

Sea Lock

An AI-generated illustration of a naval battle scene. In the foreground, a large grey naval ship is firing a missile, with a bright orange and white plume trailing behind it. Several other grey naval ships are visible in the mid-ground, some with smoke rising from them. In the background, there are industrial structures and a city skyline under a sunset sky with mountains in the distance.

A Multi-Agent Reinforcement Learning Framework
for Adversarial Naval Blockade Simulation
and Predictive Command and Control

Yuke Li

Founder, The Adversarial Lab Inc.
Carleton University (NPSIA)

Hao Ji, Husnain Qadri, Yash Malode

University of Southern California

Illustrative image (AI-generated)

MESAS 2026 · Prague
21–23 October 2026

Strategies must hold up against learned opponents

What is a learned agent?

A decision-maker trained through interaction and reward. Here, a ship learns how to act in simulation.

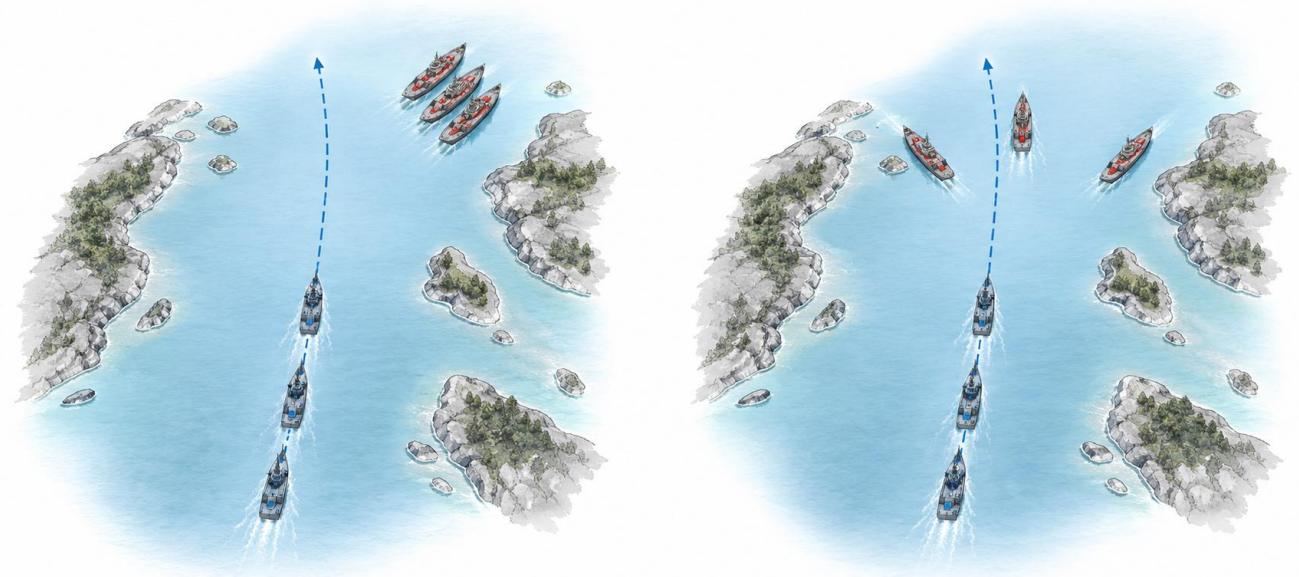
Existing tools

NATO's Maven Smart System proposes opponent actions; STILMAN's LG-RAID wargames courses of action [1-3]

What we need

Test whether learned behaviour holds when the opponent changes

Illustration: same blue, different red



Training opponent

Other opponent

Agents learn during training.
Their decision rules are fixed for evaluation.

Multi-agent reinforcement learning in Sea Lock

Why naval



Decisions are interdependent

Each ship acts alongside friendly ships and against an opposing fleet.

Learning needs interaction [1]

Scarce engagement data motivate learning through simulated encounters.

HOW NAVAL TOOLS REPRESENT THE OPPONENT



Analytic combat models

Salvo models: Hughes, Armstrong [2, 3]

Each side as striking, defensive and staying power



Simulation and wargaming

NSS, JTLS-GO; Naval War College games [4-6]

Opponents: human players or modelled commander decisions



Learned naval agents (emerging)

Littoral MARL; Aquaticus capture-the-flag; ERDC [1, 7, 8]

Set-ups vary; ERDC trains blue against a previously trained red

MARL in Sea Lock

Within each fleet

Ships learn towards a shared objective

Between fleets

Opponents shape the value of each action

Will a strategy hold, and where will it lead?

Q1 · Adaptive adversaries

Does training produce strategies that hold when the opponent changes? [1]

Why it matters



Learned policies can overfit their training partners, so a strategy that wins may fail against a new opponent.

How we test it



Save self-play strategies as snapshots, fixed for evaluation, then re-pair them against other opponents.



Would let us

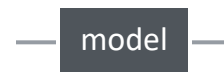
Tell which strategies hold against new opponents.

Q2 · Prediction from interaction

Given a strategy's actions, can a model project how the engagement unfolds? [2]

Why a learned model

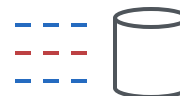
Simulation generates the games; a learned model makes them queryable from what is observed.



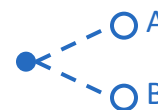
Q1's games



train Q2



Built on Q1's gameplay data
Real engagement data are scarce; Q1's self-play games supply them.



Would let us

Project how a strategy unfolds, then plan inside the model.

Model the game, train both fleets, learn to project

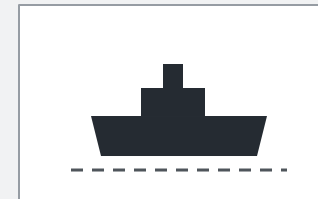


StarCraft II
AlphaStar [1]

AlphaStar vs. TLO (2019). Image:
DeepMind; StarCraft II ©
Blizzard Entertainment

Method transfers

self-play against a pool
of past opponents
✗ agent does not



Naval engagement
Explicit naval actions
and outcomes

FORMULATION



Stochastic game [2]

Simultaneous moves;
we add partial observation

Q1 SEA LOCK: CO-TRAIN IN ONE RUN, TEST ACROSS OPPONENTS



PPO + finite-pool self-play

A standard baseline [3, 4]; both
fleets learn in turn, in one run



Frozen cross-play

Snapshots, fixed for evaluation,
re-paired across opponents

snapshots replayed

Q2 PREDICTIVE WORLD MODELLING



Self-play games as data

Real engagement data are scarce,
so self-play games supply them



Recurrent world model [5]

Given a strategy's actions, project
how the engagement unfolds

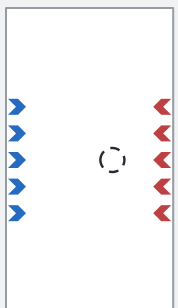
LLM reward shaping [6, 7]
An LLM's advice adds a
small, capped reward bonus,
tested in one training run.

