Neo-CYCLONE

AI-ASSISTED CONSTRUCTION OPERATION SIMULATION

User Manual for students, instructors, and practitioners who want to understand construction operations as *flow*— queues, meetings, idleness, and production— in the spirit of Daniel W. Halpin's CYCLONE.

Version 1.7.2
Live studio https://neo-cyclone.vercel.app/
Language English
References at the end of this document

Neo-CYCLONE — User Manual

Version	1.7.2
Product	AI-Assisted Construction Operation Simulation
Live app	https://neo-cyclone.vercel.app/
Language	English (product & teaching surface)
Audience	Students, instructors, and practitioners
Notation detail	NOTATION_STANDARD.md

This manual is written for people—not for scrapers. Read it with the studio open in another tab. You can finish the first run in about fifteen minutes (Chapter 2), or take an hour for the full story. **References** are at the **end**.

Prologue — Why this studio exists

Construction work is full of **repetition**: load and haul, pour and return, lift and place. Between those busy moments, resources wait. A truck sits at a loader. A crane waits while a crew finishes tying rebar. That waiting is not always “laziness.” Very often it is **the structure of the process**—how work and idle time are braided together on a real site.

Professor **Daniel W. Halpin** spent a career making that structure *visible*. His **CYCLONE** language (*CYCLic Operations NEtwork*) gave construction operations a small, honest network grammar: resources wait in queues, they work for a duration, they meet when they must, and a counter records completed production. From that grammar grew a family of tools—**MicroCYCLONE**, then DISCO, PROSIDYC, COST, WebCYCLONE, and related systems. The full reading list sits in **References** at the end of this manual.

Neo-CYCLONE does not claim to replace those research systems. It is a **teaching studio**: a browser place to meet Halpin’s ideas again, with a modern interface and an AI Assistant that stays tied to *your* model. The product name is plain on purpose: **AI-Assisted Construction Operation Simulation**.

AI here does not invent a new physics of construction. The **engine** is still a discrete-event, CYCLONE-style simulator. AI helps you phrase a model, inspect a diagram, and ask questions about the last run. **You** still click Draw Model, still Simulate, still judge whether the story is true.

If you leave with one habit, make it this:

> **Draw the cycles until they tell the truth, then run the numbers.**

Chapter 1 — Operations, flow, and idleness

1.1 What we mean by a construction operation

An *operation* in this studio is a **repeatable production process**—often measured in units per hour—not the whole project Gantt chart. Earthmoving a cut, paving a lane, loading dump trucks, serving three zones with one crane, stocking brick and mortar for masons, cycling forms in a precast yard: each is an operation with **resources**, **tasks**, and **waiting**.

Thinking at the operation level matters for **Lean Construction** and **Project Production Management**. Before you “optimize” a schedule bar, you need to see whether the *process* itself produces flow or waste. Neo-CYCLONE is built for that first, stubborn look.

1.2 Flow and idleness — waste you can measure

In CYCLONE thinking, a resource that is not working is usually **in a queue**—waiting for a partner, a space, or a task to open. That waiting time is **idleness**. It is not a moral failure; it is a signal:

- Too few trucks → the loader sits idle.
- Too many trucks → the truck queue grows and money burns in the line.
- A shared crane with the wrong priority → one zone starves while another looks busy.

Neo-CYCLONE reports **idle %** and **busy %** side by side so those signals are hard to ignore. When you later hear “waste” in Lean language, you already have a picture for it—not a slogan.

1.3 Why a network language?

You could describe an operation in paragraphs of natural language. Networks force the questions that paragraphs hide:

- Which resources exist?
- In what order do they work?
- Where do they wait when they are not working?
- Where do two or more resources **meet** (for example truck and loader)?
- What counts as **one unit of production**?

CYCLONE answered those questions with a small set of node types. Neo-CYCLONE keeps that spirit in the **logic**, even when some **glyphs** and **arrow colors** are tuned for a screen (Chapter 4). Logic first; ink second.

1.4 What Neo-CYCLONE is — and is not

IT IS	IT IS NOT
A browser teaching studio for cyclic construction operations	A full project-controls ERP
Prompt → diagram → discrete-event simulation	A black-box “AI that simulates for you” without a model
MicroCYCLONE-style reports (process, elements, cost, sensitivity)	A pixel-perfect reprint of 1970s or 1990s paper figures
English-first, classroom-friendly limits	An unlimited free chat API

You do **not** need to sign in to learn. Teaching use stands alone.

