Collective Intelligence Design Laboratory

Technical Specification

A Visual Design Environment for Multi-Mechanism Coordination Systems

Version 2.0
December 2025

Equilibria Network

1. Product Vision

1.1 The Design Problem

Real-world collective intelligence operates through multiple coordination mechanisms simultaneously. Organizations blend hierarchical decision-making with market mechanisms and network-based information sharing. Political systems embed democratic processes within regulatory frameworks that constrain markets. Understanding these hybrid systems requires tools that can represent multiple coordination mechanisms operating on shared agent populations.

Current multi-agent simulation frameworks embed an architectural assumption: that collective intelligence requires centralized orchestration. This constrains the types of collective behaviors researchers can study. Markets don't need CEOs deciding every transaction. Social networks don't need central authorities determining information spread. Democratic processes don't need master planners coordinating citizen preferences.

1.2 The Laboratory Concept

The CI Design Laboratory provides a visual environment for designing, simulating, and analyzing multi-mechanism collective intelligence systems. Users construct coordination systems by placing agents, mechanisms, and connections on a canvas — then observe how information flows, trust evolves, and collective decisions emerge over time.

The core insight: designing coordination systems is inherently spatial and relational. Network topologies, mechanism placements, and agent groupings are visual concepts. Writing adjacency matrices by hand obscures the design; visual manipulation reveals it. The laboratory treats the graph canvas as the primary design artifact, with scenario files as the portable serialization.

1.3 Design Philosophy: Examples, Not Prescriptions

Throughout this specification, concrete examples (democracy mechanisms, market nodes, adversarial agents) are used to illustrate capabilities. These are prototypical instances, not fixed requirements. The system is mechanism-agnostic: any coordination pattern that can be expressed as a graph transformation can be registered and used. The wireframes accompanying this document show one possible configuration; researchers may create entirely different mechanism vocabularies for their domains.

1.4 Target Users

Computational Social Scientists

Researchers studying democratic innovation, market dynamics, or network effects who have theoretical models but lack experimental infrastructure. They want to test whether liquid democracy outperforms direct democracy under different expertise distributions, or how trust networks affect market efficiency. They think in concepts (delegation, voting, trust) and need visual tools that match their mental models.

AI Multi-Agent Researchers

Technical researchers studying emergent coordination in agent collectives. They have custom agent implementations and need coordination infrastructure to test them in. They want to drop agents into different institutional structures and observe how coordination properties change. They need export formats compatible with their analysis pipelines and reproducibility via configuration files.

Policy Analysts

Practitioners exploring governance mechanisms for human-AI coordination. They want to model scenarios like: What happens when AI agents participate in prediction markets alongside humans? How do recommendation algorithms affect democratic deliberation? They need accessible interfaces and clear visualizations they can share with non-technical stakeholders.

2. User Workflow

The laboratory guides users through a workflow from scenario selection to results export. The stages are not strictly sequential — users can have metrics panels open during simulation, return to the canvas mid-execution, or export at any point.

2.1 Stage 1: Welcome & Navigation

The entry point provides three navigation paths: Start a Simulation (proceeds to scenario selection), Tutorial (interactive walkthrough), and Documentation (reference material). The welcome screen also displays recent scenarios for quick access.

2.2 Stage 2: Scenario Selection

Users select from predefined scenarios or start with a blank canvas. Predefined scenarios are themselves loaded from scenario files, demonstrating that user-created designs are first-class citizens equal to built-in examples.

Example scenarios (illustrative, not exhaustive):

- **Farmers Market:** Decentralized exchange with producer and consumer agents
- **Tragedy of Commons:** Shared resource pool with extraction dynamics
- **Epidemic Spread:** Information or contagion propagation through network
- **Custom:** Blank canvas for user-defined topology

2.3 Stage 3: Global Configuration

Slider controls configure simulation-wide parameters. These establish baseline characteristics before canvas editing. Parameters include simulation duration (number of rounds), random seed for reproducibility, and any scenario-specific global settings.

2.4 Stage 4: Graph Canvas (Design Surface)

The canvas is the primary design surface where users construct network topologies by placing elements and drawing connections. This is where the coordination system takes shape visually.

Registry-Driven Toolbar

The toolbar is populated dynamically from the Mechanism Registry. Each registered element type appears as a draggable item with its configured icon, color, and label. The system ships with default registrations, but researchers can register custom element types that appear alongside built-ins.

Default element types (extensible via registry):

- **Agent Node:** Represents an individual agent in the system
- **Edge:** Directed connection between nodes
- **Mechanism Node:** Coordination mechanism (market, democracy, network, or custom)

The wireframes show example registrations (Democracy ◇, Market □) but these are illustrative. A researcher studying epidemic spread might register Infection Source, Quarantine Zone, and Hospital instead.