The game: two fleets contest one port

Who and where

➤ **Breakthrough** must reach the port

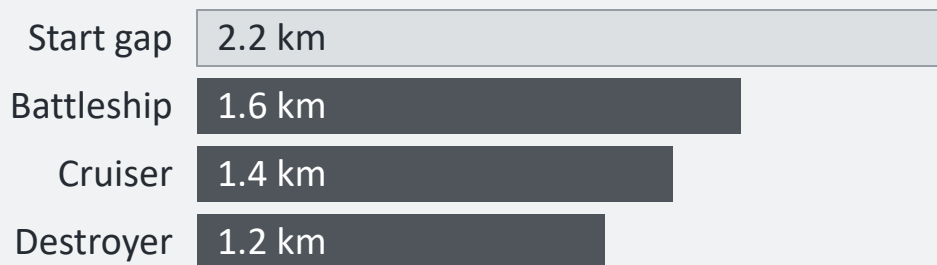
◀ **Blockade** must stop it



Class	Per side	Max speed	Reach
Destroyer	2	50 m/s	1.2 km
Cruiser	2	45 m/s	1.4 km
Battleship	1	40 m/s	1.6 km

Arena 2.2×4 km, to scale; dashed: port, radius 150 m

Weapon reach vs the start gap, to scale



Every 0.2 s, each ship chooses

- ➡ Move
throttle and turn
- 🎯 Fire
shares over live enemies
- 🛡️ Defend
share against attackers
- 👁️ Scout
sensing shares
- ⚡ Jam (EW)
jamming level
- 👉 Decoy
on or off

What the ship sees

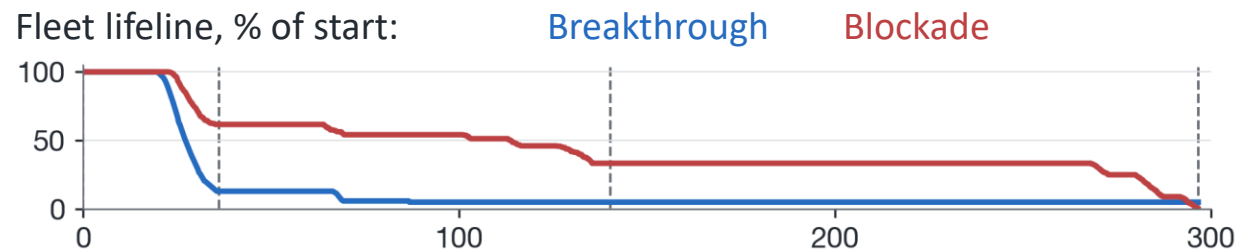
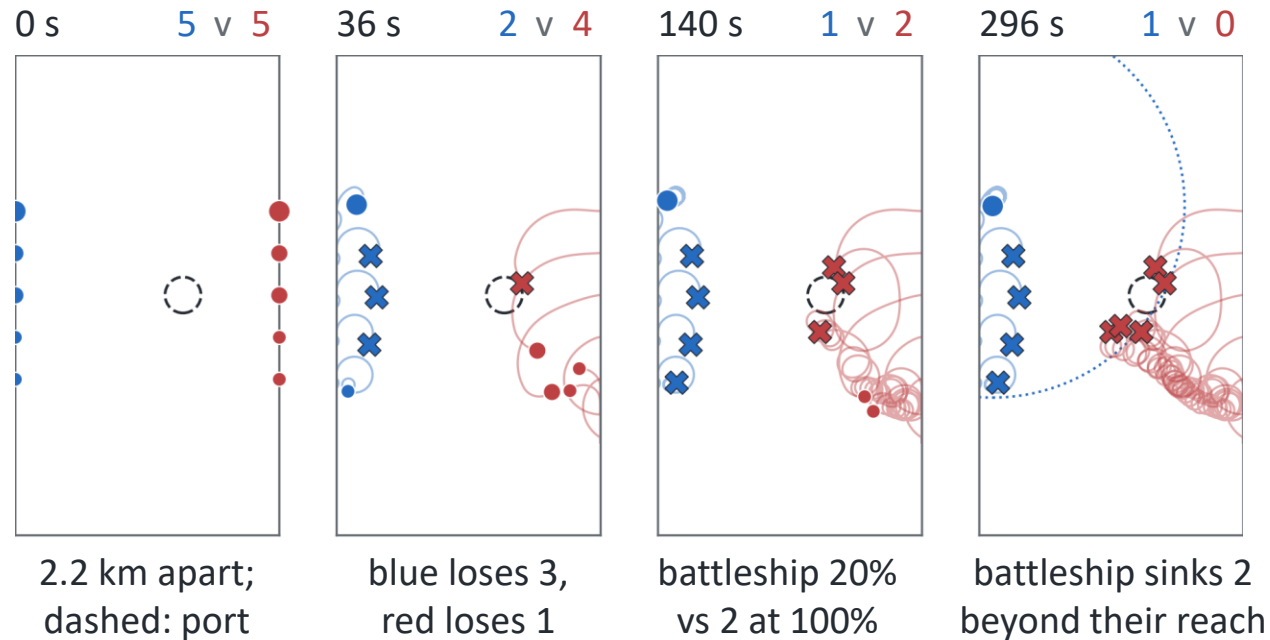
Own state, the port, its side's nearest-enemy range and how many enemies are afloat; no enemy positions.

How it ends

- Breakthrough wins: 2 ships in the port zone for 2 consecutive steps, or the Blockade fleet sunk
- Blockade wins: the Breakthrough fleet sunk
- Draw: undecided after 10,000 steps

Engine = the Sea Lock simulator: every 0.2 s it moves ships, updates detection, resolves fire, checks the end.

One engagement: decided by sinking, not at the port



In 1,500 self-play games across three seeds, no live ship ever won at the port.

A hit drains the lifeline



Within reach, damage grows with fire share and with offence over defence. At zero the ship sinks.

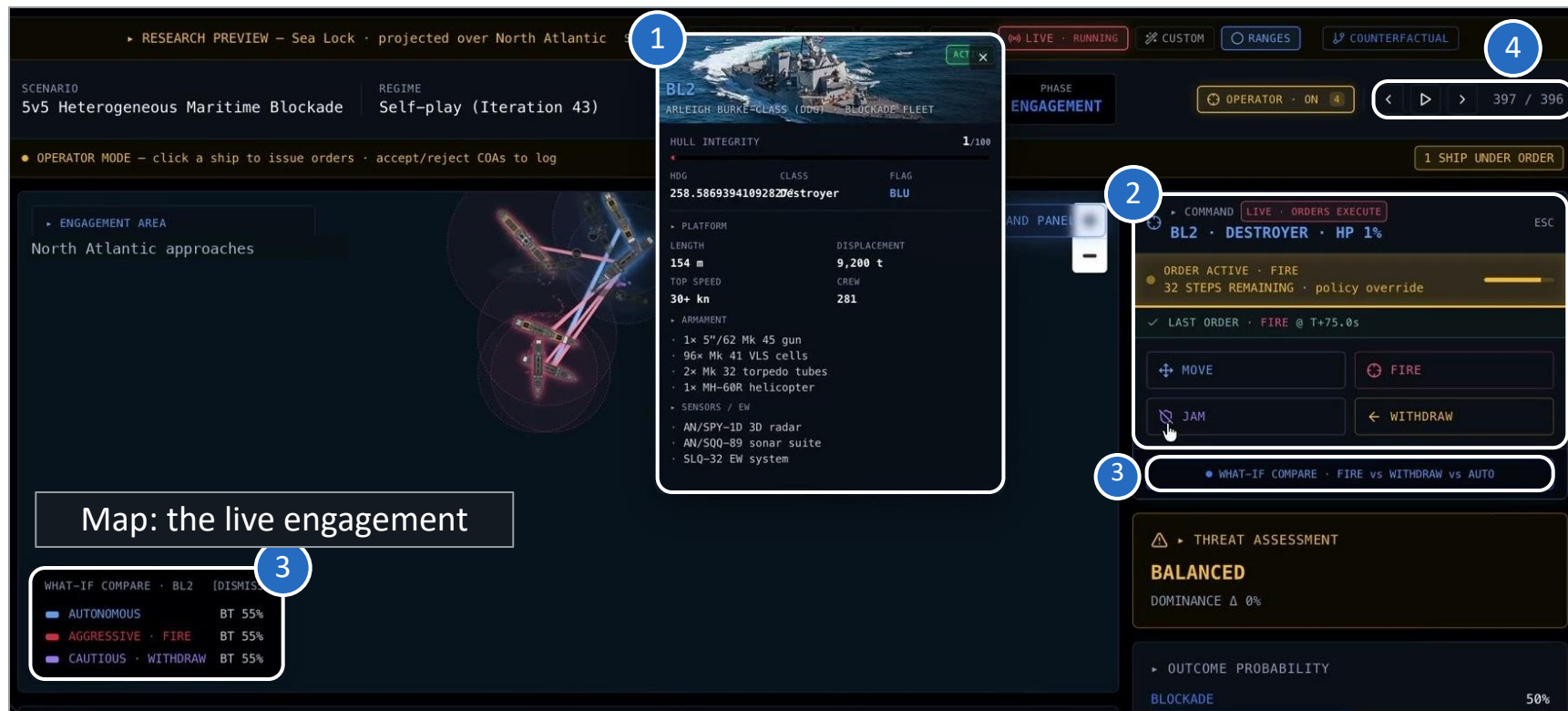
Reward: a win bonus, a trickle for surviving

Winning step: +5, plus that step's damage and scouting.
Every step: a trickle for ships afloat.