1.5 The studio at a glance

AREA	WHAT YOU DO THERE
Left	Choose an Example or write a Format Prompt, then **Draw Model**
Right	Inspect the **CYCLONE Model** diagram; set cycles and seed; **Simulate**
Below	**Results** — Simulation and Sensitivity Analysis
Lower	**AI Assistant** (optional co-pilot)
Header	This Manual
Footer	Product name, version, year — cite the version in homework

The right-hand diagram is not decoration. Chapter 4 is the legend for that panel.

Chapter 2 — Fifteen minutes that stick (Earthmoving)

This chapter is a guided first run. Do it once with Example 1 even if you already “know trucks.” Muscle memory for the studio matters more than cleverness on day one.

2.1 Open the studio

Go to the live app. You should see a prompt area and Example dropdown on the left, an empty **CYCLONE Model** on the right until you draw, and later—after a run—Results and the AI Assistant below. The footer shows product name, version, and year.

2.2 Load Example 1 and draw the model

1. Open **Example** → **1. Earthmoving**.
2. Read the prompt top to bottom. Notice the shape:

```
# notes (ignored by the engine) …

Operation: Earthmoving

Trucks: Load → Haul → Dump → Return
Loader: Load
5 trucks, 1 loader

Counter after: Dump
production = 12 m3

Cost: …
Durations: …
```

In plain language: trucks cycle load → haul → dump → return. The loader only joins at **Load**, so Load is a **meeting** (COMBI). Production is counted after **Dump** (for example 12 m³). Costs are dollars per resource-hour. Durations are in **minutes**.

1. Click **Draw Model**. Selecting an example does **not** draw by itself.
2. On the right, confirm what Chapter 4 will name carefully:

- Home **QUEUE** circles for trucks and loader (often with `n = …`).
- **Load** as COMBI (square with a top-left cut).
- Haul, Dump, Return as NORMAL rectangles (truck alone).
- **COUNTER** as a golf-flag after Dump.
- **Solid black** arrows forward; **dashed gold** arrows returning resources home.

If something looks wrong, fix the prompt and **Draw Model** again. Do not Simulate until the picture matches the story. That pause is the point of the studio.

2.3 Cycles, seed, and a fair comparison

- **Max cycles** — default **100**, hard maximum **500** (a teaching cap).
- **Seed** — default **12345**. Same seed, same model, same cycle limit → **identical** stochastic results. The dice button picks another seed when you *want* a different random path.

Seed is for **reproducibility**—homework, papers, fair classroom comparison—not a fleet decision variable. If your classmate “got different productivity,” check seed and max cycles before you rewrite the model.

2.4 Simulate and read the first results

Click **Simulate**. Open the **Simulation** tab and walk the story in order:

1. **Process Report** — run length, cycles, production pace.
2. **Units per hour by cycle** — does productivity settle? The steady-state guide (~5% over at least 10 cycles) appears as an old-gold dashed line.
3. **Resource idleness** — who waits? who works? Often the best classroom discussion in the whole app.
4. **Cost Report** (if you entered rates) — unit cost bridges “how busy?” to “how expensive per unit?”

Chapter 5 expands each of these without turning them into a checklist of empty headings.

2.5 Export and the AI Assistant

Export **Report Excel** (the filename prefers `Operation: …`), chart PNG, and diagram PNG when you need evidence for homework. Record seed, max cycles, and the operation name.

If you open the **AI Assistant**, try the general chips: resources, bottleneck, productivity, unit cost. Answers stay short. If the assistant proposes a new prompt, nothing changes in the engine until you **Apply**, then **Draw Model**, then **Simulate**. That three-step gate is intentional.

2.6 Mistakes that waste a first session

1. Expecting a diagram after only selecting an Example.
2. Simulating before the diagram matches the story.
3. Changing fleet size in chat and assuming the engine already changed.
4. Comparing runs with different seeds.
5. Reading only total production and ignoring **idleness**.
6. Assuming every rectangle must match a photocopy of a 1992 book figure (see Chapter 4).

Chapter 3 — How to talk so the studio can build a network

You do not draw QUEUE circles freehand in the prompt. You describe **resource cycles**. The builder creates queues, tasks, and arcs. Students learn the *logic* of cycles—not pixel-pushing.

3.1 Notes, operation name, and resource cycles

Lines that start with `#` or `//` are **notes only**. Use them for teaching context; the engine ignores them.

After the notes, the first data line may be:

```
Operation: Earthmoving
```

Aliases: `Model:`, `Title:`, `Op:`. That name titles reports and Excel files. All six built-in Examples place `Operation:` after their comment block.