Canvas Interactions

- **Drag from toolbar:** Instantiates element at drop location
- **Click element:** Selects element, opens property panel
- **Drag on canvas:** Repositions selected element(s)
- **Click-and-drag between nodes:** Creates edge
- **Scroll/pinch:** Zoom canvas
- **Delete key:** Remove selected elements

Multi-Connect Tool

For connecting many agents to a single mechanism (or vice versa), the Multi-Connect Tool streamlines edge creation:

1. Select multiple agent nodes (shift+click or drag-select)
2. Click a mechanism node
3. Edges are created from all selected agents to the mechanism

This enables rapid construction of patterns like "all farmers connected to the market" or "all citizens connected to the voting mechanism" without clicking each edge individually.

Property Panel

When an element is selected, a side panel displays editable properties. The property schema is defined by the element's registered type — different mechanism types expose different configurable parameters. Common properties include unique ID, display label, and type-specific attributes.

Mechanism Scheduling

Each mechanism node has an associated schedule that determines when it activates. The scheduling panel allows users to configure:

- **Activation frequency:** How often the mechanism processes (e.g., every timestep, every 10 timesteps)

- **Latency:** Processing delay before results propagate
- **Availability windows:** Time ranges when agents can engage with the mechanism

This reflects real-world dynamics: markets might process continuously while democratic decisions occur in discrete deliberation periods.

2.5 Stage 5: Simulation Execution

Once the design is complete, users execute the simulation and observe system evolution. The primary visualization emphasizes information flow — messages moving between nodes — rather than static node states.

Information Flow Visualization

The core visual metaphor is packets (or cubes) moving along edges, representing messages in transit between nodes. This makes the dynamics of the system immediately legible:

- **Packet size:** Volume or importance of information being transmitted
- **Packet speed:** Frequency of communication on that edge
- **Packet direction:** Which way information flows (shown by movement animation)
- **Edge thickness:** Cumulative traffic or connection strength

This visualization works across mechanism types: market price signals, network gossip, democratic votes, or epidemic contagion all appear as information flowing through the graph structure.

Execution Controls

- **Play/Pause:** Toggle automatic timestep advancement
- **Step Forward:** Advance one timestep
- **Step Backward:** Return to previous timestep (loads stored snapshot)
- **Timeline Scrubber:** Jump to any computed timestep
- **Speed Control:** Adjust playback rate
- **Reset:** Return to initial state (T=0)

State Snapshots and Timeline Navigation

To enable backward navigation and timeline scrubbing, the system stores a complete state snapshot after each timestep. Forward execution uses JIT-compiled transformations for performance; backward navigation simply loads a previously stored snapshot. This separation means:

- Forward execution is computationally optimized
- Timeline scrubbing is instantaneous (no re-computation)
- Any timestep's full graph state can be inspected or exported

Extensible Visual Encoding

Node and edge visual properties (color, shape, size) are configurable through the Visualization Service. Default encodings exist (e.g., mechanism types distinguished by shape) but researchers can override these to highlight domain-specific attributes. The information flow visualization (packets on edges) is the primary, stable visual; node styling is secondary and customizable.

2.6 Live Metrics & Export

Metrics and export are not a separate phase but panels available throughout simulation. Users can open the metrics panel while the simulation runs, watching charts update in real-time. Export is available at any point — mid-simulation or after completion.

Live Metrics Panel

The metrics panel displays time-series charts that update as each timestep completes. Users can select which metrics to display from those registered with the Metric Collector. Charts support dual Y-axis for metrics with different scales.

Example metrics (extensible via Metric Registry):

- Total resources across agents
- Average trust level across edges
- Gini coefficient (resource inequality)
- Network density
- Belief variance (polarization measure)
- Decision accuracy vs ground truth

Export at Any Timestep

Users can export from any point in the timeline — the current timestep, a selected past timestep, or the full history. Available exports include:

- **Graph State (JSON/YAML):** Complete node, edge, and attribute data for selected timestep
- **Scenario File:** The design configuration (can be re-loaded to reproduce)
- **Metrics CSV:** Time-series data for all tracked metrics
- **Full History JSON:** Complete state at every timestep
- **Visualization Export:** SVG, PNG, or PDF of current view

The graph export includes all metadata: node attributes, edge weights, mechanism configurations, and verification reports. This enables external analysis tools to work with the full state.

3. System Architecture

3.1 Architectural Principles

The system follows a service-oriented architecture with clear module boundaries. Each service has a single responsibility, communicates through well-defined interfaces, and can be developed and tested independently.

