

2025-11-01

section 0

[2em] Part Overview · Duration:

Beginner-Friendly Visual Study Guide

subsection 0.0 Episode Purpose

Not Describing Diagrams: This episode doesn't describe every arrow/box (audio is sequential, diagrams are spatial).

Selling the Value: Explains WHY diagrams matter, WHAT they show, WHERE to find them.

Think Trailer: Motivation to explore docs/diagrams/ yourself.

subsection 0.0 Why Diagrams Matter

- **Flow Visualization:** Data moving like water through pipes (config -> validation -> controllers -> dynamics)
- **Gestalt Understanding:** See everything at once (vs. sequential code reading)
- **Connection Mapping:** How functions connect (the plumbing), not just what each function does

section 0 Project Root Structure

subsection 0.0 Root Level (18 Visible Items)

Category	Files/Directories
Core Files (9)	simulate.py, streamlit_app.py, config.yaml, requirements.txt README.md, CHANGELOG.md, CLAUDE.md, package.json, package-lock.json
Core Directories (8)	src/, tests/, docs/, benchmarks/, scripts/ envs/, optimization_results/, data/
Hidden Dirs (9)	.git/, .github/, .ai_workspace/, .cache/, .pytest_cache/ .hypothesis/, .ruff_cache/, .mypy_cache/, .venv/

subsection 0.0 Entry Points

- **simulate.py:** Command-line simulation runner
- **streamlit_app.py:** Web UI interface
- **config.yaml:** Centralized configuration

section 0 src/ Architecture

subsection 0.0 Layer 1: Core (Foundation)

- **core/:** Simulation context, state management, base interfaces
- **plant/:** Dynamics models (simplified, full nonlinear, low-rank)
- **simulation/:** Simulation runner, execution logic

Key Principle: Everything depends on these. They depend on nothing else (foundational layer).

subsection 0.0 Layer 2: Controllers & Optimization

- **controllers/:** 7 SMC variants + factory pattern
- **optimization/:** PSO tuner (48 files, 1.4 MB)
- **optimizer/:** Backward compatibility shim (re-exports from optimization/)

Factory Pattern: Request "classical_smc", factory instantiates correct class.

subsection 0.0 Layer 3: Infrastructure

- **utils/:** Validation, logging, monitoring, visualization (40,000 lines)
- **interfaces/:** HIL testing, abstract base classes

- **config/**: Configuration loading, validation, defaults

subsection 0.0 Layer 4: Specialized Features

- **benchmarks/**: Performance measurement tools
- **analysis/**: Statistical analysis, Monte Carlo aggregation
- **hil/**: Hardware-in-the-loop plant server + controller client

section 0 Key Architectural Patterns

subsection 0.0 1. Compatibility Layers

Example: `optimizer/` and `optimization/`

- **Legacy:** Early code used from `src.optimizer` import `PSOTuner`
- **Refactor:** Moved to modular `src.optimization/`
- **Shim:** `src.optimizer/` re-exports from `src.optimization/`
- **Reason:** Avoid breaking existing scripts during transition

Documentation: Section 25 of CLAUDE.md establishes this as intentional (not duplication).

subsection 0.0 2. Re-export Chains

Example: `simulation_context.py` in 3 locations

- **Primary:** `src/core/simulation_context.py`
- **Re-exports:** `src/simulation/` and `src/plant/`
- **Reason:** Import path flexibility (users can import from any location)

subsection 0.0 3. Model Variants

8 Dynamics Files: Different accuracy/performance tradeoffs

- **Simplified:** Fast prototyping (linearized model)
- **Full Nonlinear:** Research accuracy (complete physics)
- **Low-Rank:** Real-time applications (approximations)
- **Interface:** All implement `DynamicsInterface` (plug-and-play)

subsection 0.0 4. Framework Files

Example: `src/interfaces/hil/test_automation.py`

- **Confusion:** Name suggests test file
- **Reality:** Production framework for HIL test automation
- **Location:** Correctly in `src/` (production code, not tests/)

section 0 Control Flow Visualization

subsection 0.0 Simulation Execution Path

- **1. Config Entry:** User edits `config.yaml`
- **2. Validation:** Pydantic validates all parameters (catch errors pre-runtime)

- **3. Splitting:** Config splits into controller settings + dynamics parameters
- **4. Factory:** Controller factory instantiates requested controller type
- **5. Simulation Loop:**
 - Dynamics model computes next state
 - Controller receives state, computes control signal
 - Control signal fed back to dynamics
 - History logged to monitoring system
- **6. Results:** Visualization, analysis, export

subsection 0.0 PSO Optimization Flow

- **1. Swarm Init:** 50 particles, random initial positions in parameter space
- **2. Iteration Loop (100 iterations):**
 - Each particle = candidate gain set
 - Run full simulation with candidate gains
 - Compute cost function (IAE, chattering, energy)
 - Update particle velocity based on personal best + global best
 - Move particle to new position
- **3. Convergence:** Return global best after 100 iterations
- **4. Output:** Save optimized gains to JSON file

section 0 Diagram Locations

subsection 0.0 Available Diagrams

Diagram	Location
Project Structure	docs/diagrams/project_structure.svg
Control Flow	docs/diagrams/control_flow.svg
PSO Optimization	docs/diagrams/pso_workflow.svg
Controller Factory	docs/diagrams/factory_pattern.svg
HIL Architecture	docs/diagrams/hil_setup.svg
Monitoring System	docs/diagrams/monitoring_flow.svg

subsection 0.0 How to Use Diagrams

- **1. Start Broad:** Project structure diagram (understand 4 layers)
- **2. Follow Data:** Control flow diagram (trace execution path)
- **3. Zoom In:** Controller factory (understand instantiation)
- **4. Cross-Reference:** Match diagram nodes to code files

section 0 tests/ Directory Mirroring

subsection 0.0 Peer File Structure

Rule: Every `src/*.py` has a `tests/test_*.py` peer.

```

lstnumbersrc/
lstnumber  controllers/
lstnumber  classical_smc.py
lstnumber  sta_smc.py
lstnumbertests/
lstnumber  test_controllers/
lstnumber  test_classical_smc.py  # Peer for classical_smc.py
lstnumber  test_sta_smc.py        # Peer for sta_smc.py