This engagement

Breakthrough wins by sinking all five Blockade ships, not by holding the port.

Interface: the same engine, as an analyst sees it



Demo policy: an earlier PPO checkpoint, not Seed A; "Iteration 43" is a display label.

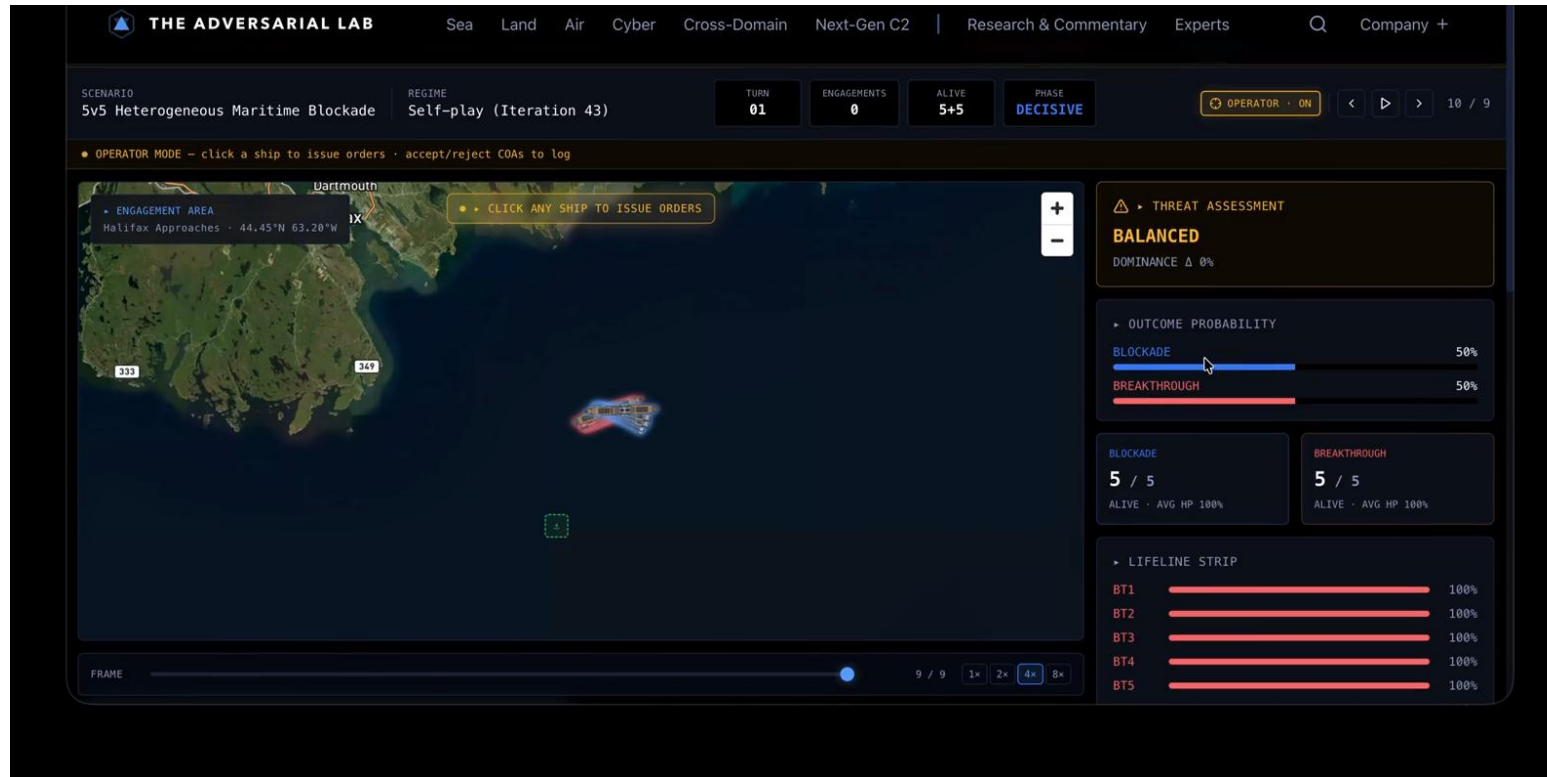
What-if: a 30-step simulator-clone preview
a copy of this engine runs 6 s ahead; not the world model

BT %: a surviving-strength heuristic
ships afloat and mean lifeline; not a learned probability

Same arena, same ships: an analyst can order, preview and replay on the engine itself.

- 1 Inspect
Click a ship to see its class, lifeline and heading.
- 2 Order
Override its policy for 10 s: move, fire, jam, withdraw.
- 3 What-if
Preview three choices 6 s ahead, before committing.
- 4 Replay
Step through a recorded engagement.

Interface in motion: a short walkthrough



What you see

- 1** Opening view
 0–6 s
 The map zooms in on both fleets at the start.
- 2** Select, preview, order
 6–21 s
 An analyst picks a ship, previews three choices and gives orders.
- 3** Compare two games
 21–33 s
 Two recorded self-play games, late and early in training, side by side.

Environment = engine + scenario

Abstract ships and sensors, fidelity scoped to the decision question.
 Training, testing and what-if all run inside it.

Today

2D web dashboard (this video)