- **Modularity:** Each subsystem is independently deployable and replaceable
- **Registry Pattern:** New capabilities (mechanisms, properties, metrics) are registered rather than hardcoded
- **Scenario as Contract:** The scenario file is the portable artifact that flows between services
- **Verification Layer:** Runtime property checking rather than compile-time type enforcement
- **Snapshot History:** Full state stored at each timestep, enabling timeline navigation and any-point export

3.2 Service Decomposition

Service	Responsibility
Canvas Service	Renders graph canvas, handles user interactions, manages visual state. Queries Mechanism Registry for toolbar contents. Exports to scenario format.
Scenario Service	Manages scenario files (load, save, validate). Provides scenario templates. Handles import/export formats.
Mechanism Registry	Stores registered element types with metadata (icon, color, property schema, transformation). Populates canvas toolbar dynamically.
Transformation Engine	Executes mechanism transformations on graph state. Handles composition and scheduling. Produces new state per timestep. Stores snapshots.
State Store	Maintains snapshot history for all computed timesteps. Enables timeline navigation and any-timestep export.
Property Registry	Stores registered verification properties. Provides property definitions for verification service.
Verification Service	Wraps transformations with pre/post metric capture. Validates properties. Reports violations.
Metric Collector	Computes aggregate metrics from graph state. Maintains time-series history. Supports custom metric registration.
Visualization Service	Renders information flow (packets on edges), node styling, metrics charts. Handles export to image formats. Supports custom visual encodings.
Session Orchestrator	Coordinates workflow across services. Manages session state. Handles play/pause/step/seek commands.

3.3 Data Flow

The scenario file serves as the central data contract. All services read from or write to this format, enabling loose coupling between components.

Design Flow

4. Canvas Service queries Mechanism Registry for available element types
5. User manipulates canvas (place, connect, configure)
6. Canvas exports to scenario format; Scenario Service validates
7. User saves scenario file for later use or sharing

Execution Flow

1. Session Orchestrator loads scenario
2. Orchestrator requests timestep from Transformation Engine
3. Engine retrieves mechanism definitions from Mechanism Registry
4. Verification Service wraps each transformation with property checks
5. Engine executes transformations; State Store saves snapshot
6. Metric Collector captures state metrics
7. Visualization Service renders information flow and updates live metrics
8. User can seek to any timestep (loads from State Store)

Export Flow

1. User selects timestep (or current) for export
2. State Store provides complete graph state for that timestep
3. Scenario Service formats output (JSON, YAML, CSV as appropriate)
4. Visualization Service exports images if requested

4. Extension Points

The system is designed for extensibility. New capabilities are added through registration rather than modification of core services. This enables researchers to extend the laboratory without disrupting existing functionality.

4.1 Mechanism Registry

New coordination mechanisms are registered with the Mechanism Registry. Each registration provides:

- **Identifier:** Unique name for the mechanism type
- **Display metadata:** Icon, color, label for canvas toolbar
- **Property schema:** Configurable parameters (rendered in property panel)
- **Transformation definition:** How the mechanism transforms graph state
- **Default schedule:** Initial activation frequency and latency

Default registrations include Network (information diffusion), Market (price discovery), and Democracy (voting aggregation). These serve as examples; the system is mechanism-agnostic.

4.2 Property Registry

Verification properties are registered with the Property Registry. Each registration provides:

- **Identifier:** Unique name for the property
- **Invariant definition:** The condition that must hold (expressed as pre/post comparison)
- **Tolerance:** Acceptable deviation for numeric properties
- **Severity:** CRITICAL (halt), HIGH (warn and flag), INFO (log only)
- **Applicable mechanisms:** Which mechanism types this property applies to

4.3 Metric Registry

Custom metrics are registered with the Metric Collector. Each registration provides:

- **Identifier:** Unique name for the metric
- **Computation:** Function from graph state to numeric value
- **Display metadata:** Label, unit, suggested scale (linear/log)

4.4 Visual Encoding Registry

Custom visual encodings can override default styling. Each registration provides:

- **Target:** Which element type or attribute this encoding applies to
- **Mapping:** Function from attribute value to visual property (color, size, shape)
- **Legend entry:** How to display in visualization legend

4.5 Agent Behavior Registry

For advanced users, custom agent behaviors can be registered. Each registration provides:

- **Identifier:** Unique name for the behavior type
- **Decision function:** How the agent responds to messages and mechanism opportunities

- **State schema:** Internal state the agent maintains

This enables integration with external agent frameworks (LLM agents, RL agents, active inference agents) while maintaining the laboratory's visual design paradigm.

5. Scenario Schema

The scenario file is the portable artifact that captures a complete design. It can be version controlled, shared between researchers, and loaded into the laboratory or executed via command line.

