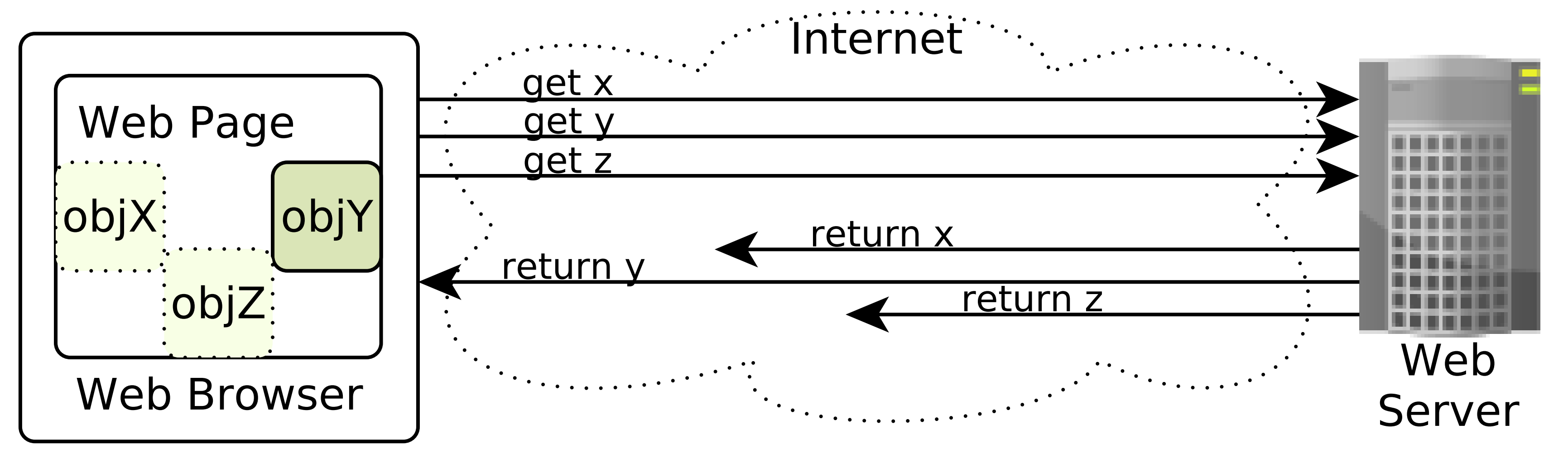
MIRROR: Automated Race Bug Detection for the Web via Network Events Replay

Sze Yiu Chau [1], Hyojeong Lee [1], Byungchan An [1], Julian Dolby [2] and **Cristina Nita-Rotaru [1]**
Computer Science, Purdue University [1]   IBM TJ Watson [2]

Race Conditions in JavaScript

- JavaScript is very popular in Web App development
- Unconventional execution model
  - → Single-threaded
  - → Asynchronous and Event Driven
- Non-blocking, improves perceived user experience
  - → Examplifed by techniques like AJAX
- Lack of sychronous mechanism in the language
  - → Crude hand-crafted solution
  - → Unexperienced developers are error-prone
- Major source of race condition is network non-determinism



- Cannot guarantee handler functions always execute as expected !

Testing and Evaluation

- We used a benchmark of eight Web Apps from literature to test MIRROR, each captures a representative buggy coding pattern:

Web App #	Benchmark Web App bug description	MIRROR detected
1	concurrent network objects: network events can happen in non-deterministic order	Yes
2	dynamic script loading: script can be loaded arbitrarily late	Yes
3	`iframe` pages can be parsed in non-deterministic order	Yes
4	timer events can happen before or after other desired events in non-deterministic ways	Yes
5	browser pauses parsing and wait for static embedded external scripts	Yes
6	long HTML file pauses parsing	No
7	`alert()` pauses execution but other events might fire in background	Yes
8	asynchronous XHR call: response can arrive arbitrarily late	Yes

- Number of test case generated and efficiency:

Web App #	User Actions	Network Objects	Test Orders Generated	Successful Test Orders Before Bug Detected
1	2	6	2520	211
2	2	7	10080	3
3	1	7	1260	253
5	3	5	420	46
7	4	3	30	4
8	2	6	1260	147