Prototype

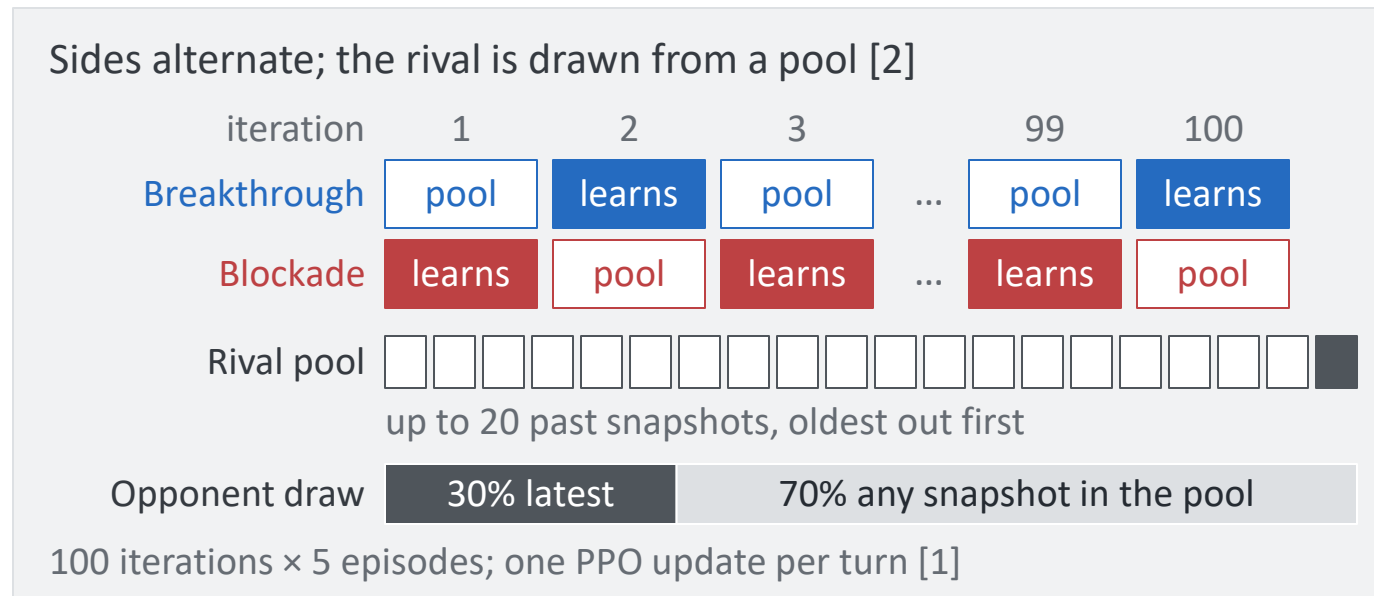
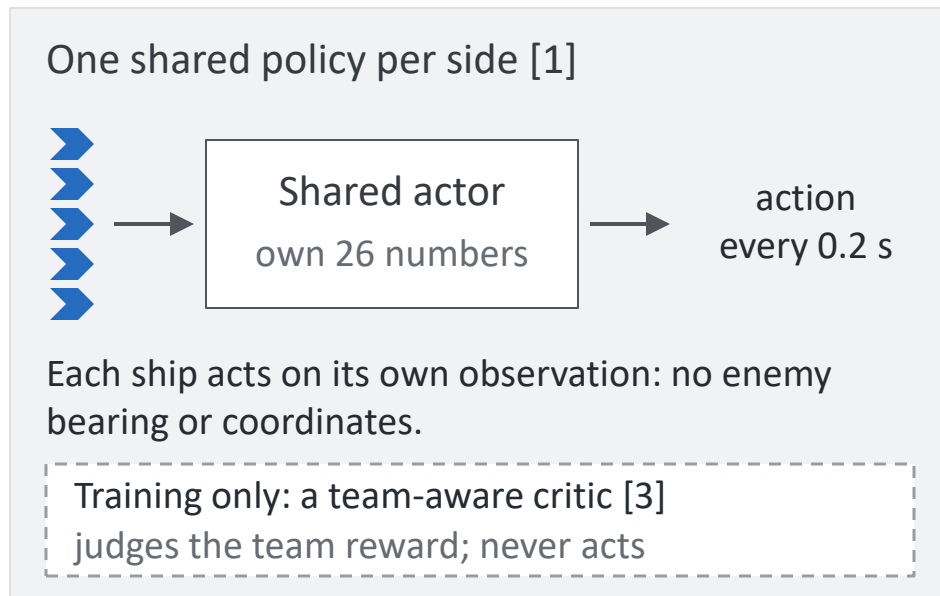
Unity 3D view, on a demo feed

Option

Command: Modern Operations, not built

Demo policies are earlier checkpoints, not Seed A. Preview: a simulator copy; percentages: a surviving-strength heuristic.

Training: shared policies, alternating snapshot pools



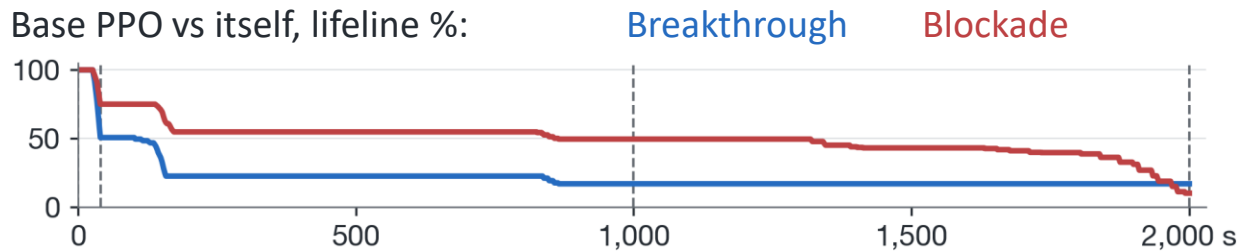
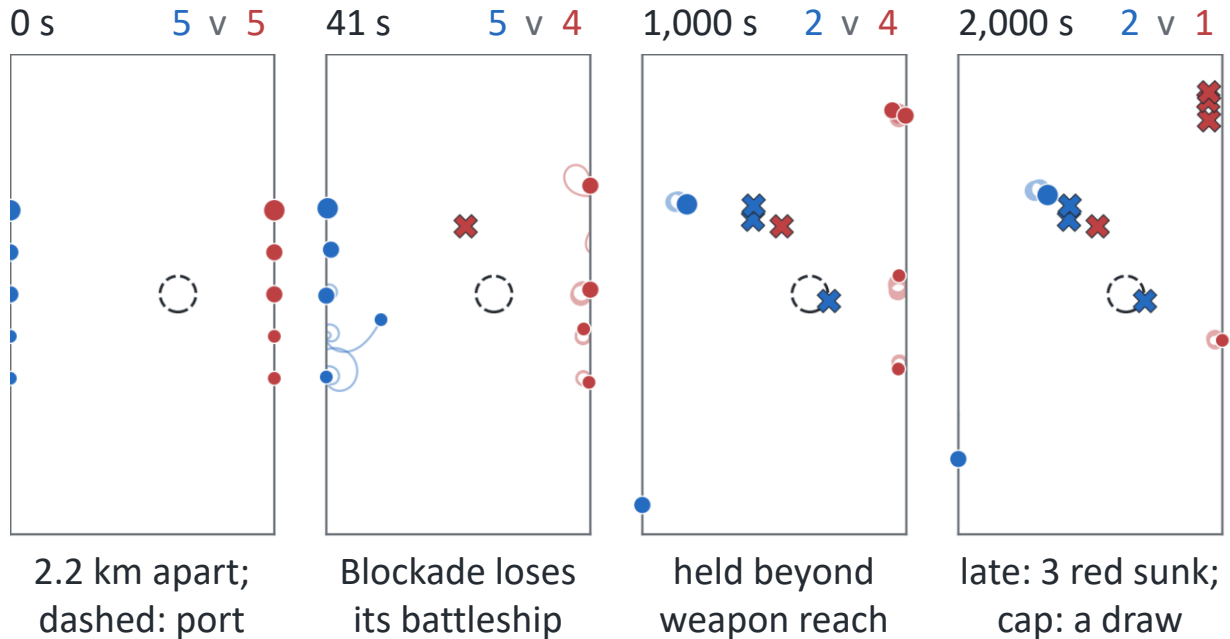
The opponents Q1 compares, simplest first



Base PPO and pool self-play also differ in curriculum, movement filter and Dirichlet settings (backup): compared as set-ups, not ablations.

Q1 asks, in order: baselines, brittleness, seeds, frozen re-pairing, checkpoint selection.

