Boyun: Multi-Agent Simulation for Financial and Economic Decisions

Cherry Q

***Abstract*—Financial and economic decisions are shaped by actors with different goals, resources, and information, whose actions influence one another over time. Understanding these decisions requires explicit models of strategic choices, information exchange, and uncertainty. We present *Boyun*, a self-hosted, locally executed *agent-swarm* simulation framework, inspired by *MiroFish*, for exploring such decisions in specified worlds where modeled actors operate as autonomous *agents* communicating across a *social graph* as a *swarm of agents*. Its construction draws on *game theory* to define each agent's preference weights, feasible actions, and consequences; *social graphs* (implemented as *directed information graphs*) to govern how the agent swarm communicates and who receives messages; and *probability and statistics* to examine declared uncertainty and evaluate probabilistic claims. Agents in the swarm receive restricted observations, and a local agent decision model, *Laya*, returns structured choices and option scores on the user's own hardware. An execution engine applies the specified rules across strategic, market, and lending ledgers and records state transitions alongside social-graph message propagation. The resulting record supports inspection of observations, choices, and numerical consequences, and allows execution to be replayed from saved decisions without repeating inference. A local agent interface supports caller-authored worlds and offline experiments, and an optional blockchain commitment supports later verification of artifact integrity. Six synthetic experiments reproduce hand-calculated game, Bayesian, scoring, and accounting results and isolate individual mechanisms. Running locally with *Laya* at zero API fee, a fourteen-day, sixteen-agent swarm token-launch case completes all 24 policy paths with simulated token-price trajectories, settled trades, recorded rejections, matched branch comparisons, and exact replay, while a comparison with the original *MiroFish* on the same brief shows that the two systems are complementary: one produces richer social discourse, the other settled financial consequences. In addition, a ninety-day collateralized lending case across four worlds separates mechanical conclusions fixed by ledger rules from behavioral ones sensitive to uncalibrated agent scores. These results do not establish human behavioral fidelity, market forecasting accuracy, or improved decision quality; these claims require further empirical and comparative evaluation.**

***Index Terms*—Game theory, agent swarm, multi-agent simulation, agent-based modeling, social graphs, probability, statistical evaluation, scenario analysis.**

I. Introduction

Financial and economic decisions depend on how people and organizations respond to one another under uncertainty. A protocol designer evaluating an incentive must weigh its direct cost, whether partners join, and how other actors react, since those reactions alter the choices available later. Comparing alternatives therefore requires scenarios that make actors, constraints, and assumed events explicit while showing how their interactions produce different outcomes.

To that end, *agent-based modeling* represents a system through an interacting *swarm of agents*—autonomous decision-makers whose local observations and actions follow explicit interaction rules—so its conclusions depend on both the assumptions and their implementation [1], [2]. However, although language-model agents can condition choices on context [3], [4], an expressed choice alone does not specify its economic consequences. For example, a disclosure decision emitted by one agent in a swarm still requires rules defining which connected agents receive the message over a social graph, when they react, and what resource or state changes follow.

Along another dimension, social-simulation frameworks examine how an *agent swarm* interacts across a shared social graph. *OASIS* (Open Agent Social Interaction Simulations) provides one such platform, while *MiroFish* uses an agent-swarm social-simulation workflow to explore scenarios [4], [5]. Inspired directly by *MiroFish*'s paradigm of constructing worlds, orchestrating an interacting swarm of agents, and synthesizing scenario reports, *Boyun* pursues specialized financial and economic analysis, combining complementary methods to connect strategic incentives, social-graph communication across an agent swarm, and numerical consequences.

Building on that *MiroFish*-inspired agent-swarm foundation, we present *Boyun* (拨云, bō yún), a self-hosted *agent-swarm* framework for exploring decisions through explicit simulated worlds. Its Chinese name means "part the clouds" and draws on 拨云见日, in which clouds are parted to reveal the sun. In *Boyun*, market and protocol actors are modeled as autonomous *agents* that communicate with one another across a *social graph* as a *swarm of agents*. The name expresses the aim of bringing that agent swarm, its incentives, assumptions, social-graph relationships, and possible consequences into view before action, while keeping that view conditional on the evidence and assumptions used to construct it. Figure 1 summarizes the foundations and their operational roles.

First, to give this agent swarm an executable structure, *Boyun* places *game theory* at the center of world construction. Agents in the swarm have preference weights, resources, information, and feasible actions, and the world's rules determine how their combined choices affect outcomes [6], [7]. This structure makes incentives, coordination, and conflicts of interest explicit without assuming that model-selected actions are optimal or form an equilibrium. The distinction matters whenever agents value different outcomes: a founder agent may prioritize participation, whereas a partner agent may prioritize credibility or retained capital. Such differences must

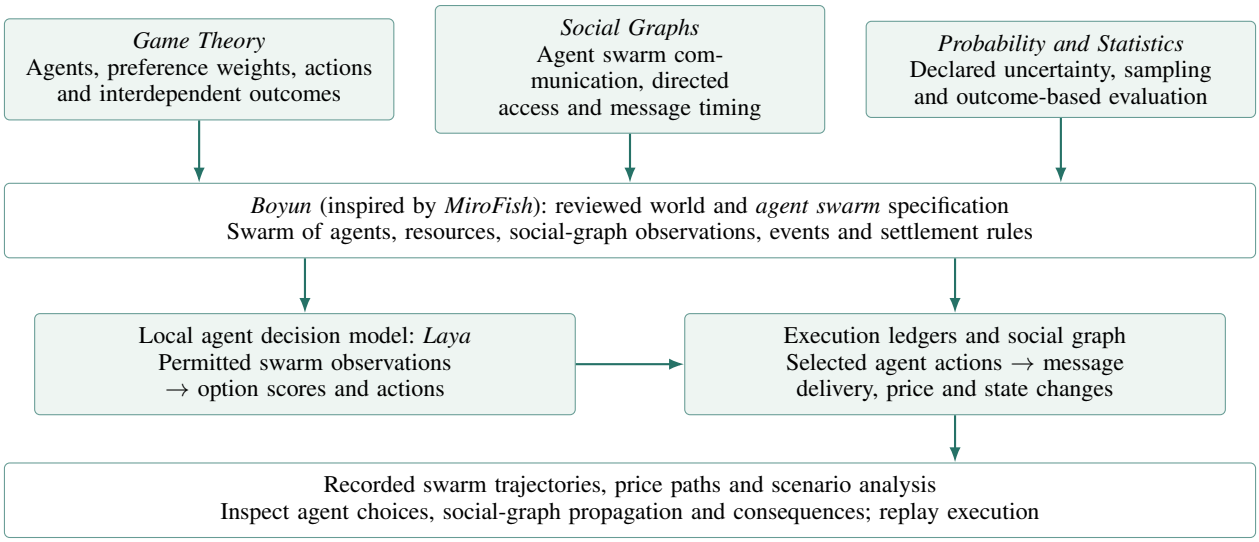


Fig. 1. Foundations of *Boyun*, inspired by *MiroFish*. The upper row identifies conceptual foundations; the lower layers show how modeled actors operate as a *swarm of agents* communicating over a *social graph* while self-hosted *Laya* supplies live agent decisions locally and deterministic ledgers apply the specified rules. Probability and statistical tools are separate analytical components, not automatic calibration of the agent decision model. Probability calculations use declared inputs; neither the diagram nor replay implies calibrated predictions or empirically validated social influence.

alter feasible actions and their consequences rather than remain descriptive labels in a persona.

On the other hand, incentives alone do not specify who knows what when inside the swarm. *Boyun* therefore connects the *swarm of agents* through a *social graph*—formalized as a *directed information graph*—that specifies which agents receive another agent's messages and how signals propagate across the swarm. Message availability and timing thus become explicit parts of the world rather than hidden assumptions. Even so, these social-graph links describe permitted delivery rather than empirically measured social influence, allowing an analyst to test how a disclosure, delayed message, or restricted channel changes what each agent in the swarm observes without giving every agent the author's complete view.

Apart from strategic incentives and social-graph communication, events and outcomes often remain uncertain. *Probability and statistics* provide a third foundation: *Bayesian* calculations update supplied beliefs, seeded sampling explores declared alternatives, and proper scoring rules evaluate probabilistic predictions [8]–[10]. Nevertheless, these tools expose uncertainty only under stated assumptions; neither repeated simulation nor model confidence establishes real-world odds.

Once those three foundations define a world, the engine must obtain choices across the *agent swarm* while keeping local inference and recorded execution explicit. Because unpublished launch plans, treasury limits, and risk parameters are sensitive, *Boyun* focuses on private, self-hosted execution using an open-weight *agent decision model* that runs locally on the user's hardware. Each agent in the swarm receives only its permitted social-graph and ledger observation, and *Laya* [11]—the local agent model used in the reported case studies—returns a structured choice and normalized *option scores* over the agent's *affordable action menu*. Alternative open-weight models are a future evaluation direction rather than a capability demonstrated by these runs. Crucially, option scores express model-assigned weights rather than validated real-world probabilities. Deterministic strategic, market, and lending ledgers then settle the chosen actions, route messages across the social graph, enforce conservation, and save every observation and choice so runs replay exactly without repeating inference, while leaving the model's internal deliberation unobserved.

Moreover, self-hosted inference addresses a practical budget constraint. Evaluating an *agent swarm* across many decision points, social-graph interactions, and matched policy branches requires tens or hundreds of thousands of model queries, which quickly exhaust small hosted-API budgets while sending private scenario states off the user's machine. Running *Laya* locally eliminates per-call API fees and keeps all swarm observations on-device, though operational efficiency and behavioral accuracy remain separate evaluation questions.

Given these capabilities and boundaries, the paper asks: *Can a self-hosted, explicitly specified agent-swarm world constrain social-graph information, spending, and settlement while making the effects of alternative information and interaction rules inspectable?* Applications include capital allocation, participation incentives, disclosure policies, and credit stress scenarios in decentralized finance. Reports connect financial outcomes and simulated price paths with coordination, social-graph message delivery, and swarm reactions to events so researchers, protocol designers, and risk analysts can trace how an outcome arises under declared assumptions.

Accordingly, we evaluate *Boyun* through six synthetic mechanism experiments, a fourteen-day sixteen-agent swarm token-launch case across 24 policy paths and a ninety-day twenty-

agent swarm collateralized lending case across four worlds—all executed locally with *Laya*—together with a matched-brief comparison against the original *MiroFish* that inspired *Boyun*'s workflow. These tests do not establish human behavioral fidelity, forecasting accuracy, or improved user decisions. Within that scope, the paper makes five contributions:

1) an explicit specification of worlds, the *swarm of agents*, and what each agent may observe over the *social graph* (Section III);
2) an execution procedure across strategic, market, and lending ledgers that records every agent decision and replays it exactly (Section IV);
3) reproducible mechanism tests with known answers (Sections V–VI);
4) a matched case comparison with the original *MiroFish* (Section VI-C);
5) full-horizon local *Laya* token-launch and lending case studies —including simulated token-price trajectories across 24 worlds—that separate mechanical conclusions fixed by ledger rules from behavioral ones sensitive to the agent decision model (Sections VI-B and VI-E).