Then state the network:

```
Trucks: Load → Haul → Dump → Return
Loader: Load
5 trucks, 1 loader
```

- One primary sequence per resource.
- Supporting resources often share a meeting task (here, Load).
- Arrows may be written `→`, `->`, `-->`, or `=>`.
- Multi-demand (one resource serves several tasks): `Crane: LiftAtA | LiftAtB | LiftAtC`.
- Priority (lower number = higher priority when several demands wait) follows MicroCYCLONE tradition:

```
Priority:
LiftAtA: 1
LiftAtB: 2
LiftAtC: 3
```

3.2 Production counter

```
Counter after: Dump
production = 12 m3
```

Name the task (or tasks) that mean “one production unit finished.” Multiple counters are allowed—for example lifts at three tower-crane zones. Explicit counters are safer for teaching than silent defaults. If production “disappears,” check this line first.

3.3 Durations (minutes)

Every named task needs a distribution. Time in this studio is **minutes** unless you deliberately document otherwise.

KIND	PARAMETERS	TYPICAL USE
`const`	value	Deterministic demo
`unif`	min, max	Flat uncertainty
`tri`	min, mode, max	Common field estimate
`normal`	mean, sd	Symmetric scatter
`lognormal`	mean, sd	Skewed positive times
`beta`	min, max, α, β	Four-parameter beta
`pert`	a, m, b	Classic PERT-beta on [a, b]
`gamma`	shape, scale	Flexible positive skew

Aliases: `pert` ≈ beta-PERT. A three-number `beta` is treated as PERT.

3.4 Branch probability, GEN, and CON

When the story forks—breakdown, rework, inspection fail—use a branch:

```
Branch:
After DumpToPaver: RefillAsphalt p=0.85, Breakdown p=0.15
```

Probabilities should look like a split of the real world, not decoration.

GENERATE multiplies entities (one truck arrival becomes five scoop-sized loads). **CONSOLIDATE** gathers *n* into one (truck becomes full). Prefer the **inline** form on the cycle:

```
Trucks: GEN 5 → Scoop → CON 5 TruckFull → Haul&Return
```

Not every model needs GEN/CON—only when the production unit logic requires scaling. Their independence is intentional: you may have one without the other.

3.5 Cost, sensitivity, and block order

```
Cost:
Trucks: 85
Loader: 120

Sensitivity:
Trucks: 2..10
Loader: 1..2
```

Costs are **USD per resource-hour**. Sensitivity varies counts for comparison runs (productivity, unit cost, idleness). Teaching caps: up to **five** resources in SA; combinations limited (~150) by stepping ranges, not by silently dropping mid-axis points.

Recommended order for humans and for the assistant:

Operation → Network → Durations → Priority → Branch → Cost → Sensitivity (last).

3.6 A minimal custom prompt you can rewrite

```
Operation: My Operation Name

ResourceA: Task1 → Task2 → Task3
ResourceB: Task1
3 ResourceA, 1 ResourceB

Counter after: Task3
production = 1 unit

Durations:
Task1: tri 1, 2, 3
Task2: normal 8, 1.5
Task3: const 1

Cost:
ResourceA: 80
ResourceB: 120
```

Replace the names with *your* operation. Draw Model. If the diagram lies, the prompt is incomplete—not “the AI failed.” The live **Format Prompt** panel in the studio shows the canonical template; prefer that order so people and software read the same story.

Chapter 4 — Reading the model: how Neo-CYCLONE draws CYCLONE

Users spend a long time staring at the **CYCLONE Model** panel. This chapter is the legend for that panel: how we model, what each symbol means, and where we deliberately differ from textbook Halpin figures while keeping the same *ideas*.

4.1 Modeling idea (the same spirit as Halpin)

Neo-CYCLONE still models a **cyclic construction operation** as:

1. **Resources** that wait in **queues** when idle.
2. **Work** that consumes resource units for a duration.
3. **Meetings** when two or more resources must be present to start work.
4. **Returns** of each resource to its home idle pool so the cycle can repeat.
5. A **counter** (or counters) that record completed production units.
6. Optional **functions** that scale entities (GEN / CON) and optional **probabilistic branches**.

You never draw that by hand in the prompt. You state resource cycles in text; the studio **builds** the network:

story in Format Prompt → Draw Model → inspect diagram → fix story → Simulate.