Base PPO stands off; most frozen pairings are decided



Same policy on both sides

Random random actions, 100 runs

BT 55

BL 45

Every game decided, near even

Rule-based nearest enemy, all fire on it

BT · one run, repeated

Both sink in one step; BT is checked first

Base PPO one network, both roles, no pool

draw · one run, repeated

Standing off still earns the survival bonus

Pool self-play trained vs a pool of past rivals

7 × 7 frozen Seed A snapshots, one trajectory each

BT 29

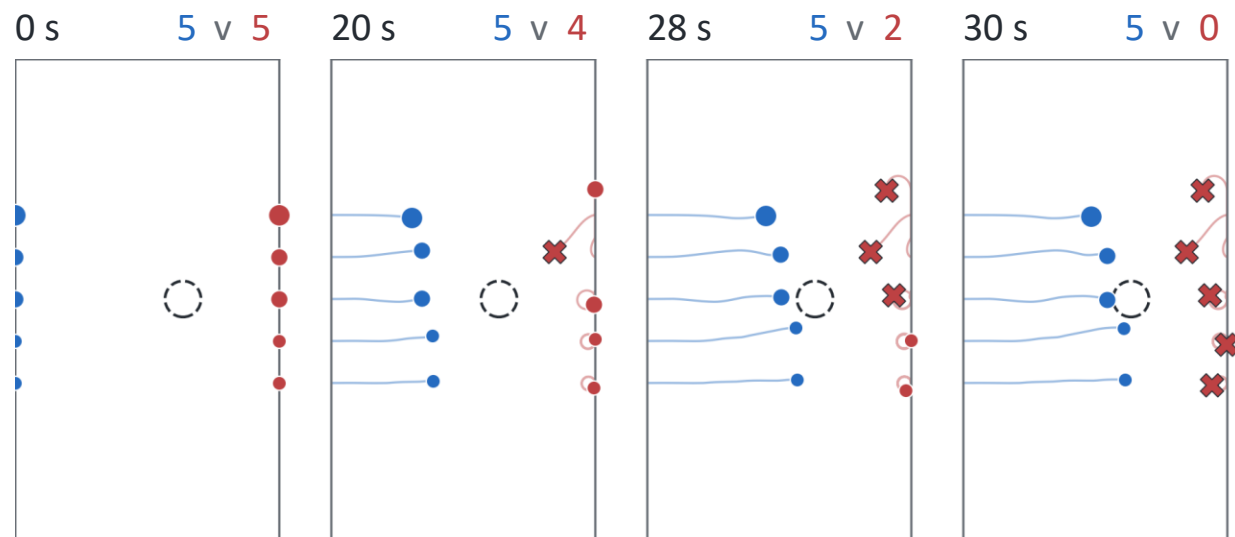
BL 8

draw 12

37 of 49 pairings decided

Frozen pairings give varied outcomes to test against; the set-ups differ, so this is not credited to the pool.

A scripted rush beats both learned Blockades



rush: full throttle,
all fire on nearest

Blockade loses
its battleship

Blockade circles,
even fire spread

no rush ship lost;
fleet at 35%



Base PPO vs scripted policies, wins

as BT vs Rule BL*	50/50
as BT vs Random BL	50/50
as BL vs Random BT	49/50
as BL vs Rule BT* (rush)	0/50

* one deterministic trajectory, repeated

Trained against the rush: test unchanged

Self-play Blockade (0/50 before), 60 iterations

15/60

batches won 10 of 10,
last at iteration 37

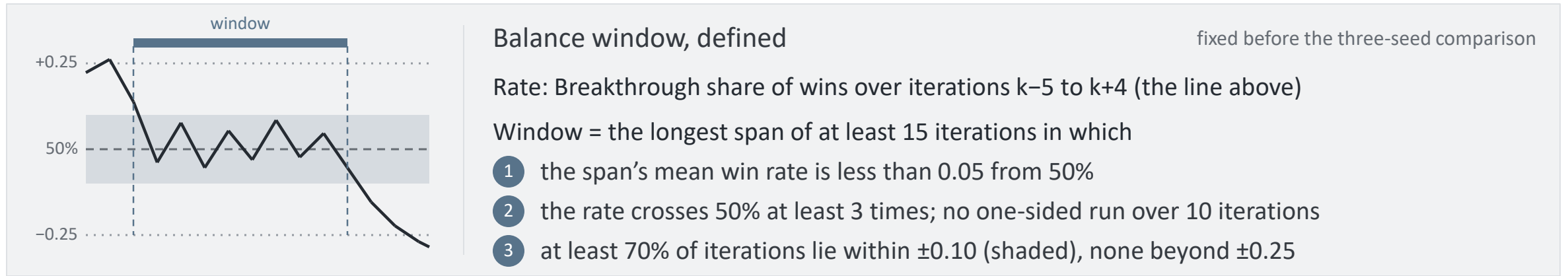
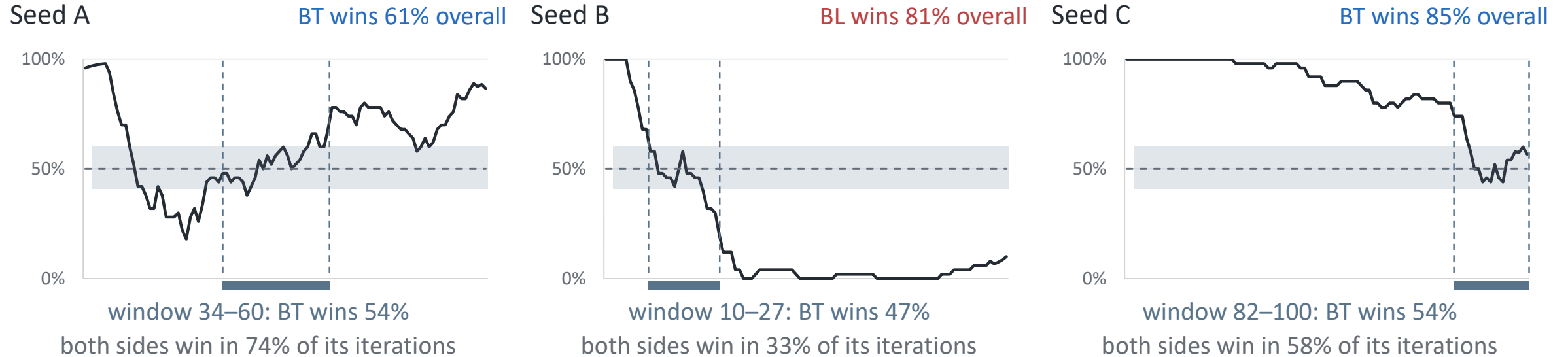


0/50

deterministic test,
final and best

Base PPO wins the other three pairings; training against the rush left the test at 0/50.

Each seed balances briefly, at a different time



Three seeds: different windows, different overall winners. Next, freeze Seed A's window and re-pair it.

Balanced in training, lopsided when frozen

Rows: Breakthrough snapshot

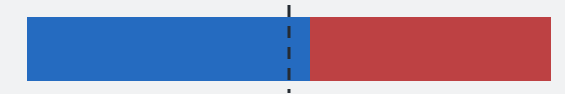
Columns: Blockade snapshot

BT \ BL	10	20	34	47	60	80	100
10	BT	BT	Draw	Draw	BT	BT	BT
20	BL	Draw	BT	BT	BL	BT	BL
34	BT	BT	BT	BT	BT	BT	BL
47	BT	Draw	BT	BT	BT	BT	BL
60	BT	Draw	BT	BT	Draw	BT	BL
80	BT	Draw	BT	BT	Draw	BT	BL
100	Draw	Draw	BT	Draw	Draw	BT	BL

The balance window, two ways

In training

54%



73 of 135 · sampled, movement filter on

Frozen for evaluation, re-paired

8 of 9



9 pairings · deterministic, no filter

Blue row: iteration 34 wins 6 of 7

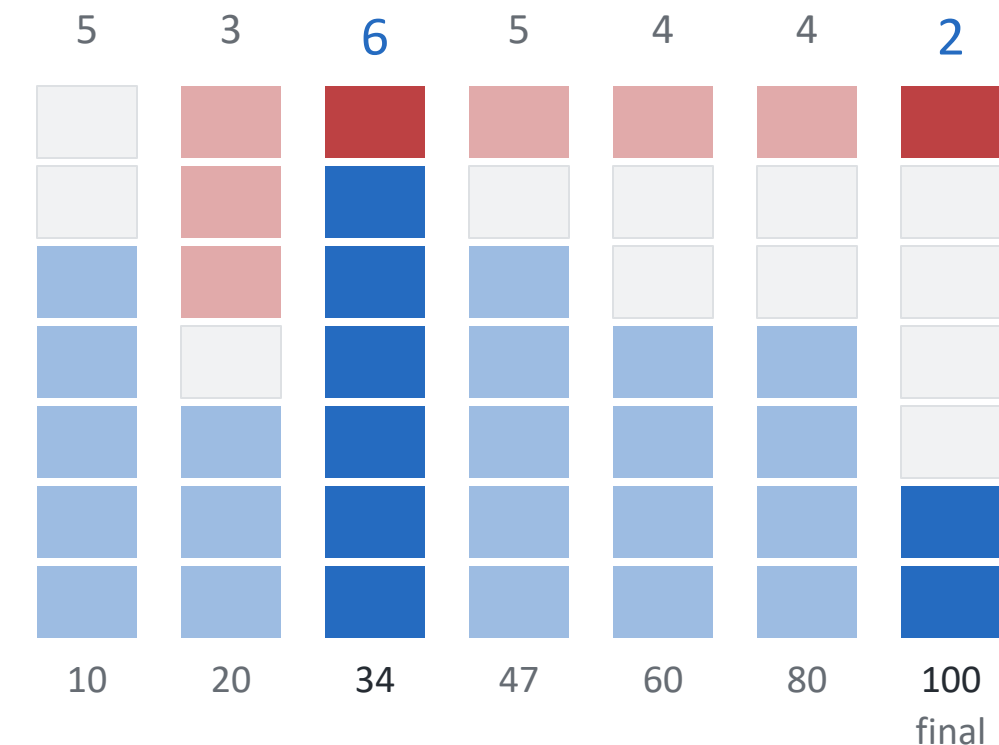
All 49 pairings: 29 BT, 8 BL, 12 draws

Framed: Seed A's balance window, iterations 34–60 · faded: outside it

Balance in training is not balance among frozen snapshots: which one you keep decides the outcome.