The following section situates these contributions within prior work.

II. Related Work

The Introduction linked strategic incentives, message delivery, and uncertainty within a self-hosted simulated world. To situate that design, this section first examines how agent-based models are specified, and then reviews game-theoretic and financial environments, social simulation, and probabilistic decision evaluation. Throughout, comparisons concern modeling scope and preserved evidence rather than a performance ranking of systems.

A. Agent-Based Simulation and Model Specification

To make interacting actors a reproducible object of study, agent-based research requires an explicit description of the environment and its rules. In particular, the Overview, Design concepts, and Details (*ODD*) protocol standardizes descriptions of entities, initialization, schedules, inputs, and submodels [1]. Complementing specification standards, Galán et al. distinguish conceptual modeling assumptions from artifacts introduced by software implementation [2]. Together, these works motivate separating verification—whether code executes stated rules accurately—from validation of whether those rules represent real markets.

Building on that distinction, *Boyun* specifies actors, resources, observation boundaries, actions, and settlement rules before a run begins. Preserving these inputs opens assumptions to review and enables exact replay checks. However, an explicit specification alone cannot show that assumed incentives or actor choices are realistic, which requires examining both the economic environment and the behavioral model.

B. Game-Theoretic and Economic Environments

On the strategic side, *game theory* links actors' choices to outcomes that depend on what others do. *Nash equilibrium* provides a classic benchmark in which no actor gains from a unilateral deviation [7], and *OpenSpiel* operationalizes finite games through explicit states, legal actions, observations, transitions, and returns [12]. *Boyun* adopts explicit game specification to construct decision worlds; by contrast, it neither incorporates *OpenSpiel*'s equilibrium solvers nor assumes that its actor decision model plays an equilibrium.

Beyond abstract games, economic simulation environments enforce incentives through allocation and accounting rules. For instance, the *AI Economist* couples economic agents with a tax-policy planner, separating individual utility from a designer's objective [13], while *EconAgent* places language-model agents inside numerical macroeconomic rules for labor and consumption [14]. Similarly, *Boyun* separates actor preference weights, action selection, and engine-computed consequences, though it targets bounded financial and protocol decisions rather than macroeconomic policy learning.

When questions turn specifically to trading and decentralized credit, settlement rules must reflect venue mechanics. *ABIDES* (Agent-Based Interactive Discrete Event Simulation) models trading agents communicating with an order-book exchange over configurable network latency [15]. On the other hand, decentralized exchanges commonly replace order books with *automated market makers* (AMMs), whose reserve formulas govern pricing, fees, and slippage [16], [17], while decentralized lending pools price credit through utilization-based interest curves and collateral liquidations [18]. These precedents show why a narrative choice or participation score cannot substitute for a settled swap, loan, or cash balance; *Boyun* therefore pairs its ledgers with an explicit social-information layer.

C. Social Graphs and Agent-Swarm Simulation

Turning to that social layer, language-model simulations study how an *agent swarm* observes and reacts to connected peers across a *social graph*. *Generative Agents* combines memory, reflection, and planning to produce contextual behavior [3], whereas *OASIS* provides large-scale social-platform environments with posting actions and recommendation feeds [4]. Both frameworks motivate treating observation access and interaction history as explicit parts of the world rather than assuming shared global knowledge across the swarm.

Directly inspiring *Boyun*, *MiroFish* pioneered an end-to-end workflow from world construction to agent-swarm simulation and structured reporting [5]. However, whereas *MiroFish* runs open-ended social discourse on *OASIS*, *Boyun* models actors as a *swarm of agents* communicating over an explicit *social graph* (implemented as a directed information graph) while executing bounded actions over deterministic strategic, market, and lending ledgers. Section VI-C compares the two systems on the same launch brief, documenting their complementary outputs rather than claiming a performance ranking.

Even when social and financial rules are explicit, plausible prose or contextual responsiveness does not guarantee incentive-consistent choices. Controlled repeated-game studies with language models test how models respond to explicit payoffs and opponent histories [19]. Accordingly, evaluating *Boyun* requires separating rule-enforced consequences from model-selected behavior and distinguishing option scores from outcome probabilities.

D. Probabilistic Decisions and Statistical Evaluation

To obtain structured choices without free-text parsing, typed decision models score a finite menu of options given a state [20], [21]. Because financial scenarios often contain confidential strategy, open-weight implementations are particularly relevant: *Laya* fine-tunes a 421-million-parameter encoder to return normalized option scores on a local CPU [11], and *Kev* trains compatible open-weight variants on Qwen backbones [22], whereas hosted services such as *Jev* process requests remotely [21]. *Boyun* uses local *Laya* inference in its experiments while keeping feasibility checks and accounting in code. At the same time, *Laya*'s model card notes that its training workflows exclude financial markets and that its confidence values are uncalibrated [11], so returned option scores must never be read as empirical market odds.

By contrast, reasoning about external event uncertainty requires explicit hypotheses, priors, and likelihoods via *Bayesian updating* [8]. Repeated sampling can explore consequences under declared priors, yet it cannot supply missing likelihoods or correct a biased behavioral model. Ultimately, any probabilistic forecast claim must be evaluated against resolved outcomes using strictly proper scoring rules such as the *Brier* and *logarithmic scores* [9], [10].

Taken together, prior work motivates combining explicit game incentives, directed message delivery, self-hosted typed actor decisions, and ledger-based settlement, while maintaining a strict boundary between verified mechanics and unvalidated market forecasts. The next section formalizes how a world is constructed within that boundary.

III. Problem Formulation and World Creation

The preceding section identified the conceptual foundations needed to study strategic interactions. To turn those foundations into a reproducible experiment, *Boyun*—following *MiroFish*'s world-first workflow—constructs a *world*: an explicit specification of an *agent swarm* (modeled actors operating as autonomous agents), resources, a *social graph* governing observations and communication, actions, events, and settlement rules. Construction is performed by a user or a separate authoring model, followed by structural checks and human review, and finishes before the agent decision model is ever called. Figure 2 summarizes this preparation; the execution procedure follows in Section IV.

A. Core Terminology

Before specifying a world, a reader must distinguish the static specification from the record produced by executing it. Table I defines worlds and their execution, while actor and decision terms appear in the subsections below. In particular, a world is a model of a decision setting, whereas a run is one execution of that model; two runs of the same world can differ whenever the selection rule uses sampling. For an everyday analogy, a board game's rules and starting setup define the world, whereas one played game is a run and its recorded move sequence is a trajectory.

TABLE I
WORLDS AND EXECUTION TERMINOLOGY.

Term	Definition
World	A specified decision setting: actors, resources, observations, actions, events, and settlement rules.
Scenario	An alternative setting examined through a world or matched branch; its label does not establish a probability.
State	Ledger-specific data at step t: balances and metrics, plus reserves, collateral, debt, or claims where modeled.
Run / trajectory	A run executes one world. Its trajectory records observations, actions, and settled states.
Round / horizon	Index t orders rounds or decision points up to T; elapsed time is recorded separately. A cycle is not inherently an hour or day.
Outcome / metric	An outcome is a simulated result. A metric $x_{k,t}$ describes shared state within bounds $[L_k, H_k]$; metric points are not money.

Once those terms are separated, the author can state what the simulation is designed to compare. Consequently, world creation begins with the decision question rather than with narrative persona writing.

B. Decision Question, Alternatives, and Horizon

Specifically, the author first defines the focal decision, the alternatives under comparison, the observation horizon T, and the outcomes of interest. For example, a protocol designer might compare early reserve disclosure against delayed disclosure and track partner participation alongside retained cash. That question dictates which actors and ledgers belong in the world; adding unrelated detail does not strengthen the comparison.

From that focal question, the author defines a reference world and isolates which inputs change in each matched branch. Comparing disclosure policies, for instance, keeps initial balances and unrelated rules fixed wherever possible; if several inputs change together, the resulting difference reflects their joint intervention rather than a single cause. Likewise, the horizon T bounds only the observation window: reaching T does not imply that the modeled protocol or market has ended.

C. Agent Swarm, Preference Weights, and Strategic Dependence

With the comparison fixed, agents enter the *swarm* (the world's cast) only if they control relevant resources, hold

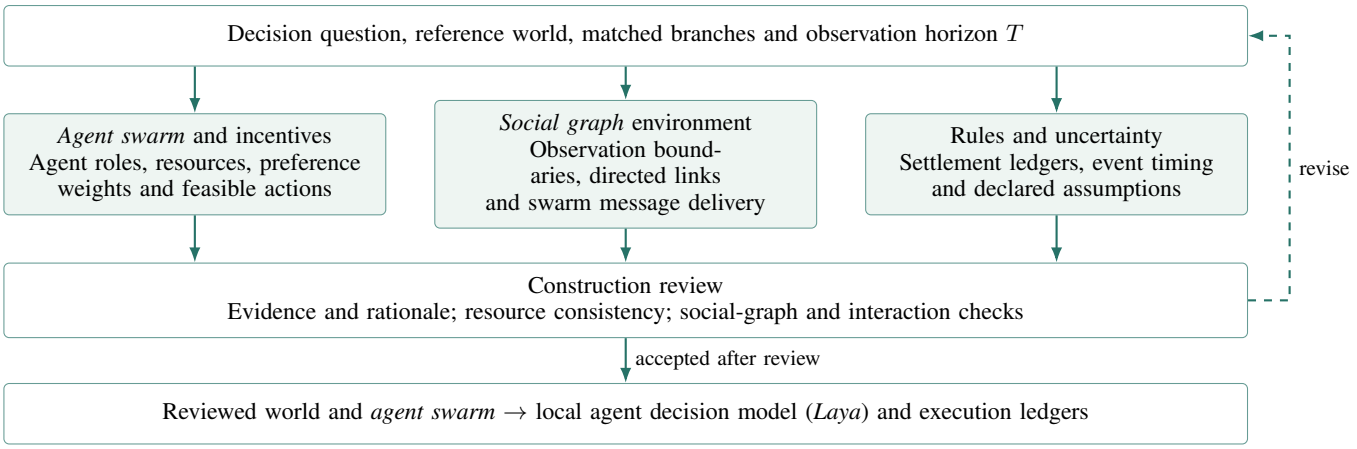


Fig. 2. World and *agent swarm* construction in *Boyun*, inspired by *MiroFish*'s world-building workflow. A decision question determines three complementary specification tasks defining the swarm of agents, their social graph, and the settlement rules. Their outputs undergo structural checks and human review, with author-directed revision where needed. The process specifies agent preference weights, social-graph links, and settlement rules before *Laya* selects live agent actions locally.

distinct information, or can alter an outcome that another agent values. An *actor* in *Boyun* is an autonomous modeled *agent*, while a *persona* describes its role and motivation. To make that persona executable within the *swarm of agents*, the world assigns explicit goals, initial balances, feasible actions, social-graph observation access, and numeric preference weights, giving the agent swarm a game-theoretic structure rather than a list of biographies [6], [7]. Table II separates these elements.