4.2 Node shapes on screen

ELEMENT	SHAPE IN NEO-CYCLONE	MEANING
QUEUE	Circle with a lower-right slash (reads like a **Q**)	Waiting / idle pool. **Home** queues hold initial units (`n = …`)
COMBI	Square with **top-left corner cut**	Work that needs **≥2 resources** meeting
NORMAL	Plain rectangle	Work that needs **one** resource unit stream
COUNTER	**Golf flag** (pole + triangle flag)	Production count (+units when the flag is passed)
GEN	**Inverted triangle** (point down)	On arrival, create **k** units (scale up)
CON	**Upright triangle** (point up)	Gather **n** units, release 1 (scale down)

Labels under shapes typically show initial `n`, duration text, `GEN k`, `CON n`, or `+production`.

4.3 Arrows — direction always matters

STYLE	APPEARANCE	MEANING
Forward	**Solid black** line + black arrowhead	Work progresses (including into staging queues before a COMBI)
Return	**Dashed gold** line + gold arrowhead (often curved)	Resource closes its cycle into a **home** QUEUE only
Branch	Forward style, often with **`p=…`**	Probabilistic choice among outs

When you read a diagram: follow **black** to see how production moves; follow **gold dashed** to see how each resource **goes home** to wait again. If gold dashed points at something that is not an idle home pool, the model is suspicious—re-draw after fixing the prompt.

4.4 COMBI versus NORMAL

Ask one question: *Do two or more distinct resources have to be present for this task to start?*

SITUATION	NODE
Truck **and** loader both needed at Load	**COMBI**
Truck alone hauls or returns	**NORMAL**
Crane lift that also needs a crew at the hook	**COMBI**
Crew works alone after material is placed	**NORMAL**

If the diagram shows COMBI for a solo task, your prompt probably listed two resources on the same step by accident—or the reverse if a true meeting was written as a single-resource line.

4.5 A resource cycle as a mental picture

For a truck in earthmoving, the diagram encodes roughly:

1. Sit in **Trucks Idle** (QUEUE, `n = 5`).
2. Enter **Load** (COMBI) with a loader unit.
3. **Haul** → **Dump** → **Return** (NORMAL steps).
4. Pass **COUNTER** after Dump when production is counted.
5. **Gold dashed** arc back to **Trucks Idle**.

The loader has a shorter cycle: idle → Load (COMBI) → gold return home. Once you can tell that story out loud while pointing at the screen, you understand the model.

4.6 Same spirit as CYCLONE — different surface

Neo-CYCLONE is **loyal to Halpin's logic**, not always to the **exact ink** of every textbook figure. Users who open Halpin & Riggs (or MicroCYCLONE printouts) side by side with the studio will notice differences. That is intentional for **screen teaching**.

TOPIC	CLASSIC CYCLONE / MICROCYCLONE (TYPICAL PRINT)	NEO-CYCLONE (THIS STUDIO)
Purpose	Methodology + desktop / research tools	Browser **teaching** studio + AI co-pilot
How you build	Often node/link editors, cards, or input files	**Format Prompt** (resource cycles) → auto layout
QUEUE look	Circle (sometimes plain)	Circle with **Q-like slash**
COMBI look	Square / constrained conventions vary by book era	Square with **top-left cut**
COUNTER look	Flag-like or marked node in teaching materials	Explicit **golf-flag** icon
GEN / CON	Function nodes in full systems	▽ **GEN** / △ **CON**; prefer **inline** in the prompt
Arrows	Usually black linework; returns not always color-coded	**Black solid = forward**, **gold dashed = return home**
Layout	Author-drawn publication figures	Automatic teaching grid
AI	None historically	Context-bound Assistant (**Apply** required)
If figures disagree	Printed book / original software	**This app's legend + engine** (`NOTATION_STANDARD.md`)

What must stay the same for the model to still "be CYCLONE": resources wait in queues; work takes time and holds units; meetings need all required resources; cycles close so production can repeat; counters define the production unit.

What may look different on purpose: colors and dashes on return arcs; exact corner cuts and flag art; automatic layout; prompt-first authoring.

If you write a paper, say you used **Neo-CYCLONE's teaching notation** inspired by Halpin CYCLONE—not that a screenshot is a facsimile of Figure X in the 1992 book.

4.7 Checklist before you Simulate

1. Every resource has a visible **home QUEUE**.
2. True meetings are **COMBI**; solo work is **NORMAL**.
3. **Counter after:** names match real tasks.
4. Gold dashed returns only into home idles.
5. GEN/CON only if unit logic needs them.
6. Branch probabilities look like a real split of the world.
7. `Operation:` is set if you care about Excel and report names.

Full geometric rules live in `docs/NOTATION_STANDARD.md`.

Chapter 5 — Simulation results

After **Simulate**, the Results area is for process literacy—not only a green checkmark. Walk the panels in the order below the first few times; later you will jump to the question you care about.