The best Breakthrough snapshot never breaks through

Each Breakthrough snapshot vs 7 Blockades

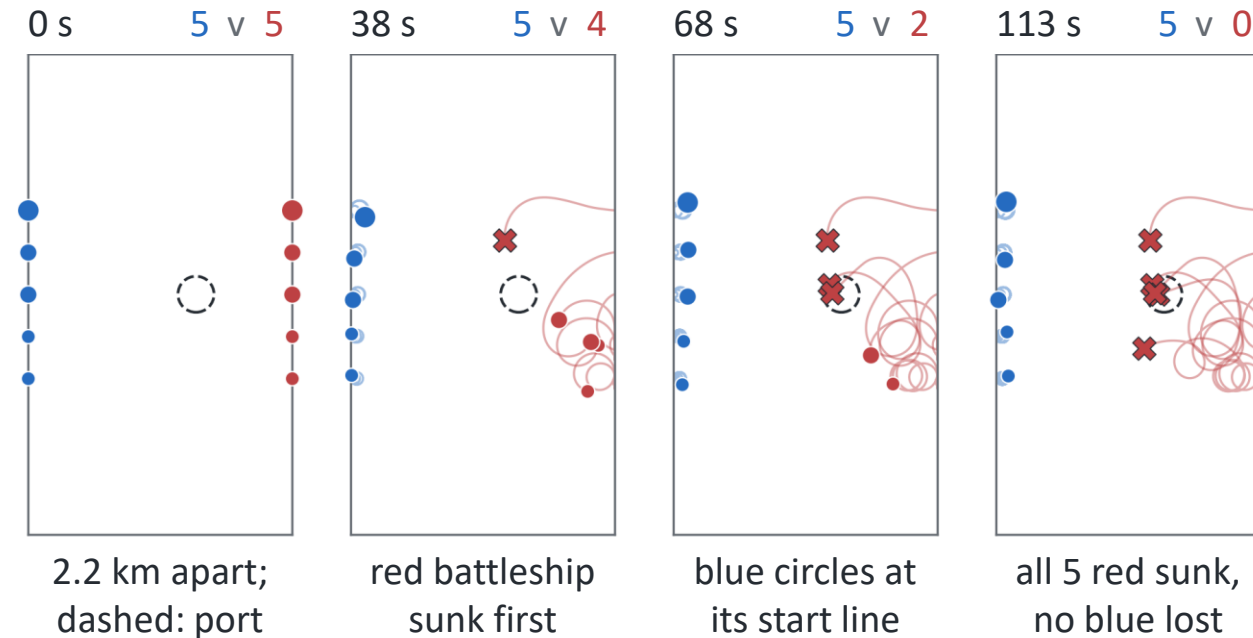


■ win ■ draw ■ loss

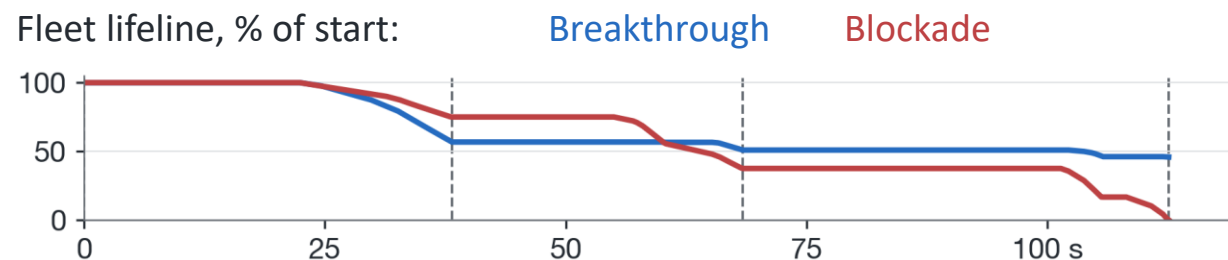
All 6 wins sink all 5 Blockade ships; no Breakthrough ship gets within 1 km of the port.

Sinking the Blockade fleet also counts as a Breakthrough win, so the label need not mean breaking through.

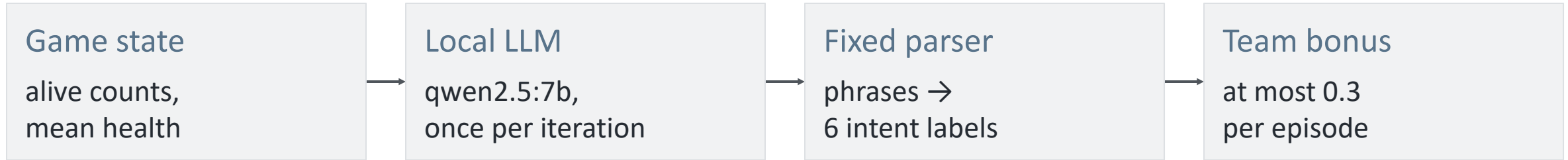
One win: iteration 34 vs Blockade 34, frozen



Fleet lifeline, % of start:



With a bounded language bonus, one pairing flips



What it asked for: 3 of 6 intents

100 calls, two intents each

Fire concentration 93

Mobility 78

Reconnaissance 29

The other three 0

* Rule pairings: one deterministic trajectory, repeated

Same four pairings, wins of 50

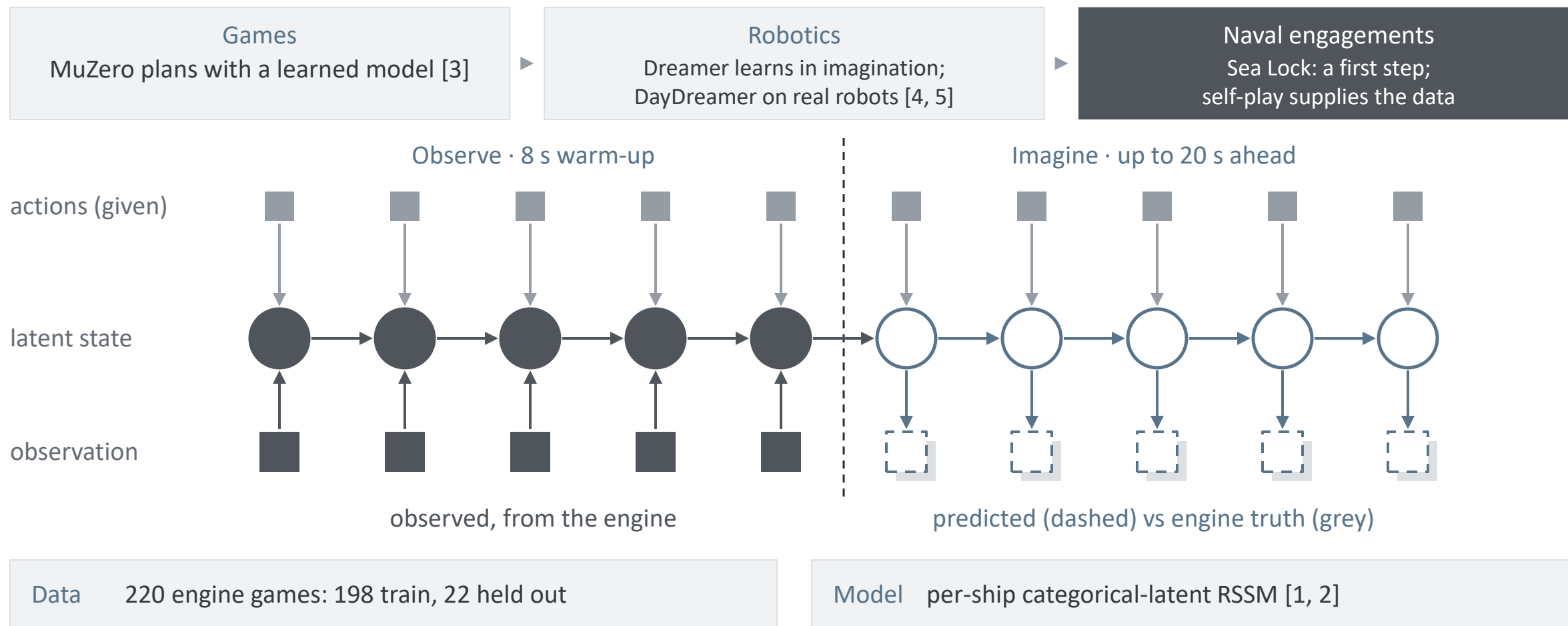
Pairing	Base PPO	Language-shaped
as BT vs Rule BL*	50	0
as BT vs Random BL	50	49
as BL vs Random BT	49	47
as BL vs Rule BT* (rush)	0	0