TABLE II
AGENT SWARM, GOALS, AND INCENTIVES.

Term	Definition
Actor / agent	An autonomous modeled decision-maker indexed by i (a player in game theory); the interacting collection of agents forms the *agent swarm* (or *cast*).
Persona	The agent's descriptive role and motivation, paired with explicit preference weights and actions.
Goal	The outcome the agent seeks, distinct from the analyst's research question.
Incentive	A cost, benefit, or conditional effect that makes an action more or less attractive under the agent's preference weights.
Preference weights / utility	In the strategic ledger, declared weights $w_{i,\text{cash}}, w_{i,k}$ map cash and metrics to utility $U_{i,t}$; utility need not equal profit.
Resource / constraint	An available balance (such as cash $b_{i,t}$) and an enforced limit excluding unaffordable actions.

For instance, consider a partner deciding whether to fund a pilot. The partner is an actor whose goal is preserving capital while gaining credible participation; its preference weights quantify that trade-off, whereas action costs and credibility gains create incentives. Funding and waiting are actions, and selecting one from the actor's observation is a decision whose consequences the ledger settles. This example illustrates how the terms connect rather than claiming empirical partner behavior.

Moreover, within the same cast, actors can assign different preference weights to retained cash, participation, credibility, or a declared risk metric. The author must state what each metric represents and why its weight reflects an assumed trade-off, because calling a persona "cautious" in prose does not enforce caution in settlement. In everyday terms, *utility* resembles commuters ranking routes: one commuter weights a lower fare more heavily, whereas another weights a shorter trip, so different choices emerge without assuming everyone minimizes fare.

Beyond individual preference weights, strategic dependence requires rules that couple actors' choices. Supporting a launch may pay off only if an independent reviewer verifies it, or two simultaneous actions may shift a shared metric more than the sum of their individual effects. However, a promise or threat in persona text has no binding force unless an executable rule enforces it. Similarly, splitting one actor cohort into multiple accounts requires partitioning its balances and per-action effects rather than copying its full market power into each new account.

D. Social Graph, Information, Rules, and Event Timing

Alongside incentives, each agent in the swarm operates behind an explicit observation boundary defined by a *social graph*. Directed edges in the *social graph* (implemented as an *information graph*) determine how agents in the swarm communicate and whose messages an agent receives, while the engine withholds other agents' private preference weights and unannounced future events. In the strategic ledger, messages attached to round-t actions arrive in round $t + 1$, ensuring that query order within a round never grants an unintended information advantage. By analogy, an *observation* is a card player's view of their own hand and public table cards,

whereas the swarm's *social graph* works like mailing-list subscriptions, which need neither be mutual nor imply trust. Table III summarizes these terms.

TABLE III
DECISIONS, SOCIAL-GRAPH OBSERVATIONS, AND RULES.

Term	Definition
Action	A permitted option $a \in \mathrm{Menu}_i$ with declared cost $c_i(a)$, effects, and optional preset message; `wait` is an action.
Decision	Selection of an affordable action $a_{i,t} \in A_{i,t}$ from agent i's observation; the ledger settles consequences separately.
Observation	Information visible to one agent at t: its own profile and balances, public state, visible events, and delivered social-graph messages.
Social / information graph	Directed communication network through which the *swarm of agents* exchanges messages, specifying permitted delivery rather than measuring social influence.
Event	An exogenous change with declared timing, audience visibility, and state effects.
Rule / interaction	A settlement law; in the strategic ledger, a joint-action rule r adds effect $g_{k,r}$ when a specified action combination occurs.

To settle choices consistently, the world also declares action costs, metric bounds $[L_k, H_k]$, ledger parameters, and exogenous event schedules. Default strategic worlds apply deterministic event schedules, whereas probabilistic event trees belong to a separate event-chain module. Keeping exogenous events distinct from endogenous actor actions lets later analysis trace whether a trajectory shift came from an external shock or from cast reactions.

E. Evidence, Assumptions, and Uncertainty

Because settlement rules fix the meaning of every simulated number, each material parameter must trace back to cited evidence or a labeled assumption. An *assumption* is an unverified modeling input, whereas *evidence* is a dated external record motivating part of the world. For example, a parliamentary agenda may establish the date of a debate without establishing how traders will react; world construction therefore records the dated source and the assumed behavioral response separately, following established model-documentation practice [1].

On the other hand, whenever a scenario introduces numerical probabilities, the author must distinguish a user-chosen scenario branch, a model-returned option score $q_{i,t}(a)$, a seeded sampling draw, and an empirical frequency. In particular, *Bayesian* updating requires declared hypotheses, priors, and likelihoods [8] rather than inferring odds from narrative text; uncalibrated scenario weights are therefore reported strictly as assumptions.

F. Construction Review and Output

Before any run starts, automated checks and human review audit the assembled world. Specifically, construction checks verify balance conservation, metric units, affordable action menus, isolated unilateral deltas, information links, and joint-action couplings, flagging degenerate setups such as actors with only one affordable option or goals unaffected by any permitted action.

Consequently, passing construction review guarantees an explicit, executable specification rather than an empirically calibrated population; human judgment remains responsible for checking whether preference weights match intended roles and whether cited sources justify the parameters. With that specification frozen, Section IV details how *Boyun* executes rounds and decision points locally.

IV. Simulation Method and System Design

Section III ended with a reviewed world specification; this section explains how *Boyun* executes it across a *swarm of agents*. Returning to the board-game analogy, world creation fixes the rules and opening balances, while execution plays the game one step at a time. At each step t, the engine shows every agent i in the *agent swarm* only its permitted observation along the *social graph*, queries the local actor decision model (*Laya*) over the actions that agent i can afford, and settles all selected actions on a deterministic ledger. Consequently, the model selects moves for each agent in the swarm but can never alter permissions or accounting. Figure 3 summarizes this pipeline. After fixing the execution terms (Section IV-A), we trace *agent-swarm* observation construction, action selection, *social-graph* message and event timing, and settlement across the strategic, market, and lending ledgers, followed by analytical modules, replay, and self-hosted privacy boundaries.

A. Execution Terminology

To keep terminology consistent across all three ledgers, Table IV defines the runtime concepts that complement Tables I–III. Discrete strategic worlds advance by *rounds*, whereas timestamped market and lending worlds advance by scheduled *decision points*; both use an ordered index $t \in \{1, \ldots, T\}$. The associated timestamp τ_t records elapsed physical time. Hours and days in the case results refer to τ_t, not to the decision index; finer launch steps therefore do not count as additional elapsed hours.

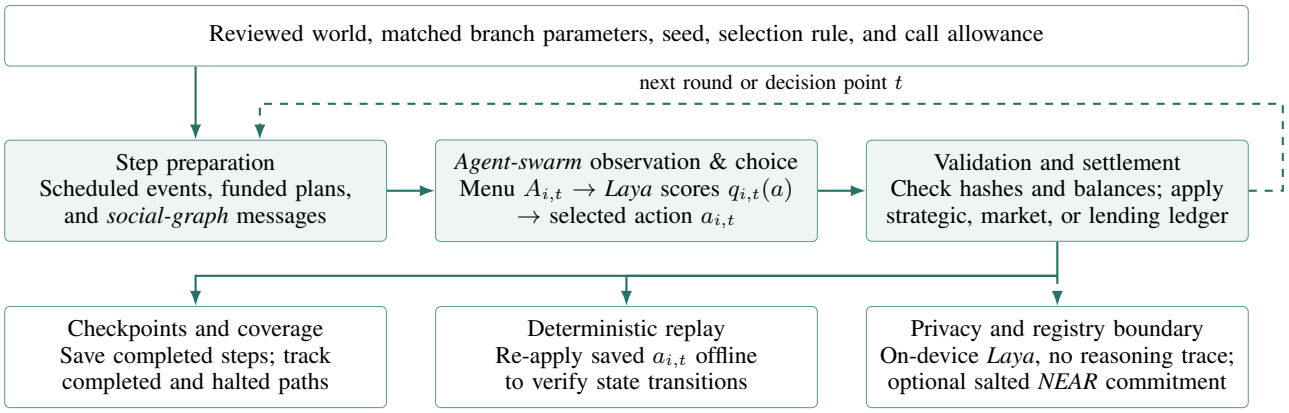


Fig. 3. Simulation pipeline and evidence boundaries in *Boyun*. At each round or decision point t, the engine prepares exogenous updates and *social-graph* deliveries, builds per-agent observations across the *swarm of agents*, obtains option scores $q_{i,t}(a)$ locally from *Laya* (or an explicit offline baseline), and settles transitions on the active ledger. Option scores $q_{i,t}(a)$, selected actions $a_{i,t}$, and settled states remain distinct in the saved record; dashed arrows denote continuation up to horizon T and optional on-chain commitment.

TABLE IV
EXECUTION TERMINOLOGY AND NOTATION.

Term	Definition
Affordable action menu $A_{i,t}$	Feasible options for actor i at step t. In the strategic ledger this filters the reviewed menu by cash $b_{i,t}$; market and lending menus use their own ledger constraints.
Option scores $q_{i,t}(a)$	Normalized weights in $[0,1]$ summing to 1 returned by the actor decision model over $A_{i,t}$; not empirical odds.
Selection rule	Pre-declared rule (*argmax* or *sampling*) mapping $q_{i,t}(a)$ to one action $a_{i,t}$.
Settlement	Atomic state transition applying chosen actions on the strategic, market, or lending ledger.
Matched branch	Counterfactual run sharing initial state, seed, and schedule up to an intervention while varying one policy.
Checkpoint / replay	Saved step state and offline re-execution of saved $a_{i,t}$ without calling the model to verify exact state agreement.

Among these concepts, deterministic replay is central to verification: much like replaying a recorded chess game from algebraic notation, replay re-applies saved moves without re-querying the player, so any mismatch isolates a rule or accounting defect.

B. Agent-Swarm Observations and Affordable Actions

Executing step t begins by restricting what each agent i in the *swarm of agents* can see. If every agent observed rivals' private preference weights and future events, choices would reflect the world author's omniscience rather than local incentives across the *social graph*. For each agent i, the engine constructs a canonical JSON *observation* containing its persona, goal, preference weights, and pre-decision cash $b_{i,t}$; visible public state (such as shared metrics $x_{k,t}$ indexed by k, pool reserves, or oracle prices); delivered *social-graph* messages and visible events; and receipts from agent i's own earlier actions, together with a SHA-256 observation hash. At the same time, withholding private fields enforces an information boundary inside the *agent swarm* rather than encrypting memory on the host machine.

Restricting information, however, does not by itself prevent overspending. In the strategic ledger, before querying *Laya*, code filters actor i's reviewed menu Menu_i against current cash $b_{i,t}$ to form the *affordable action menu*:

$$A_{i,t} = \{a \in \mathrm{Menu}_i : c_i(a) \leq b_{i,t}\}, \tag{1}$$

where $c_i(a)$ is the integer cash cost of action a. Enforcing Eq. (1) in code guarantees that affordability depends on ledger balances rather than on how the model reads narrative text.