5.1 Top-Level Structure

Section	Contents
metadata	Name, description, author, version, created/modified timestamps
config	Global simulation parameters: rounds, random seed, custom globals
nodes	List of nodes with IDs, types (from registry), positions, and type-specific attributes
edges	List of connections with source, target, type, and edge-specific attributes
mechanisms	Configuration for each mechanism node: type, parameters, schedule
verification	Which properties to verify, severity overrides, tolerance settings
metrics	Which metrics to track, custom metric definitions
visual	Custom visual encoding overrides for this scenario

5.2 Node Schema

Each node entry contains:

- **id:** Unique string identifier
- **type:** Registered type from Mechanism Registry
- **position:** Canvas coordinates [x, y]
- **attributes:** Type-specific properties as defined by registry schema

5.3 Mechanism Schedule Schema

Each mechanism has an associated schedule:

- **frequency:** Activation interval in timesteps
- **latency:** Processing delay before results propagate
- **windows:** List of [start, end] timestep ranges when mechanism is available
- **priority:** Execution order when multiple mechanisms activate simultaneously

6. Verification Framework

The verification framework provides runtime property checking. This design choice optimizes for research agility — researchers can modify transformations without fighting a type system — while maintaining safety guarantees through explicit runtime validation.

6.1 Verification Process

Every transformation is wrapped with verification:

1. Capture pre-transformation metrics from current state
2. Execute transformation, producing new state
3. Capture post-transformation metrics from new state
4. For each applicable property, evaluate invariant against pre/post metrics
5. Produce verification report with any violations
6. Based on severity: halt (CRITICAL), warn (HIGH), or log (INFO)

6.2 Example Properties

The following are illustrative properties that might be registered. Actual properties depend on the mechanisms in use:

- **Conservation:** A quantity (resources, voting power, etc.) remains constant across transformation
- **Boundedness:** Values stay within expected range (e.g., beliefs in [0,1])
- **Acyclicity:** A graph structure (e.g., delegation) remains a DAG
- **Connectivity:** Network remains connected (no isolated components)
- **Monotonicity:** A value only increases (or only decreases)

6.3 Verification Report

Each timestep produces a verification report containing: timestep number, which transformation executed, execution duration, pre/post metrics, any violations with details, and warnings. The UI surfaces CRITICAL violations as blocking dialogs; HIGH/WARN as dismissible banners.

7. Functional Requirements Summary

7.1 Canvas Requirements

- Toolbar populated dynamically from Mechanism Registry
- Drag-and-drop placement of all registered element types
- Edge creation via click-and-drag between nodes
- Multi-Connect Tool for many-to-one edge creation
- Property panel rendering type-specific schemas from registry
- Zoom and pan navigation
- Undo/redo for all canvas operations

7.2 Simulation Requirements

- Discrete timestep execution with configurable count
- Play/pause/step-forward/step-backward controls
- Timeline scrubber accessing stored state snapshots
- State snapshot stored after each timestep
- Mechanism scheduling with configurable frequency and latency
- Verification on every transformation
- Random seed specification for reproducibility

7.3 Visualization Requirements

- Information flow visualization: packets on edges with size/speed encoding
- Extensible visual encoding through registry
- Live metrics panel updating during simulation
- Time-series metrics chart with configurable metric selection
- Export to SVG, PNG, PDF formats

7.4 Export Requirements

- Export available at any timestep (not just end of simulation)
- Full graph state export (JSON/YAML) with all attributes and metadata
- Scenario file export for reproducibility
- Metrics time-series export (CSV)
- Full history export (all timesteps)

7.5 Extension Requirements

- Registration interface for custom mechanisms
- Registration interface for custom properties
- Registration interface for custom metrics
- Registration interface for custom visual encodings
- Registration interface for custom agent behaviors

8. Appendix: Wireframe Reference

The accompanying wireframes illustrate one possible configuration of the laboratory. They show example mechanism types (Democracy, Market), example visual encodings, and example scenarios. These are prototypical instances demonstrating capabilities, not fixed requirements.

Wireframe	Illustrates
Initial_Pictures.png	Welcome screen, example scenario cards, global configuration sliders. Scenarios shown are examples.
Graph_Initialisation.png	Canvas with example toolbar (registry-populated in practice), example graph topology. Multi-Connect Tool use case visible.
Step_5_Run.png	Timestep grid view. Information flow visualization (packets on edges) is the primary visual. Node colors are example encodings.
Step_6_Export.png	Live metrics panel (available during simulation). Metrics shown are examples from Metric Registry.

Design Principles Recap

- **Examples, not prescriptions:** Wireframes show one configuration; the system supports many
- **Registry-driven extensibility:** New capabilities added via registration, not modification
- **Information flow as primary visual:** Packets moving on edges, not just static node states
- **Snapshot history:** Full state at each timestep enables timeline navigation and any-point export
- **Live metrics:** Analysis panels available during simulation, not just after
- **Scenario as portable artifact:** Version-controllable, shareable, reproducible

— End of Specification —