Bounded and inspectable: one capped bonus, every call and intent logged

One seed. Next: more seeds and an intent-level ablation to find the cause.

Language can enter training through a capped, logged channel; ablations come next.

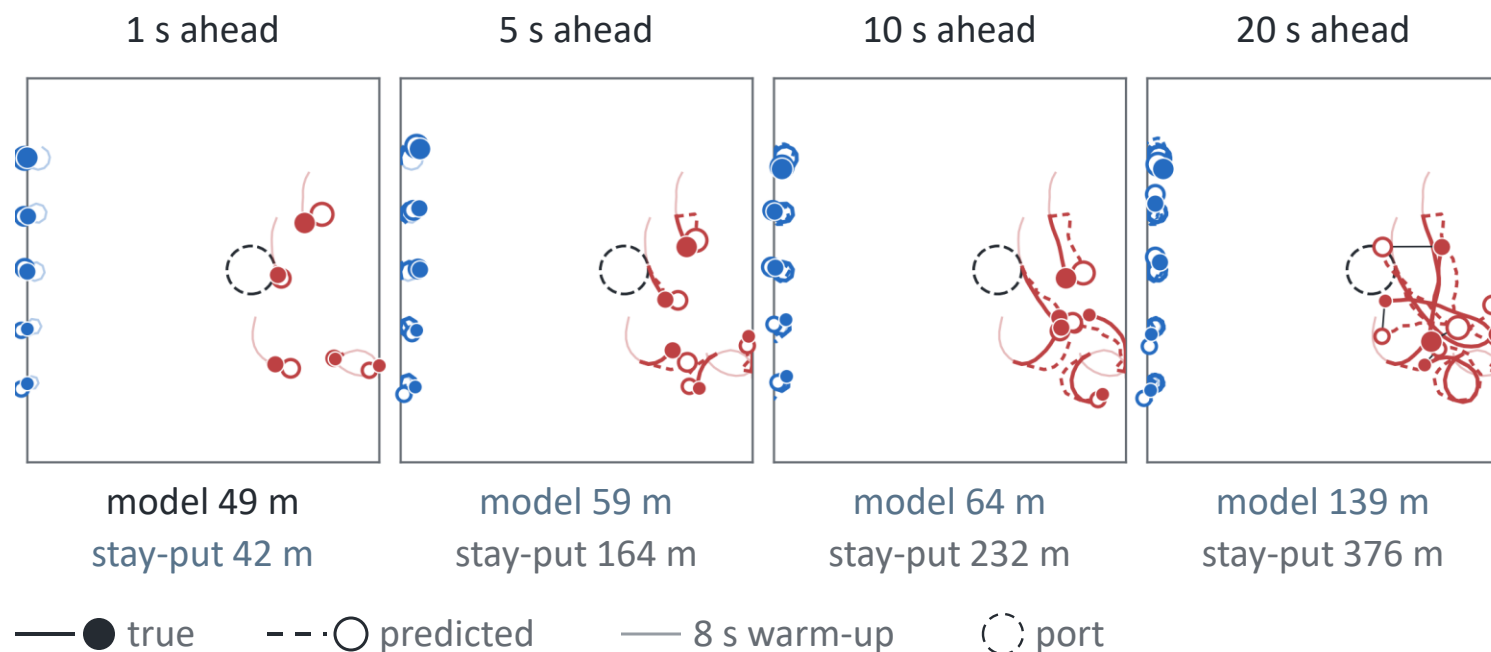
Toward learned world models for adversarial simulation



The engine plays the games; the world model learns from them and makes them queryable.

The world model outpredicts simple extrapolation

Position error, metres · one validation engagement



One moving window, picked by a fixed rule: median model error.

Per-ship model: each ship's track from its own observations and actions

Seed A iteration-80 snapshots, Breakthrough win · arena to scale, 2.2 km wide

Given the actions, from 5 s it tracks each ship far better than assuming nothing moves.

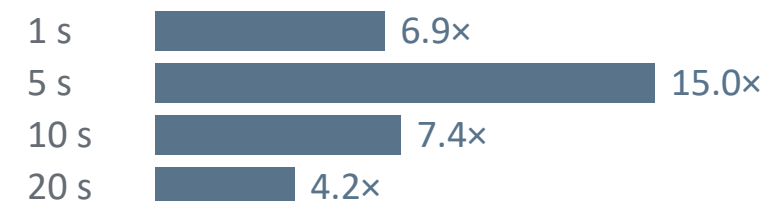
Median position error (m)

	Model	Stay-put	Const. vel.
1 s	42	31	12
5 s	56	131	129
10 s	82	198	265
20 s	131	264	572

From 5 s, given the actions, it beats both in all 146 overlapping windows from 13 held-out engagements.

Full-observation MSE ratio

persistence ÷ model, all 27 dims, not position only



It separates winners late in play and reacts to actions

Who wins? 13 decided validation engagements



x: share of the engagement's final length (retrospective)