C. Action Selection

Once $A_{i,t}$ is fixed, the engine selects exactly one action $a_{i,t}$. In the strategic ledger, whenever only `wait` is affordable ($|A_{i,t}| = 1$), no choice exists, so the engine records `wait` without a model query. Otherwise, it passes the hashed observation and a typed `Choice` prompt to *Laya* on the local CPU [11]. Other open-weight backends are a future evaluation direction, not an inference result established here. In strategic worlds, each option is paired with a code-computed unilateral projection of immediate post-action cash, clamped metrics, and utility delta holding peer actions fixed, so arithmetic is supplied by code rather than delegated to the model.

Because *Laya* outputs option scores rather than state mutations, the adapter validates every response before selection. Specifically, the response must assign a finite score $q_{i,t}(a) \in [0,1]$ to each $a \in A_{i,t}$ with $\sum_{a \in A_{i,t}} q_{i,t}(a) = 1$, identify a top-scoring action, and include the model's confidence field; malformed responses halt the step without partial settlement (allowing only a ≤ 0.001 rescaling tolerance for *Laya*'s four-decimal rounding).

Given a validated distribution, the pre-declared *selection rule* maps $q_{i,t}(a)$ to $a_{i,t}$ via either *argmax* or seeded *sampling*:

$$\text{argmax:} \quad a_{i,t} = \arg\max_{a \in A_{i,t}} q_{i,t}(a), \tag{2}$$

$$\text{sampling:} \quad a_{i,t} = a_{(m^*)}, \text{ where}$$
$$m^* = \min\left\{m : \textstyle\sum_{\nu=1}^{m} q_{i,t}\big(a_{(\nu)}\big) > \xi_{i,t}\right\}. \tag{3}$$

Here $a_{(1)}, a_{(2)}, \ldots$ order $A_{i,t}$ canonically, ν indexes the cumulative sum, and $\xi_{i,t} \in [0, 1)$ is a deterministic pseudorandom draw keyed to the run seed and decision identity. Under Eq. (3), an action with score 0.30 is drawn for 30% of $\xi_{i,t}$ values while remaining exactly reproducible across replays. Nevertheless, neither selection rule turns $q_{i,t}(a)$ into an empirically calibrated probability or a solved mixed-strategy Nash equilibrium [7], [23].

D. Social-Graph Messages, Swarm Memory, and Event Timing

Beyond changing balances, $a_{i,t}$ can transmit information through the *social graph* to other agents in the *swarm of agents* at later steps. In the strategic ledger, a preset message attached to $a_{i,t}$ follows the *social graph* (*directed information graph*) and arrives at subscribed agents at the start of round $t+1$; in timestamped ledgers, messages obey declared delivery delays and recipient lists across the swarm. Delaying delivery ensures that within-step evaluation order never creates an artificial first-mover leak. Observations also retain bounded memory of agent i's own prior actions and public state history without exposing peers' private balances, while delivered *social-graph* messages remain unverified claims unless backed by executable rules [23], [24].

On the other side, exogenous events and pre-committed plans enter on explicit schedules separate from actor reactions. Strategic rounds apply scheduled events before building round-t observations, whereas timestamped ledgers record both occurrence time and availability time to represent reporting lag. Furthermore, when a scenario tests a funded launch or marketing plan, committed actions execute and deduct payer cash at their scheduled decision point before outside actors observe the result and choose $a_{i,t}$.

E. Strategic Ledger Settlement and Accounting

After all actors choose $a_{i,t}$ from the same frozen state, the strategic ledger applies *synchronous settlement* so actor ordering cannot distort bounded metrics. First, the engine reverifies every observation hash and Eq. (1); if any check fails, the round aborts without mutating state. Otherwise, cash balances and shared metrics update atomically:

$$b^{+}_{i,t} = b_{i,t} - c_i(a_{i,t}), \tag{4}$$

$$x^{+}_{k,t} = \mathrm{clip}_{[L_k,\, H_k]}\left(x_{k,t} + \sum_i e_k(a_{i,t}) + \sum_{r \in R_t} g_{k,r}\right), \tag{5}$$

where $e_k(a)$ is action a's effect on metric k, R_t is the set of joint-action rules triggered at round t, $g_{k,r}$ is rule r's effect on metric k, and $\mathrm{clip}_{[L,H]}(z) = \min(H, \max(L, z))$ enforces bounds once after summing all effects. The superscript $+$ denotes the post-settlement state of this round; scheduled events for the next round are applied afterward to construct its observation state. Algorithm 1 summarizes the strategic-round loop and offline replay. Timestamped ledgers have their own execution procedures; this algorithm is not a shared implementation for all three ledgers.

Algorithm 1 Strategic-round execution, synchronous settlement, and replay

Require: Reviewed strategic world, selection rule, seed, call allowance
Validate cast, balances, bounds, events, and information graph
for each strategic round $t = 1, \ldots, T$ **do**
  Apply scheduled events and funded commitments; deliver prior messages
  Freeze pre-decision state and build hashed observations
  for each actor i **do**
    Form $A_{i,t}$ by Eq. (1)
    if $|A_{i,t}| = 1$ **then**
      Set $a_{i,t} \leftarrow$ `wait` without querying *Laya*
    else
      Query local *Laya* (or offline baseline) within call allowance
      Validate $q_{i,t}(a)$; select $a_{i,t}$ via Eq. (2) or (3)
    end if
  end for
  if any observation hash or action $a_{i,t}$ is invalid **then**
    Reject step t without mutating state; **stop**
  end if
  Settle the strategic state; save receipts and checkpoint
end for
Re-apply saved $a_{i,t}$ offline without *Laya*; verify identical states

To evaluate settled strategic states across heterogeneous roles, actor i's utility combines retained cash and metrics via its declared preference weights:

$$U_{i,t} = w_{i,\mathrm{cash}}\, b_{i,t} + \sum_k w_{i,k}\, x_{k,t}. \tag{6}$$

Here $U_{i,t}$ is the strategic ledger's preference measure, not dollar profit. Weights carry the units needed to convert cash and metric units into the chosen utility scale. This linear formula is not an estimated investor utility function and does not define a common utility calculation for the market and lending ledgers. In addition, because Eq. (1) bounds $c_i(a_{i,t}) \le b_{i,t}$ and Eq. (4) adds each integer deduction to cumulative cast spending S^{+}_{t} through settlement of round t, every settled round satisfies exact cash conservation:

$$b^{+}_{i,t} \ge 0 \quad \text{and} \quad \sum_i b^{+}_{i,t} + S^{+}_{t} = \sum_i b_{i,0}. \tag{7}$$

F. Market Ledger

Whereas the strategic ledger treats cumulative spending S^{+}_{t} as effort leaving the cast, market scenarios require counterparty fills, reserves, and prices. The market ledger tracks integer cash and tokens and settles swaps using a simplified constant-product automated market maker (AMM) [16], [17]. A swap's output depends on reserves, the configured fee, and integer rounding; pre-migration phases use specified virtual reserves [25]. Orders outside slippage or inventory limits are rejected atomically with a recorded reason. This execution model is a stated approximation, not a complete implementation of a

live venue. Reports distinguish realized cash from the quoted value of unsold inventory.

G. Lending Ledger

Complementing spot trading, protocol credit decisions require tracking deposits, collateralized debt, and liquidations on a separate *lending ledger*. *Suppliers* deposit stablecoins into a pool with idle cash C_t and lent principal P_t, holding claims c_s (withdrawals exceeding C_t wait in a first-in, first-out queue ahead of new loans), while *borrowers* pledge volatile collateral against loans indexed by ℓ. With pool *utilization* $u_t = P_t/(C_t+P_t)$, the annual borrow rate follows the standard kinked rate model [18]:

$$r(u_t) = \begin{cases} r_0 + (r^* - r_0)\, u_t/u^*, & u_t \le u^*, \\ r^* + (r_1 - r^*)\,(u_t - u^*)/(1-u^*), & u_t > u^*, \end{cases} \tag{8}$$

where r_0 is the base rate, u^* is the kink utilization, r^* is the rate at u^*, and r_1 is the full-utilization rate. Over Δt seconds (Y seconds per 365-day year), loan ℓ incurs simple interest $I_\ell = P_\ell\, r(u_t)\, \Delta t/Y$, accruing to the protocol (share φ) and each supplier s:

$$\Delta I_\ell^{\text{prot}} = \big\lfloor \varphi\, I_\ell \big\rfloor_\epsilon, \qquad \Delta I_\ell^{s} = \left\lfloor (1-\varphi)\, I_\ell\, \frac{c_s}{\sum_{s'} c_{s'}} \right\rfloor_\epsilon, \tag{9}$$

rounded down to $\epsilon = 10^{-12}$ stablecoin units. Specifically, this fixed grid bounds arithmetic precision over repeated hourly accruals while keeping interest allocation within the declared rounding rules.

Meanwhile, solvency is governed by each loan ℓ's *health factor* [26]:

$$h_\ell = \frac{Q_\ell\, p_t\, \theta}{D_\ell\, p_t^{\text{usd}}}, \tag{10}$$

where Q_ℓ is pledged collateral at oracle price p_t, θ is the liquidation threshold, and D_ℓ is principal plus accrued interest at stablecoin price p_t^{usd}. When $h_\ell < 1$, a liquidator repays debt (partially, or fully once $h_\ell \le 0.95$) for collateral worth $1+\beta$ times repayment, executed only if net market proceeds cover repayment and costs; residual underwater debt writes off against a capped loss reserve, and stale oracle timestamps halt borrowing and liquidations.

H. Analytical Modules

Apart from time-stepped ledgers, *Boyun* provides three standalone analytical modules that operate strictly on user-supplied inputs. First, for small normal-form games (≤ 4 actors and ≤ 256 profiles) with exact payoffs $U_i(a_i, a_{-i})$ and supplied opponent beliefs $\pi_i(a_{-i})$, code computes expected utility, regret, and pure-strategy Nash equilibria [7]:

$$\text{EU}_i(a_i) = \textstyle\sum_{a_{-i}} \pi_i(a_{-i})\, U_i(a_i, a_{-i}), \tag{11}$$

$$\text{Reg}_i(a_i) = \max_{a'} \text{EU}_i(a') - \text{EU}_i(a_i). \tag{12}$$

Second, for finite hypothesis sets H with supplied priors $P(h)$ and signal likelihoods $P(E \mid h)$, *Bayesian updating* computes [8]

$$P(h \mid E) = \frac{P(E \mid h)\, P(h)}{\sum_{h' \in H} P(E \mid h')\, P(h')}. \tag{13}$$

Third, seeded *Monte Carlo* batches summarize declared branch outcomes via nearest-rank quantiles (5th, 50th, and 95th percentiles), describing batch dispersion rather than empirical market confidence intervals.

I. Matched Branches, Checkpoints, Replay, and Privacy

To isolate policy effects across any ledger, *matched branches* hold opening balances, seeds, and exogenous schedules identical while changing one policy rule. Checksummed *checkpoints* persist every settled step (and, in the lending runner, an append-only request-keyed decision cache) so interrupted local runs resume without repeating completed work. Afterward, *replay* re-runs the saved actions $a_{i,t}$ through the hash-pinned engine without calling *Laya* to verify bit-identical state trajectories.