```

subsection 0.0 Benefits

- Easy navigation (predictable test location)
- Coverage tracking (identify untested files)
- Parallel structure (mirrors production architecture)

section 0 docs/ Documentation

subsection 0.0 Documentation Scale

- **Total Files:** 985 (814 in docs/, 171 in .ai_workspace/)
- **Navigation Systems:** 11 (NAVIGATION.md is master hub)
- **Category Indexes:** 43 across all domains
- **Learning Paths:** 5 (Path 0: 125-150 hrs -> Path 4: 12 hrs)

subsection 0.0 Key Documentation Entry Points

Entry Point	Purpose
docs/NAVIGATION.md	Master hub connecting all 11 navigation systems
docs/index.md	Sphinx documentation root
docs/guides/INDEX.md	Quick-start guides
README.md	Project overview (root level)

section 0 Key Takeaways

subsection 0.0 Architectural Principles

- **1. Layered Design:** Core (L1) -> Controllers/Optimization (L2) -> Infrastructure (L3) -> Specialized (L4)
- **2. Intentional Patterns:** Compatibility layers, re-export chains, model variants (documented in CLAUDE.md Section 25)
- **3. Interface Abstraction:** Swap implementations without changing dependents
- **4. Peer File Structure:** Every src/ file has tests/ peer

subsection 0.0 Where to Go Next

- **Explore Diagrams:** docs/diagrams/ directory
- **Read Architecture Docs:** docs/architecture/
- **Check Code Examples:** docs/examples/
- **Review Navigation Hub:** docs/NAVIGATION.md

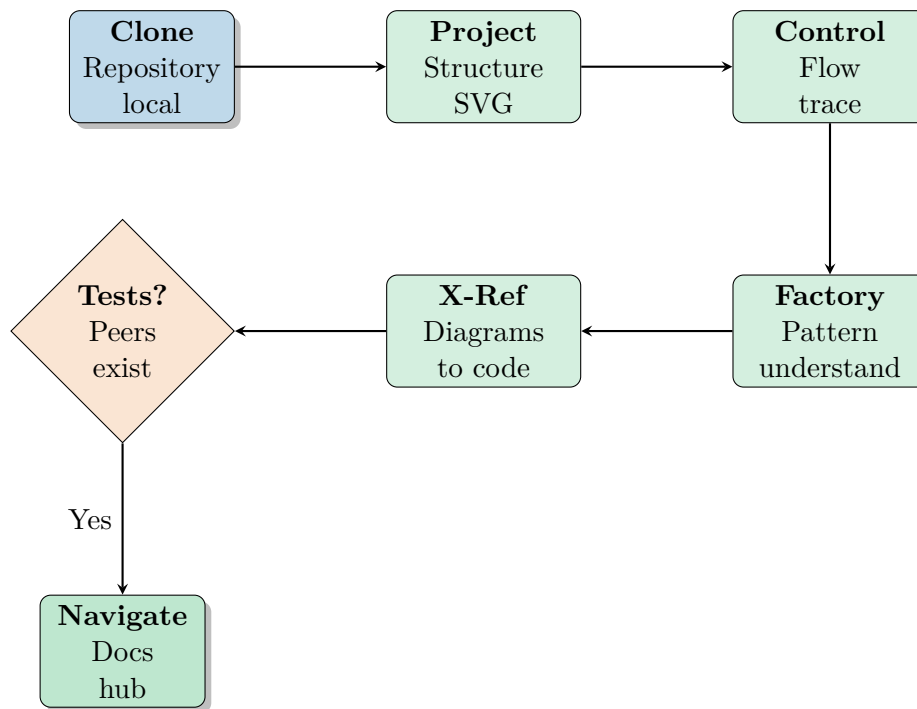
subsection 0.0 The Gestalt Principle

Audio Limitation: Sequential description loses spatial relationships.

Visual Advantage: Diagrams show connections at-a-glance.

Action: Open docs/diagrams/, start with project_structure.svg, follow the flow.

Checklist: Explore Visual Documentation



- ☐ **Clone Repository:** Get local copy of diagrams
- ☐ **Project Structure:** Open docs/diagrams/project_structure.svg
- ☐ **Control Flow:** Trace execution path in control_flow.svg
- ☐ **Factory Pattern:** Understand controller instantiation
- ☐ **Cross-Reference:** Match diagram nodes to src/ files
- ☐ **Test Peers:** Verify every src/*.py has tests/test_*.py
- ☐ **Documentation Hub:** Explore docs/NAVIGATION.md
- ☐ **Architecture Guide:** Read CLAUDE.md Section 25 (standards)

Next Steps

- **E024:** Lessons learned and best practices (6-month retrospective)
- **E025-E029:** Appendix reference (5-part technical deep dive)