Does it react to actions? Prediction shift per step

Random streams

0.212

10 pairs, 1 model
uniform random actions

Recorded BT vs BL

0.124

1 pair, mean of 3 models
first 20 s of each ship

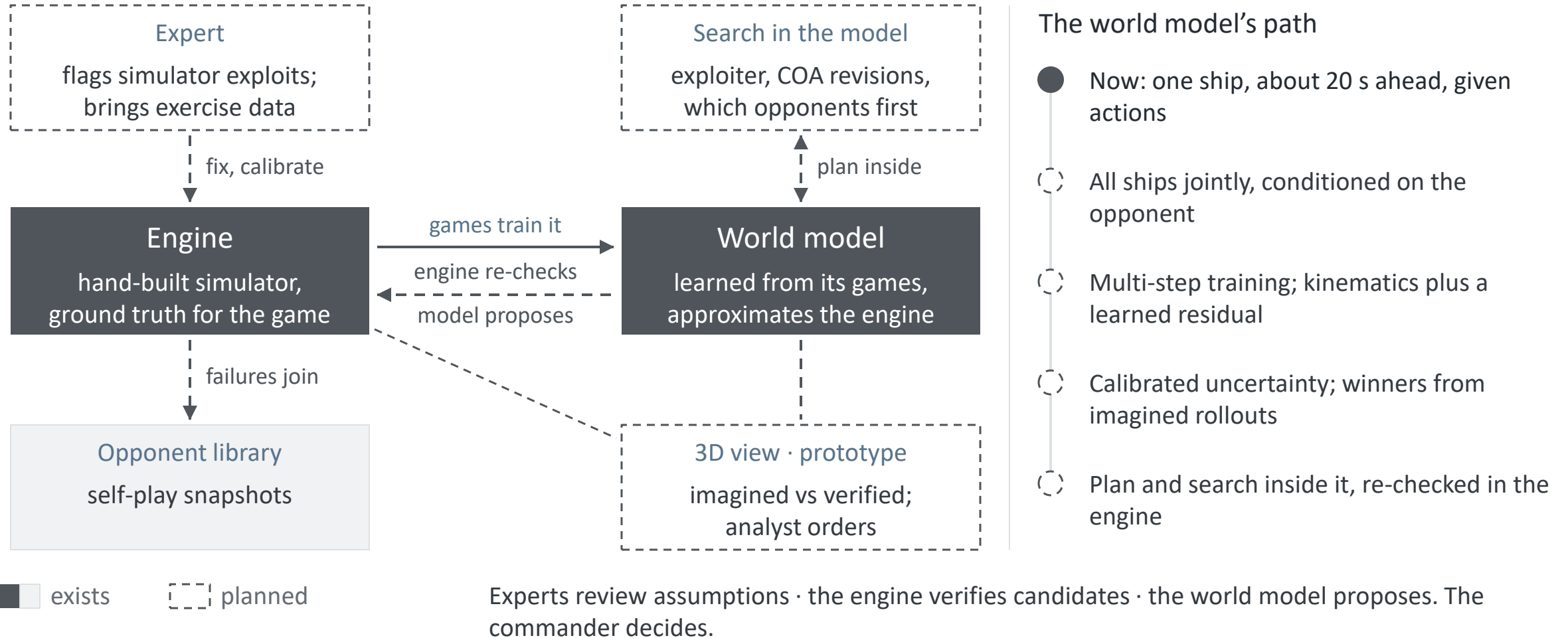
Swap the action stream fed to the model, and its projection moves: it is conditioned on what the ships do.

Limits one run's snapshots, in-distribution; doctrine-level effects untested

Next larger-scale simulation: more scenarios and opponents, a multi-ship joint model

Given the actions, it projects each ship 20 s ahead and separates winners by 75% of final length.

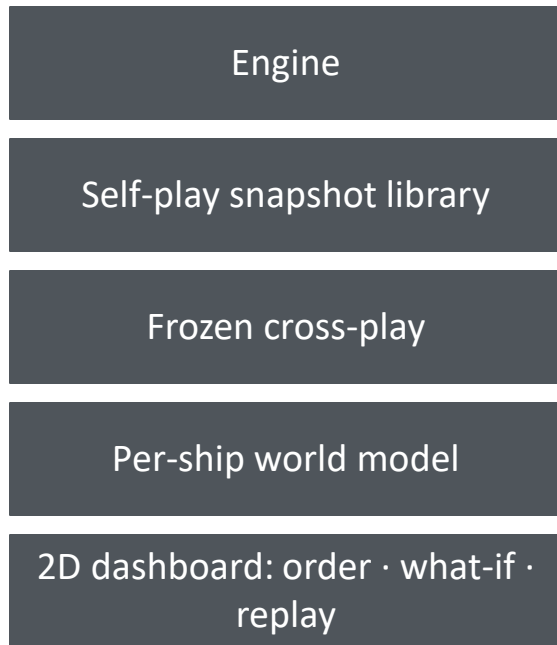
Next: from projection to a closed loop



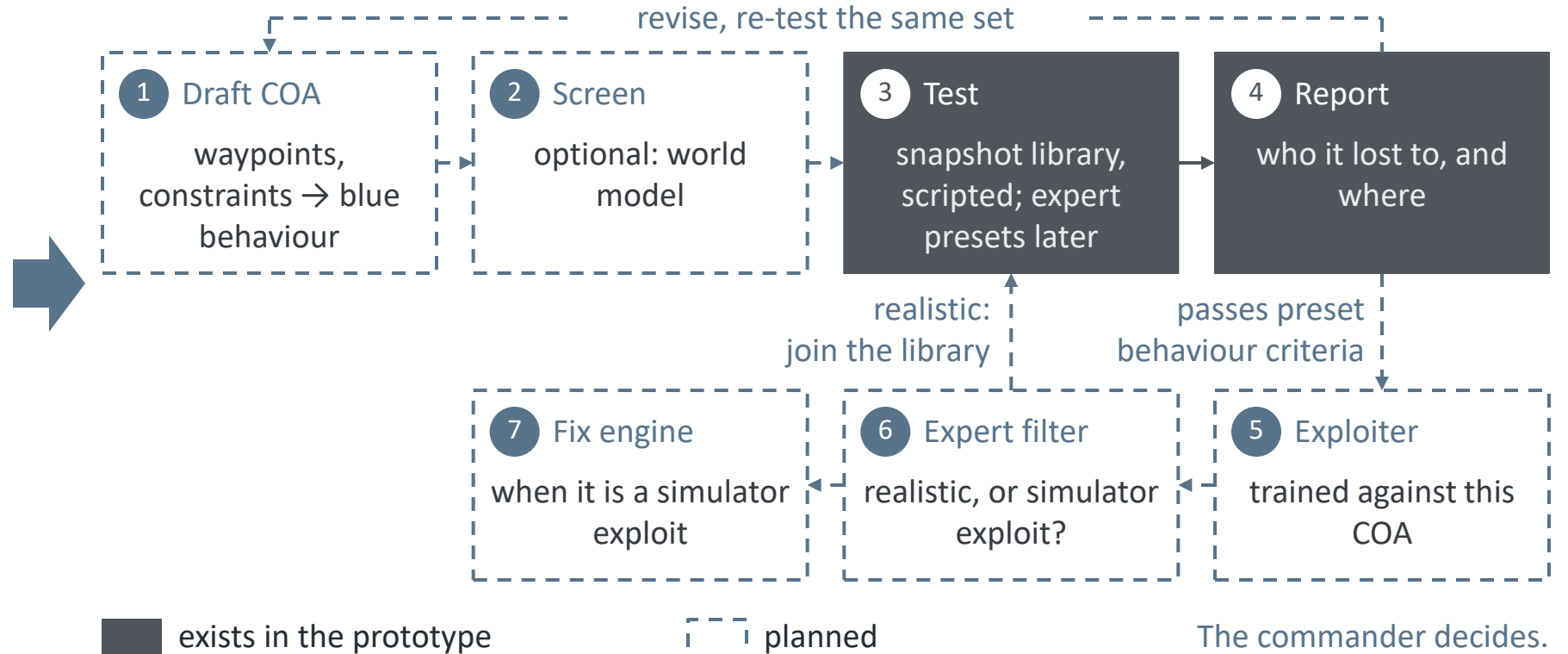
The aim: engine re-checks become training data, so the model improves where search pushes it.

Next: test a course of action, not a snapshot

Research prototype
this paper



Decision Copilot
in development at The Adversarial Lab



Not built yet: the COA compiler and the planned workflow. The dashboard what-if is a simulator clone, not the world model.

What our tests detect that the usual readouts miss

Usual readout

What our test detects

Training curve: 54%, balanced	→ 8 of 9	Frozen re-pairing is lopsided pairings won by Breakthrough
One seed: one overall winner	→ 3 seeds	Winners disagree; one rule compares 61% BT · 81% BL · 85% BT
Win label: Breakthrough won	→ 0 of 6	No iteration-34 win reaches the port each sinks the Blockade fleet instead
Stay-put or constant velocity	→ 146/146	World model beats both from 5 s windows, given actions; AUC 1.00 by 75%
Set or modelled opponent	→  	Both fleets learn so the opponent can always be changed

What's next



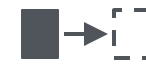
More simulation

more scenarios, opponents and seeds; gate fire on detection



Closed loop

world model proposes, engine re-checks; real-to-sim calibration



Prototype → product

Decision Copilot, in development: COAs against a growing library

Before trusting a training win, change the opponent.

Thank you. Questions welcome.