Future Work

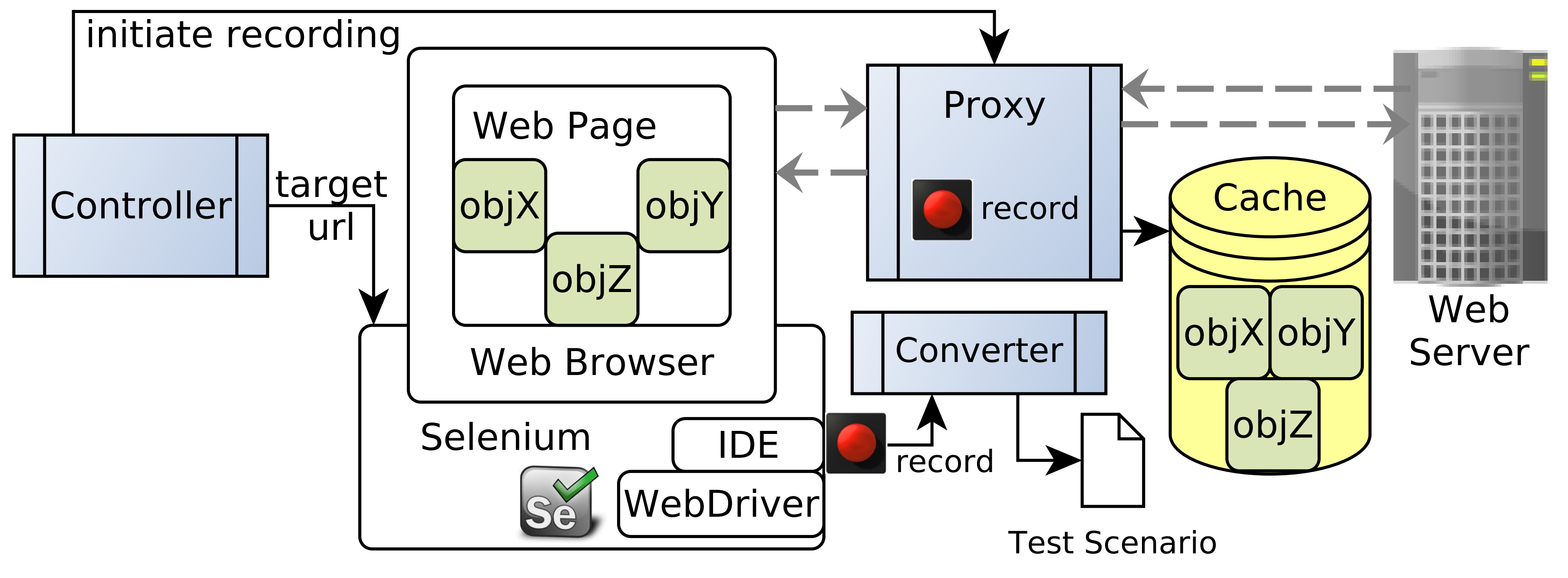
- Towards a debugging tool: reverse identify faulty DOM elements
- More heuristics in analysis to futher improve performance

MIRROR

- Goal: Automatically detect race bugs in Web App with high portability and minumum human intervention
- Approach: **3-phase proxy-based** solution
  - → Proxy is used to capture network non-determinism
  - → Controller orchestrates components and decides if a bug is found