Finally, self-hosted execution defines *Boyun*'s cost, privacy, and auditability profile. Running *Laya* locally incurs zero API fees and avoids sending actor observations to a hosted inference endpoint. This claim remains scoped to the local inference path: external data requests, artifact sharing, optional on-chain commitments, and host access are separate privacy boundaries. In exchange, *Laya* enforces a 1024-token context ceiling (truncating inputs are rejected) and exposes no internal chain of thought [11]. For no-API baselines and external agent workflows, a deterministic immediate-utility rule and a local stdio *Model Context Protocol* (MCP) server [27] run completely offline. When tamper evidence is needed prior to disclosure, an optional *cryptographic commitment* hashes the artifact with a fresh 32-byte salt via SHA-256 and records the digest on *NEAR* testnet [28], [29], binding timestamped integrity without certifying behavioral realism. Section V designs the experiments that test these capabilities and boundaries.

V. Evaluation Design

Section IV specified how *Boyun* executes a world; this section specifies how we test whether that execution does what it claims and what it adds in practice. Two kinds of test are needed, and they answer different questions. *Verification* asks whether the program carries out the stated rules correctly. *Validation* asks whether those rules and the model's choices resemble real actors and markets. The tests below are mainly verification and capability tests; none of them validates *Boyun* as a forecaster, and the results in Section VI are interpreted accordingly.

A. Research Questions

Given that boundary, the evaluation addresses six research questions (RQs), ordered from the most controlled test to the most open comparison:

RQ1 Do the implemented calculations, timing, accounting, and replay match their specifications in Section IV?

RQ2 Do *social-graph* information delivery, joint rules, and numeric preferences have separate and inspectable effects on *agent-swarm* choices and outcomes?

RQ3 Can a fourteen-day financial case with a sixteen-agent swarm run with settled trades, simulated token-price trajectories, recorded rejections, matched policy branches, replay, and a bounded budget, and what does it record?

RQ4 Given the same brief, what does *Boyun*'s constrained *agent swarm* record that the original *MiroFish* does not, and what does *MiroFish* provide that *Boyun* lacks?

RQ5 What do the local model's score distributions and regime responses reveal about the limits of its simulated *agent-swarm* behavior?

RQ6 In a ninety-day lending launch across a twenty-agent swarm, which observations follow from explicit rules and which depend on unvalidated agent choices?

B. Synthetic Mechanism Experiments

The first two questions require tests whose correct answers are known in advance. Six small synthetic experiments provide them (Table V). Each uses invented inputs, and wherever an action is needed it uses a fixed policy rather than a model, so any discrepancy must come from the engine. No model is called and network access is disabled while they run. Experiments 1, 2, and 4 test the analytical tools and scoring formulas against hand calculations; Experiment 3 tests timing, accounting, and replay; Experiments 5 and 6 address RQ2 by switching single mechanisms on and off. These experiments are complemented by the engine's automated test suite.

TABLE V
SYNTHETIC MECHANISM EXPERIMENTS.

Exp.	Mechanism	Known answer checked
1	Game analysis	Expected utility, regret, and pure Nash profiles, Eqs. (11)–(12)
2	Bayesian updating	Two sequential posteriors, Eq. (13)
3	Timing and accounting	Event order, clipping, conservation, replay, Eqs. (4)–(7)
4	Forecast scoring	Brier and log scores against a baseline, Eqs. (15)–(16)
5	Communication ablation	Delivery and a joint rule switched on and off
6	Preferences vs. prose	Changing a weight versus changing persona text

C. MACRON Case Study

Synthetic checks cannot show whether the engine copes with a realistic case, so RQ3 uses *MACRON*, a prospective fourteen-day launch of a hypothetical political memecoin on a *Solana* launch venue, beginning 2 November 2026. A founder with $50,000 of capital, of which $10,000 is a protected reserve, launches the token on a bonding curve that later migrates to a pool (Section IV-F). A *swarm of agents* comprising sixteen modeled participants takes part: the founder, a market maker, four retail traders with different strategies, a large buyer, an attention trader, four promoters, a meme curator, an organic supporter, a campaign operator, and a foundation. Each agent in the swarm communicates over the *social graph* and receives its own state, the public market history, and only the private instructions it is permitted to see. Real dated events, such as scheduled parliamentary sessions, enter as exogenous events, without assuming their outcomes.

The case compares founder plans in a full factorial design. The founder's initial position is $5,000, $10,000, or $15,000; upfront marketing and operating costs are $5,000 or $10,000; the market regime is *BULL* or *BEAR*; and the response to an exit alert is either *pause-and-observe*, which stops discretionary spending and keeps the tokens, or *staged exit*, which sells in tranches. The regimes are hypotheses about risk appetite, not observed market states. The resulting $3 \times 2 \times 2 \times 2 = 24$ paths run hourly for up to 336 hours, with finer steps around the launch, recording both per-agent actions and the resulting token-price trajectory across every simulated world. Each path uses one seed, so the design compares plans under common conditions rather than sampling a population of outcomes. All 24 paths use local *Laya* inference from hour 0 to hour 336. No hosted-model decisions or mixed-model continuations enter these case results. The packages, seeds, response policies, and evidence registers are fixed before execution. Offline reference paths remain engineering checks rather than substitutes for these model-driven paths.

D. Comparison with MiroFish

Because *Boyun* was inspired by *MiroFish*'s *agent-swarm* paradigm, RQ4 compares the two on the same case. The original *MiroFish* (source revision `7657031`) [5] received the same frozen *MACRON* brief and event calendar, generated its sixteen personas with its own native prompts, and ran its social simulation on the *OASIS* platform [4] for 336 hourly rounds in each regime, followed by its own report generation. It used *Gemini* 3.5 Flash-Lite on free quotas. Interruptions caused by quota limits required restoring the saved actor memories and databases; the original random state could not be recovered and was reseeded, so the run is a declared recovery-adapted version rather than an uninterrupted stock run.

The comparison is therefore matched in its inputs but not in everything else. The two systems use different actor models, budgets, and actor activation rules, and they are designed to produce different kinds of output, social posts across a free-form social feed in one case and settled trades plus constrained *social-graph* actions across a ledger-settled *swarm of agents* in the other. We therefore compare what each system records and reports, rather than ranking them on a single score, and we make no claim that either forecasts the real launch.

E. Local Agent-Swarm Behavior

Beyond completion, RQ5 examines how local agent scores translate into observed swarm behavior. We report score concentration, filled and rejected trades, simulated token-price paths across the 24 worlds, and differences between the declared *BULL* and *BEAR* regimes. The lifecycle engine samples actions from the saved score distributions; therefore a weak preference between options can produce substantial turnover. These are descriptive diagnostics of this model and

selection rule, not a calibration study or a measure of real investor behavior.

F. NEAR Lending Case Study

Apart from token trading, RQ6 changes the financial mechanism to collateralized lending, retaining local *Laya* inference. Its case is a prospective launch of a lending market on *NEAR* that accepts native NEAR as collateral and lends Circle's native USDC, evaluated over its first 90 days. The world was built with the user through 41 recorded answers. Because no partner terms had been negotiated, the user chose to assume that $1,000,000 of USDC from friendly partners is already deposited at opening. The team holds $100,000, of which $10,000 is a capped loss reserve, and suppliers earn points only on USDC that is actually borrowed. A twenty-agent swarm of decision-makers takes part: the team, four friendly partners, two large independent suppliers and two supplier groups, four borrowers and one borrower group, an actor who both supplies and borrows to earn points, two liquidators, and three distribution partners. Their balances are synthetic, sized against the user's premises and a dated snapshot of *RHEA*, the main existing NEAR lending venue, which paid 9.88% a year to native-USDC suppliers and charged borrowers 15.29% [30]. Every agent in the swarm sees these rates as a fixed outside option.

Within that cast, the rules of Section IV-G apply with a 50% maximum loan-to-value, liquidation at 65%, a 5% liquidation bonus, a $500,000 aggregate borrowing cap, and, in the baseline, a $50,000 cap per borrower. Outsiders in the *agent swarm* can act only after a *social-graph* channel they follow delivers a campaign item; paid partners may accept, delay, or decline, and cash is spent only on delivery, so no conversion rate is assumed. The ledger settles hourly, and actors who are aware of the launch decide daily and whenever a trigger fires, such as a NEAR price move of 8% within 24 hours. Prices are real hourly closes from Binance [31], rescaled so that each window opens at the snapshot price: a choppy window beginning 12 August 2025, which contains the market-wide fall of 10 October 2025, and a bear window beginning 9 November 2025.

These inputs define four local *Laya* worlds (Table VI). Codes identify the price path (C for choppy, B for bear), per-borrower cap in thousands of dollars, rate curve, and local model. Each world has one seed and there is no no-launch arm; real-world odds are unknown. Raising the cap also splits a borrower cohort, so that comparison changes two design factors and does not isolate the cap alone.

G. Metrics and Reporting

The metrics follow from the questions and are reported with their units and coverage. Mechanism tests report exact values against hand calculations. Case runs report fills, rejections, traded cash, local model calls, recorded run time, and API fees separately from hardware and electricity costs. Because live paths may stop at different hours, coverage is reported as completed path-hours out of the planned 24×336, and

TABLE VI
LOCAL *Laya* LENDING WORLDS. R2: 2% BASE RATE, 10% AT 80% UTILIZATION, 50% AT FULL USE. R8: 8% BASE, 15% AT 50%, 60% AT FULL USE. WORLDS WITH A $100K CAP ALSO SPLIT THE 80-PERSON BORROWER GROUP INTO EIGHT ACCOUNTS.

World	Path	Cap	Curve	Model
C-P50-R2-L	Choppy	$50K	R2	*Laya*
C-P100-R2-L	Choppy	$100K	R2	*Laya*
C-P100-R8-L	Choppy	$100K	R8	*Laya*
B-P100-R8-L	Bear	$100K	R8	*Laya*

every path is labeled completed, budget-limited, or interrupted. Policies are compared only at a common time. For each plan pair p, let τ_p be the latest hour saved in both of its branches, and let $C_p(\tau_p)$ be the founder's realized net-cash position relative to initial capital at that hour. The matched difference is

$$D_p = C_p^{\text{exit}}(\tau_p) - C_p^{\text{pause}}(\tau_p). \tag{14}$$

A positive D_p means staged exit held more cash than pausing at the same moment. It counts only realized cash; unsold tokens are reported separately, so D_p is not a comparison of total wealth.