Phase 1: Recording

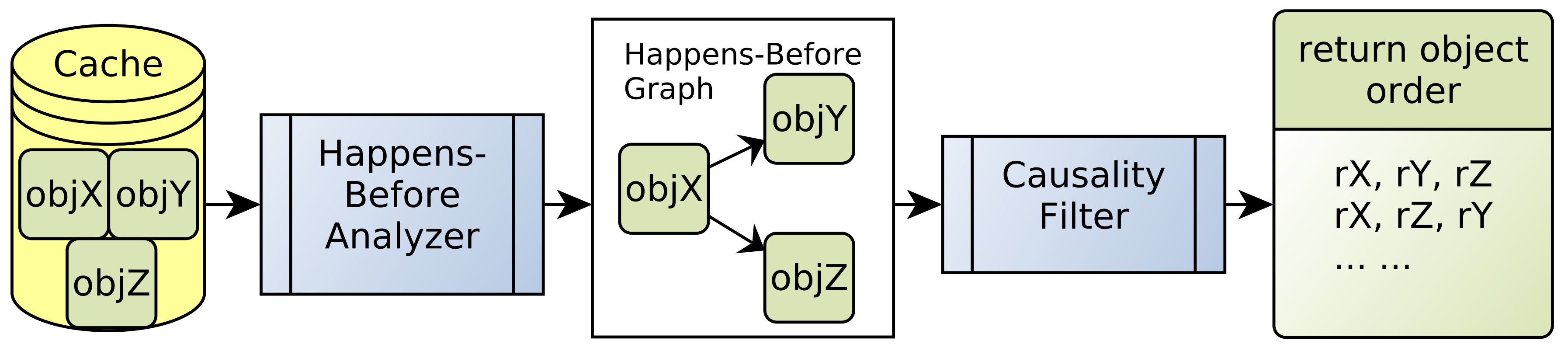
Purpose: Record both network objects and user actions



Saves a recorded session on disk, ready to be analyzed

Phase 2: Analysis

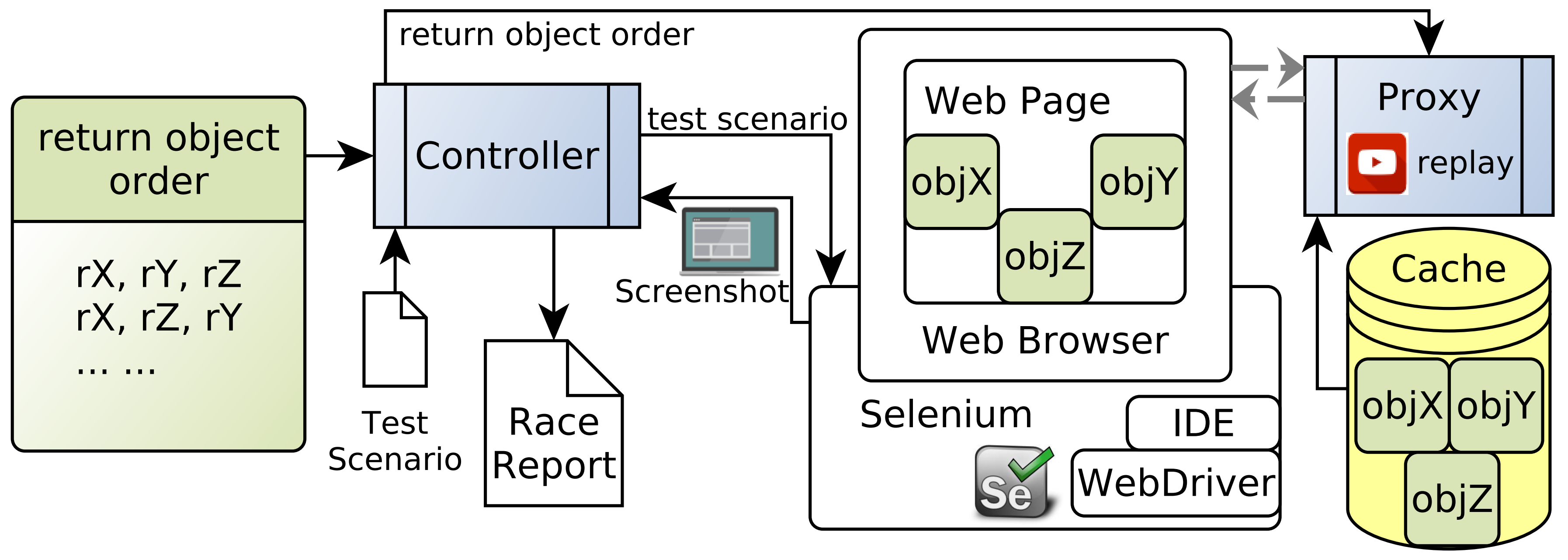
Purpose: Stratigically filter out meaningless replay orders



Returns a list of replay orders that maybe worth testing

Phase 3: Dectection

Purpose: Reorder arrival of network objects and observe script execution



Produces a report containing visual clues if a race bug is found