Where probabilities can be compared with outcomes, *Boyun* uses the Brier and logarithmic scores [9], [10]. For n binary cases with forecast probability p_j and outcome $y_j \in \{0, 1\}$,

$$\text{Brier} = \frac{1}{n}\sum_{j=1}^{n}(p_j - y_j)^2, \tag{15}$$

$$\text{LogLoss} = -\frac{1}{n}\sum_{j=1}^{n}\big[y_j \log p_j + (1 - y_j)\log(1 - p_j)\big]. \tag{16}$$

Both are averages, lower is better, and both are minimized in expectation by reporting the true probability; the logarithmic score penalizes confident errors much more heavily. To avoid $\log 0$, probabilities are bounded to $[10^{-15}, 1 - 10^{-15}]$. Each case path has one seed, so we report observed ranges rather than confidence intervals. For the lending case, the measures are borrowed principal, utilization, liquidations, bad debt, the protocol's share of interest against campaign spending, and how often *Laya* borrowers chose to borrow when offered the option.

VI. Results

This section reports the outcomes of the tests defined in Section V, in the order of the research questions. Results from synthetic experiments and full-horizon local *Laya* case runs are distinguished because they support different kinds of claim.

A. Mechanism Checks (RQ1, RQ2)

The first experiment checks the game-analysis tool on a small coordination game. A founder chooses whether to fund an independent review, and a liquidity partner chooses whether to commit funds to escrow or wait. Table VII gives both players' assumed utilities, which are preference points rather than cash. The tool finds exactly two pure Nash equilibria,

TABLE VII
EXPERIMENT 1 UTILITIES: (FOUNDER, PARTNER).

	Partner commits	Partner waits
Founder funds review	$(6, 5)$	$(-2, 1)$
Founder skips review	$(2, -4)$	$(0, 0)$

TABLE VIII
EXPERIMENT 3: STATE OVER TIME. CASH IS IN CENTS.

Stage	Readiness	Alice	Bob	Spent
Initialization	1	100	100	0
Round-one observation	2	100	100	0
Round-one settlement	10	40	40	120
Round-two observation	9	40	40	120
Terminal state	9	40	40	120

review/commit and skip/wait: in each, either player would lose by changing its action alone. If the founder believes the partner commits with probability 0.4, Eq. (11) gives an expected utility of $0.4 \times 6 + 0.6 \times (-2) = 1.2$ for review and $0.4 \times 2 + 0.6 \times 0 = 0.8$ for skipping, so skipping has a regret of 0.4 by Eq. (12). The tool returns these values as exact fractions ($6/5$, $4/5$, and $2/5$).

The second experiment applies Eq. (13) twice. Two hypotheses, that attention to the token persists or fades, start with priors 0.4 and 0.6. A first signal has likelihood 0.75 if attention persists and 0.25 if it fades, which raises the persistence posterior to $2/3$. A second signal with likelihoods 0.6 and 0.3, assumed independent of the first given the hypothesis, raises it to $4/5$. Both match the hand calculation. The independence assumption is what allows the two likelihoods to be multiplied; correlated signals would need a joint likelihood to avoid counting the same information twice.

The third experiment follows two builders, each with 100 cents, through two rounds of a world with one bounded readiness metric (Table VIII). A scheduled event raises readiness from 1 to 2 before the first observations. Both builders then spend 60 cents on an action worth 6 points each; the effects sum to 14 and Eq. (5) clips the result to the upper bound of 10. A second scheduled event lowers it to 9 before the next observations, and with only 40 cents left, building is no longer affordable, so both builders wait. Remaining cash plus spending is $40 + 40 + 120 = 200$ cents, as Eq. (7) requires, and replay reproduces every state exactly. The build-then-wait policy is imposed, so this tests execution rather than any model's choices.

The fourth experiment checks the scoring formulas. Two synthetic training outcomes give a baseline positive rate of 0.5. On two later cases with outcomes $(0, 1)$, supplied forecasts $(0.25, 0.75)$ score a Brier value of 0.0625 and a log loss of 0.287682 by Eqs. (15) and (16), against 0.25 and 0.693147 for the constant baseline. Two invented cases confirm the formulas, not any forecasting skill.

Experiments 5 and 6 turn from arithmetic to RQ2, isolating one mechanism at a time. In Experiment 5, Alice builds in round one and publishes a message; Bob joins in round two only if he receives it, and a declared joint rule adds three readiness points when Alice waits while Bob joins. Switching delivery and the joint rule on and off (Table IX) separates the two effects. Removing only the joint rule leaves Bob's choice unchanged but lowers final readiness from 6 to 3. Removing delivery changes Bob's choice to waiting, so the joint rule never activates and readiness stays at 1. One mechanism therefore changes what an actor does, and the other changes what that action is worth. All four runs replay exactly and conserve the 200 cents of starting cash.

TABLE IX
EXPERIMENT 5: DELIVERY AND JOINT-RULE ABLATION. COST AND CASH ARE IN CENTS; STARTING CASH IS 200 IN EVERY CONDITION.

Delivery	Joint rule	Readiness	Cost	Cash left
On	On	6	50	150
On	Off	3	50	150
Off	On	1	20	180
Off	Off	1	20	180

Experiment 6 asks whether an actor's behavior follows its numeric preferences or its persona text. An eight-role world is run with the deterministic offline rule, which picks the action with the greatest immediate utility change. Under the declared weights, the participation-oriented role joins and the risk-sensitive role waits. Changing only the former's risk weight from -500 to $-5{,}000$ turns its joining value from $8{,}500$ to $-5{,}000$ utility points, and it waits instead; final participation falls from 44 to 34, risk from 24 to 18, and total spending from 120,000 to 110,000 cents. Rewriting every role's motivation and goal to emphasize caution, while keeping the weights fixed, changes nothing at all. In this baseline, then, the prose describes an actor but the weights determine its behavior. Whether *Laya* also follows the weights rather than the prose is a separate question that these experiments do not answer.

Beyond the six experiments, the engine's automated suite of 1,904 passed tests (with two optional-dependency skips) verified the code used for this paper, covering, among others, the market maker, the collateralized lending ledger, checkpoint recovery, actor inference adapters, and the MCP server. The six experiments' outputs are regenerated with network access disabled and match the saved records byte for byte.

B. MACRON Case Study (RQ3)

Moving from mechanism checks to full-horizon execution across the sixteen-agent swarm, all 24 local *Laya* paths reached hour 336, covering all 8,064 planned path-hours. They record 8,754 decision points and 136,841 local swarm-agent requests with no per-call API fee (Table X). All paths were independently replayed from packages, seeds, and saved choices. These results demonstrate execution coverage under the selected world rules, not predictive accuracy.

Across the saved receipts, cumulative inference elapsed time is 146,873.669 seconds (about 40.8 hours). This sums per-call elapsed times across concurrent workers; it is neither

TABLE X

FULL-HORIZON *MACRON* PATHS DECIDED BY *Laya* FROM HOUR 0. EACH REGIME HAS 12 PATHS; CASH AMOUNTS ARE SIMULATED U.S. DOLLARS, AND ALL REJECTIONS ARE SLIPPAGE REJECTIONS.

Measure	*BULL*	*BEAR*
Last hour reached (all paths)	336	336
Completed path-hours (of 4,032)	4,032	4,032
Replay-verified decision points	4,381	4,373
Local *Laya* requests	68,508	68,333
Filled buys	13,913	13,433
Filled sells	11,459	11,446
Rejected trades	5,894	5,456
Traded cash ($)	8,493,778	8,282,843

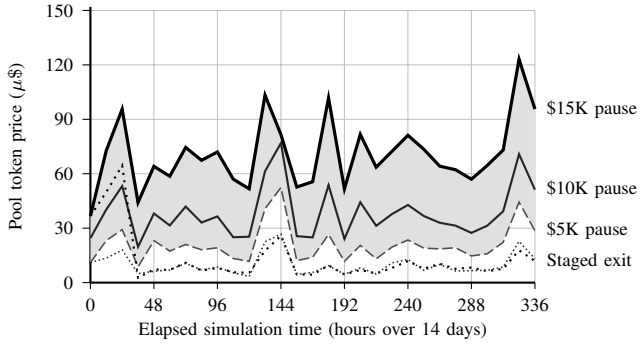


Fig. 4. Simulated *MACRON* token-price trajectories (μ\$, sampled at 12-hour intervals over 336 hours) across the 24 local *Laya* worlds. Solid and dashed curves show mean trajectories under *pause-and-observe* for \$15K, \$10K, and \$5K initial founder positions (shaded band spans the pause envelope); dotted curves show *staged exit* (\$15K and \$5K tiers), where tranche sales compress pool prices across all capital tiers to 10.6–14.7 μ\$ at hour 336.

end-to-end wall-clock duration nor CPU core-hours. Every receipt identifies local *Laya* transport and records zero API fee. Hardware, electricity, setup, data access, and other execution overhead remain outside that fee measurement.

Within those paths, the engine filled 25,372 trades in *BULL* and 24,879 in *BEAR*. It recorded 11,350 slippage rejections in total instead of settling orders outside the permitted limits. The terminal state is the end of observation; it does not by itself establish termination of the hypothetical project.

Tracing the pool price across those 24 worlds shows how founder policy and *agent-swarm* trading interact on the market ledger (Figure 4). Opening prices stratify by the founder's initial purchase size at 11.0 μ\$ (\$5,000 position), 23.5–25.5 μ\$ (\$10,000 position), and 36.7 μ\$ (\$15,000 position). Under *pause-and-observe*, where the founder halts discretionary marketing upon the exit alert but retains token inventory while the remaining fifteen agents trade, simulated prices oscillate with scheduled catalysts and reach a single-path hourly peak of 287.2 μ\$ (with 12-hour group means peaking at 52.4, 76.9, and 123.1 μ\$) before settling at hour 336 at 26.7–29.8 μ\$ (\$5,000), 48.2–58.3 μ\$ (\$10,000), and 75.3–117.1 μ\$ (\$15,000). Under *staged exit*, by contrast, selling the founder's inventory in tranches into the constant-product pool depresses the token price after hour 24 and compresses all twelve staged-exit worlds into a terminal band of 10.6–14.7 μ\$ at hour 336. These curves are conditional scenario trajectories under the declared rules, not price forecasts.

TABLE XI

MACRON UNDER THE ORIGINAL *MiroFish* AND UNDER *Boyun* (FULL-HORIZON LOCAL *Laya* PATHS), AS *BULL* / *BEAR*. COUNTS ARE RECOMPUTED FROM EACH SYSTEM'S SAVED DATABASES.

Measure	*MiroFish*	*Boyun*
Actor model	*Gemini* 3.5 Flash-Lite	*Laya*
Paths per regime	1 social path	12 plan paths
Hours completed	336 / 336	336 / 336
Original posts	537 / 371	–
Quote posts	396 / 206	–
Likes and reposts	153 / 154	–
Do-nothing actions	819 / 859	–
Filled trades	not modeled	25,372 / 24,879
Rejected trades	not modeled	5,894 / 5,456
Settled cash and tokens	not modeled	integer ledgers
Matched policy pairs	none	6 / 6
Replay of decisions	not available	4,381 / 4,373 points
Confirmed model requests	1,911 / 1,499	136,841 local calls
Model cost	free quota	zero API fee
Reporting focus	social discourse	settled finances and policy comparison

C. Comparison with MiroFish (RQ4)

With the same brief, *MiroFish* and *Boyun* produced records of a different kind, and Table XI places them side by side. Both *MiroFish* regimes and all 24 local *Boyun* paths completed the planned 336 hours. Over those rounds, its sixteen personas wrote 537 original posts and 396 quote posts under *BULL*, against 371 and 206 under *BEAR*, and chose to do nothing 819 and 859 times. The posts are written in each persona's own voice and respond to one another; a promoter, for example, quotes the founder's launch post to pitch paid attention as the real driver of demand. *Boyun*'s *social-graph* messages, by contrast, are pre-written texts attached to actions taken by agents in the swarm, so its social layer is far less expressive.

The financial side of the comparison runs the other way. *MiroFish* models no orders, balances, or pool, so a post enthusiastic about buying changes no price and no founder's cash, and the founder's question, how much money each plan keeps, cannot be answered from its output. *Boyun* settles every trade, records why rejected trades failed, compares plans at matched times, and replays every saved decision. *MiroFish*'s original random state was not checkpointed and had to be reseeded during recovery, so its runs cannot be reproduced decision by decision.

The output distinction also shapes reporting. The saved *MiroFish* reports discuss sentiment, stakeholder behavior, and risks in prose, whereas *Boyun*'s case analysis uses settled trades, recorded failures, and policy differences. Neither output establishes what will happen in a real launch. Different models, activation rules, and recovery procedures prevent interpreting this comparison as a general performance ranking.

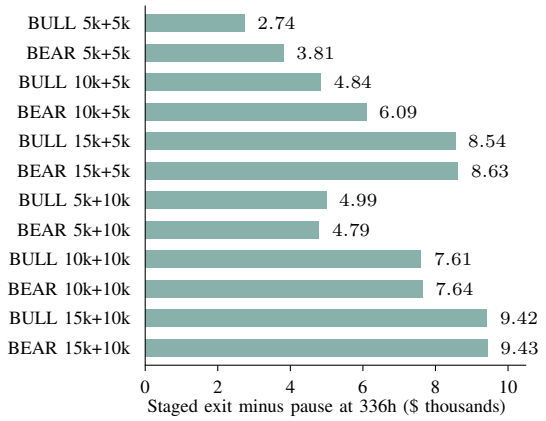


Fig. 5. Cash difference D_p for all 12 local *Laya* plan pairs at the common terminal hour 336. Labels give regime, founder position budget, and upfront cost, in thousands of dollars. Unsold inventory is excluded; therefore these bars do not rank total wealth.

D. Local Agent-Swarm Behavior (RQ5)

Beyond coverage, the completed matrix exposes limits of the local *agent-swarm* model. *Laya* produced similar traded cash across the two regimes: $8,493,778 in *BULL* and $8,282,843 in *BEAR*, a gap of approximately 2.5%. Across 136,841 multi-option requests, its mean top score was 0.42. Near-uniform option scores, combined with the engine's sampling rule, are consistent with heavy two-way turnover and weak regime separation. The recorded rejection count is about 1.30 per decision point. These observations motivate behavioral validation rather than a market forecast.

Despite weak regime separation, staged exit held between $2,736.37 and $9,434.99 more realized cash than pause in all 12 pairs at hour 336 (Figure 5). Founder purchases and exit tranches follow explicit policy rules, so this cash comparison is not evidence that the agent model discovered a superior strategy. In all six $10,000-upfront pairs, upfront spending alone meets the $10,000 loss-budget alert at launch.

Including unsold inventory changes the interpretation because of the pool-price divergence shown in Figure 4. At hour 336 its pool quote is between $4,278 and $11,805 under pause, reflecting terminal prices of 26.7–117.1 μ\$ when the founder's inventory remains off the market. Adding that quote to cash puts pause ahead by $464–$3,260 in the six $5,000-upfront pairs; staged exit remains $256–$843 ahead in the six $10,000-upfront pairs. A pool quote is not guaranteed executable proceeds. The comparison must therefore show realized cash and retained inventory separately.

E. NEAR Lending Case Study (RQ6)

Apart from sampled token trading, four local *Laya* lending worlds complete 90 days each (2,160 hours), with decisions selected by argmax. Their event journals support exact ledger replay. Table XII summarizes only those four worlds; hosted and mixed-model runs are excluded.

First, the observed borrowing ceilings reflect explicit caps, while whether actors reach them depends on their choices.

TABLE XII
LOCAL *Laya* LENDING WORLDS AFTER 90 DAYS. PROTOCOL SHARE IS 15% OF BORROWER INTEREST, COLLECTED PLUS STILL OWED. CAMPAIGN SPENDING WAS $20,000 IN EACH LOCAL WORLD. "BORROWED" COUNTS DECISIONS IN WHICH BORROWING WAS OFFERED AND CHOSEN.

World	Lent, day 30	Peak use	Liquid.	Protocol $	Borrowed
C-P50-R2-L	$300K	15.6%	1	361	54/54
C-P100-R2-L	$500K	25.8%	12	703	292/295
C-P100-R8-L	$500K	25.8%	13	1,802	258/260
B-P100-R8-L	$374K	19.5%	20	1,077	431/433

In C-P50-R2-L each of the six borrowing accounts reached its $50,000 cap by day 30, and once that cap was raised, both *Laya* choppy worlds reached the $500,000 aggregate cap by the same day. With about $2,000,000 supplied, utilization could not exceed about 25%. Under Eq. (8), suppliers therefore earned about 1% a year under curve R2 and 2.4% under R8, against *RHEA*'s 9.88%; matching *RHEA* at 25% utilization would require a borrow rate near 46%. Second, the risk rules left no bad debt in any world, and the loss reserve was never used. On the bear path, hourly closes fell 68% and twenty liquidations repaid $302,000 without loss.

That second result carries two conditions. It depends on the assumed liquidators' cash and exit depth, which were not measured, and on the oracle's hourly sampling. On 10 October 2025 NEAR briefly traded 40% below the choppy window's opening price but closed that hour about 9% lower, so the deepest fall the ledger saw on that path was 31%, and its liquidations came on days 60–61, 67, and 80–85. Liquidations during such intrahour moves remain untested.

The behavioral observations require a different interpretation. Borrowing was selected in 258 of 260 offered decisions under curve R8 and 292 of 295 under curve R2 in the choppy worlds. This near-saturation persists despite materially different rates and should not be read as measured demand. Partner acceptance, supply retention, and campaign reach also depend on uncalibrated model choices and synthetic balances. Real borrower intentions, observed conversion, and multiple seeds are needed before extrapolating those behaviors.

Within these four worlds, the launch's modeled economics are thin. Over 90 days, the protocol's share of interest ranged from $361 to $1,802 in the *Laya* worlds, against $20,000 spent on the campaign.

VII. Discussion and Limitations

The results in Section VI establish what the system does reliably and where its evidence stops. This section interprets them, explains the local actor-inference boundary, relates *Boyun* to *MiroFish*, and sets out the work needed before any predictive claim could be made.

A. What the Results Show

Taken together, the mechanism checks answer RQ1 and RQ2 within their stated scope: arithmetic, timing, accounting, and saved-decision replay agree with specified rules. Changing individual mechanisms changes results in controlled ways.

In particular, the offline preference test demonstrates that changing numeric weights affects its policy, whereas rewriting persona prose alone does not. That test does not establish how a learned actor model interprets either.

Extending those checks to a larger setting, local *Laya* inference carries all 24 token-launch paths and their simulated price trajectories through 336 hours across the sixteen-agent swarm. This verifies the ability to record and settle repeated *agent-swarm* decisions without a per-call API fee. It does not validate the regime hypotheses or the trading behavior. Realized cash favors staged exit in the observed pairs, yet including inventory quotes derived from the terminal pool prices in Figure 4 changes that comparison in six pairs. The metric selected therefore affects what a reader may conclude.

Apart from trading, the four local lending worlds distinguish a numerical constraint from behavior that happens to reach it. A borrower cap limits the maximum possible loan, but reaching that cap depends on model choices. Likewise, zero observed bad debt is conditional on the tested prices, liquidation rules, liquidator resources, and depth. Reports should identify these assumptions rather than treating either result as an unconditional property of a launch.

B. Self-Hosted Actor Inference

The local execution path addresses a practical privacy requirement: world observations can be scored on the user's hardware instead of being sent to a hosted actor API. This narrows a data-transfer boundary; it is not a proof of system-wide confidentiality. External data services, optional commitments, artifact sharing, and host access remain separate considerations. Zero API fee likewise excludes hardware, electricity, setup, and maintenance costs.

Within that local path, *Laya*'s limitations are consequential. Its context is limited to 1024 tokens, observations that would be truncated are rejected, and its option scores are uncalibrated [11]. The recorded trading distributions show limited regime separation, while lending choices frequently select borrowing under both tested rate curves. These patterns should motivate validation against actual behavior, not be presented as measured investor or borrower demand.

Looking beyond the current experiments, other open-weight typed models, such as *Kev*, are candidates for comparison [22]. They have not been evaluated in these case studies, and compatibility alone would not imply behavioral quality. Multiple seeds, measured demand, and matched evaluations are stronger next steps than claiming predictive value from local inference or from agreement between models.

C. Relation to MiroFish

Although *Boyun* was inspired by *MiroFish*'s *agent-swarm* design, the comparison with *MiroFish* suggests a division of labor rather than a contest. *MiroFish* produces rich, persona-specific discourse across an open-ended social feed but no financial consequences; *Boyun* organizes its participants as a constrained *swarm of agents* communicating over an explicit *social graph* and produces settled, replayable financial consequences and token-price trajectories, but only pre-written messages. A natural next step is to let language-model agents in the swarm write *Boyun*'s *social-graph* messages while the engine keeps control of delivery, timing, and settlement, so that what an agent says is generated freely but what it can see and spend is still enforced. Doing so would raise the cost per decision and introduce new questions about how text should influence trading, which would need their own evaluation.

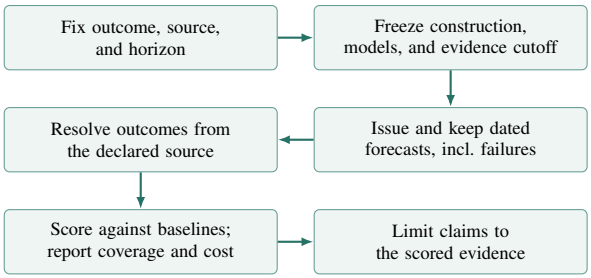


Fig. 6. Proposed prospective validation protocol. This is future work, not an experiment reported in this paper.

D. Toward Predictive Validation

None of the tests in this paper measures forecasting accuracy, and Figure 6 outlines what such a test would require. A study must first fix one measurable outcome, its resolution source, and its horizon, and freeze the world-construction procedure, models, and evidence cutoff before any forecast is issued. Forecasts must then be dated and kept, including failed runs, and scored against resolved outcomes using Eqs. (15) and (16), alongside simple baselines such as a historical base rate and a single-model forecast given the same evidence. Cases that share a time period or narrative are not independent and must be treated as such when estimating uncertainty. A separate user study would be needed to test whether *Boyun*'s reports improve decisions, which is a different question from whether its probabilities are accurate.

E. Limitations

Several limitations apply to the full evaluation. The case results use one seed per plan, synthetic actor balances and preferences, and declared event or price schedules. They do not represent independent observations of human behavior. In *MACRON*, near-uniform option scores weaken regime separation, and upfront spending alone triggers the loss-budget alert in the higher-upfront plans. Founder trades are partly policy-driven, so model choice cannot explain every cash difference.

On the market side, a simplified constant-product pool omits order books, transaction ordering, front-running, outages, and venue-specific execution details. On the lending side, assumed liquidator cash and exit depth determine which liquidations can settle; historical hourly closes do not test intrahour price shocks. The competing venue's rates are fixed at one snapshot and campaign reach has no measured conversion. These conditions bound the scope of every financial conclusion.

Finally, messages attached to actions are pre-written, giving *Boyun* a narrower social layer than open-ended discourse. The *MiroFish* comparison matches the brief, but actor models, activation rules, and recovery procedures differ. It compares recorded capabilities rather than forecasting accuracy or general superiority.

F. Responsible Use

These limits also shape how *Boyun* should and should not be used. Its outputs are conditional explorations of stated assumptions, not financial advice or price forecasts. Because the case study concerns paid promotion of a speculative token, the same tools could in principle be used to rehearse campaigns that mislead buyers; reports therefore state their assumptions and limits explicitly, and they should not be presented to investors as evidence of likely returns. World files and reports can contain private strategy, so authors should share them deliberately. Local actor inference avoids sending observations to an actor API; external services and publication remain separate disclosure decisions.

VIII. Conclusion

This paper examined how an explicitly specified multi-agent world constrains *agent-swarm* observations, actions, and settlement while preserving records of the resulting choices and consequences. Inspired by *MiroFish*, *Boyun* combines game-theoretic specification, a *social graph* (*directed information graph*) connecting a *swarm of agents*, typed local actor inference, and ledger-specific execution. Saved decisions support deterministic replay, while model scores remain distinct from calibrated event probabilities.

Within this design, synthetic checks verify calculations and isolate selected mechanisms. A full-horizon local *Laya* token-launch matrix records 24 paths and their conditional token-price trajectories through fourteen days, while four local lending worlds cover ninety days each. The same-brief comparison with *MiroFish* illustrates the difference between generating social discourse and settling financial actions across an *agent swarm*; it does not establish a performance ranking. Case outcomes remain conditional on their policies, synthetic agents, and execution assumptions.

The next steps follow from these evidence boundaries: test multiple seeds and alternative open-weight models, measure behavior rather than assume demand, and evaluate dated forecasts against resolved outcomes. Self-hosted inference provides a useful execution option, but the quality of the resulting financial analysis must be established separately.

References

[1] V. Grimm, S. F. Railsback, C. E. Vincenot, U. Berger, C. Gallagher, D. L. DeAngelis, B. Edmonds, J. Ge, J. Giske, J. Groeneveld, A. S. A. Johnston, A. Milles, J. Nabe-Nielsen, J. G. Polhill, V. Radchuk, M.-S. Rohwäder, R. A. Stillman, J. C. Thiele, and D. Ayllón, "The ODD protocol for describing agent-based and other simulation models: A second update to improve clarity, replication, and structural realism," *Journal of Artificial Societies and Social Simulation*, vol. 23, no. 2, p. 7, 2020. [Online]. Available: https://www.jasss.org/23/2/7.html

[2] J. M. Galán, L. R. Izquierdo, S. S. Izquierdo, J. I. Santos, R. del Olmo, A. López-Paredes, and B. Edmonds, "Errors and artefacts in agent-based modelling," *Journal of Artificial Societies and Social Simulation*, vol. 12, no. 1, p. 1, 2009. [Online]. Available: https://www.jasss.org/12/1/1.html

[3] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," arXiv:2304.03442, version 2, 2023, version-specific preprint cited; paper also appeared at UIST 2023. [Online]. Available: https://arxiv.org/abs/2304.03442v2

[4] Z. Yang, Z. Zhang, Z. Zheng, Y. Jiang, Z. Gan, Z. Wang, Z. Ling, J. Chen, M. Ma, B. Dong, P. Gupta, S. Hu, Z. Yin, G. Li, X. Jia, L. Wang, B. Ghanem, H. Lu, C. Lu, W. Ouyang, Y. Qiao, P. Torr, and J. Shao, "OASIS: Open agent social interaction simulations with one million agents," arXiv:2411.11581, version 5, 2025, original preprint 2024; cited revision 2025. [Online]. Available: https://arxiv.org/abs/2411.11581v5

[5] MiroFish contributors, "MiroFish source repository," GitHub, 2026, software revision 7657031ac01184afe2cb220f5ee3545573b5e843. [Online]. Available: https://github.com/666ghj/MiroFish/tree/7657031ac01184afe2cb220f5ee3545573b5e843

[6] B. Polak, "ECON 159, lecture 2: Putting yourselves into other people's shoes," Open Yale Courses, Yale University, 2007, open course lecture. [Online]. Available: https://oyc.yale.edu/economics/econ-159/lecture-2

[7] J. F. Nash, Jr., "Equilibrium points in n-person games," *Proceedings of the National Academy of Sciences*, vol. 36, no. 1, pp. 48–49, 1950. [Online]. Available: https://doi.org/10.1073/pnas.36.1.48

[8] H. Pishro-Nik, *Introduction to Probability, Statistics, and Random Processes*. Kappa Research, 2014, online section 1.4.3, Bayes' Rule. [Online]. Available: https://www.probabilitycourse.com/chapter1/1_4_3_bayes_rule.php

[9] G. W. Brier, "Verification of forecasts expressed in terms of probability," *Monthly Weather Review*, vol. 78, no. 1, pp. 1–3, 1950.

[10] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American Statistical Association*, vol. 102, no. 477, pp. 359–378, 2007. [Online]. Available: https://doi.org/10.1198/016214506000001437

[11] Convai Innovations, "laya-typed-decisions: Fine-tuned Laya system one decision model, model card," Hugging Face, 2026, open-weight model card, Apache 2.0 license; accessed November 4, 2026. [Online]. Available: https://huggingface.co/convaiinnovations/laya-typed-decisions

[12] M. Lanctot, E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei *et al.*, "OpenSpiel: A framework for reinforcement learning in games," arXiv:1908.09453, version 1, 2019. [Online]. Available: https://arxiv.org/abs/1908.09453v1

[13] S. Zheng, A. Trott, S. Srinivasa, N. Naik, M. Gruesbeck, D. C. Parkes, and R. Socher, "The AI economist: Improving equality and productivity with AI-driven tax policies," arXiv:2004.13332, version 1, 2020, research system and Foundation economic environment. [Online]. Available: https://arxiv.org/abs/2004.13332v1

[14] N. Li, C. Gao, M. Li, Y. Li, and Q. Liao, "EconAgent: Large language model-empowered agents for simulating macroeconomic activities," arXiv:2310.10436, version 4, 2024, aCL 2024 main conference; original preprint 2023. [Online]. Available: https://arxiv.org/abs/2310.10436v4

[15] D. Byrd, M. Hybinette, and T. H. Balch, "ABIDES: Towards high-fidelity market simulation for AI research," arXiv:1904.12066, version 1, 2019. [Online]. Available: https://arxiv.org/abs/1904.12066v1

[16] G. Angeris, H.-T. Kao, R. Chiang, C. Noyes, and T. Chitra, "An analysis of Uniswap markets," arXiv:1911.03380, version 7, 2021, original preprint 2019; cited revision 2021. [Online]. Available: https://arxiv.org/abs/1911.03380v7

[17] J. Xu, K. Paruch, S. Cousaert, and Y. Feng, "SoK: Decentralized exchanges (DEX) with automated market maker (AMM) protocols," *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–50, 2023. [Online]. Available: https://doi.org/10.1145/3570639

[18] L. Gudgeon, S. M. Werner, D. Perez, and W. J. Knottenbelt, "DeFi protocols for loanable funds: Interest rates, liquidity and market efficiency," in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 2020, pp. 92–112.

[19] E. Akata, L. Schulz, J. Coda-Forno, S. J. Oh, M. Bethge, and E. Schulz, "Playing repeated games with large language models,"

arXiv:2305.16867, version 2, 2025, original preprint 2023; cited revision May 2025. [Online]. Available: https://arxiv.org/abs/2305.16867v2

[20] D. Almeida, "Introducing system one models and jev," TypeSafe AI, September 2026, official model introduction, September 15, 2026; accessed October 29, 2026. [Online]. Available: https://typesafe.ai/blog/introducing-system-one-models-and-jev

[21] TypeSafe AI, "System one api and list models," 2026, official documentation; accessed October 28, 2026. [Online]. Available: https://docs.typesafe.ai/api/system-one

[22] J. Palmer, "Kev: Jev-like decision models built on Qwen you can train and run on your own, software repository," GitHub, 2026, open-source repository, Apache 2.0 license; accessed November 4, 2026. [Online]. Available: https://github.com/jaredpalmer/kev

[23] B. Polak, "ECON 159, lecture 15: Backward induction: Chess, strategies, and credible threats," Open Yale Courses, Yale University, 2007, open course lecture. [Online]. Available: https://oyc.yale.edu/economics/econ-159/lecture-15

[24] ——, "ECON 159, lecture 18: Imperfect information: Information sets and sub-game perfection," Open Yale Courses, Yale University, 2007, open course lecture. [Online]. Available: https://oyc.yale.edu/economics/econ-159/lecture-18

[25] Pump.fun, "Pump program: Public documentation of the bonding-curve program," GitHub, 2026, official program documentation; accessed November 2, 2026. [Online]. Available: https://github.com/pump-fun/pump-public-docs/blob/main/docs/PUMP_PROGRAM_README.md

[26] Aave, "Liquidations and the health factor," 2026, official help documentation; accessed November 5, 2026. [Online]. Available: https://aave.com/help/borrowing/liquidations

[27] Model Context Protocol contributors, "Model context protocol: Architecture," 2025, protocol version 2025-11-25; accessed October 29, 2026. [Online]. Available: https://modelcontextprotocol.io/specification/2025-11-25/architecture

[28] NEAR Protocol, "NEAR developer documentation," 2026, official protocol and smart-contract documentation; accessed October 28, 2026. [Online]. Available: https://docs.near.org/

[29] A. Skidanov, I. Polosukhin, and B. Wang, "Nightshade: NEAR protocol sharding design 2.0," NEAR Protocol, Tech. Rep., February 2024, original design July 2019; cited update February 2024. [Online]. Available: https://discovery-domain.org/papers/nightshade.pdf

[30] RHEA Finance, "RHEA lending markets," 2026, live market page captured November 5, 2026, 08:14 UTC. [Online]. Available: https://app.rhea.finance/lending

[31] Binance, "Spot API: Kline and candlestick data," 2026, nEARUSDT and USDCUSDT hourly candles; accessed November 5, 2026. [Online]. Available: https://developers.binance.com/docs/binance-spot-api-docs/rest-api/market-data-endpoints