5.1 Process Report

This is the MicroCYCLONE-flavored summary: how long the run lasted, how many production events occurred, units per event, total production, when the first unit appeared, and the average time between units. Use it to answer, in plain language: *Did we produce what we thought, at what overall pace?*

5.2 Productivity by cycle and steady state

The units-per-hour chart starts at cycle **0**. Early cycles are often noisy—the system is “filling.” **Steady state** in Neo-CYCLONE is a practical teaching rule, not a theorem: productivity stays within about **5%** across a window of at least **10** cycles. The guide appears as an **old-gold dashed** line with a readable value so a class can say, “We would quote about *this* productivity,” not the wild first spike.

5.3 Idleness and busy time

For each resource, idle % and busy % are both labeled so a tiny idle bar still has a story (busy may sit near 100%). High idle on a costly resource is a design smell. High idle on a cheap buffer may be intentional. If you only remember one chart from the studio, remember this pair.

5.4 Cost Report

When you supply hourly rates:

- cost per resource ≈ count × (USD/h) × run hours
- **unit cost** ≈ total cost ÷ production

Unit cost is often the decision metric students remember—especially next to sensitivity charts. No `Cost:` block means no cost report; that is expected, not a bug.

5.5 Sensitivity Analysis

When the prompt defines `Sensitivity:`, the Sensitivity tab compares combinations (for example trucks versus loaders): productivity and unit cost side by side, best markers, and idleness views. Pairwise comparison supports more than two resources within teaching caps. Detail tables can be hidden so the story stays visual.

Sensitivity batches prefer a **Web Worker** so the interface stays responsive; if Workers fail, the same engine runs on the main thread (same numbers, possible brief pause). Single **Simulate** stays on the main thread—it is fast enough for classroom sizes.

5.6 Export and a one-minute discussion

Export Excel and PNG when you need figures for slides or homework. Always note Operation name, seed, max cycles, and any sensitivity ranges.

In one minute of class discussion:

1. What is steady-state **units/hour**?

2. Which resource has the highest **idle %**?
3. What is **unit cost** (if costs were entered)?
4. If we add one unit of the scarce resource, what do we *expect*—then test with Sensitivity or a re-run.

Chapter 6 — Six Examples as a learning path

Selecting an Example only fills the prompt. **You** click Draw Model. Keep Chapter 4 open in your mind while you look at each diagram.

#	NAME	WHAT YOU SHOULD NOTICE
1	**Earthmoving**	Classic two-resource cycle; cost; steady state. No branch, no SA—learn the spine first.
2	**Asphalt Paving**	Meeting at dump-to-paver; **branch** breakdown then refill; count after pave.
3	**Loading Dump Truck**	Inline **GEN/CON**: excavator scoops fill a truck before haul-return.
4	**Tower Crane**	Multi-demand `
5	**Masonry**	Face stocks; helper multi-demand; **sensitivity** introduction.
6	**Precast Plant**	Longer line production; richer SA—systems thinking.

Suggested path: 1 → 2 → 3 for mechanics; 4 for shared resources; 5–6 for decisions under sensitivity. Rewrite any example. Change counts. Break a duration. Re-draw. That is the point.

Chapter 7 — What “AI-assisted” should mean here

The Assistant sits under Results. It should feel like a teaching assistant who has read your board—not like a search engine that invents another project.

7.1 Purpose and honest technology

The Assistant should:

- Explain *this* model’s cycles, COMBI versus NORMAL, counter, GEN/CON, and branch.
- Point at bottleneck and idleness from the *last run*.
- Propose a full Format Prompt edit when you ask to change fleet or durations.
- Stay short (about ≤20 lines) so the studio remains the focus.

It must **not** silently re-simulate, invent another operation, or replace CYCLONE with a mystery model.

MODE	WHEN	BEHAVIOR
AI mode	Host has `XAI_API_KEY` (for example on Vercel)	Chat model sees a **compact CONTEXT** snapshot only
Local mode	No key or API failure	English-first intent helper for common studio questions

Product UI, Manual, and keywords are English-first for an international classroom. Rate limits on the shared host (about **30** Assistant requests per hour per IP) protect classroom use from abuse; normal class pace is fine.

7.2 How the chat behaves

- **You** — gold bubble, right-aligned, compact (WhatsApp-style).
- **Assistant** — light bubble, left-aligned.
- System guidance lives in the **placeholder** of the input box, not as a permanent chat bubble.
- Quick chips are **general** (no truck-only assumptions): resources, bottleneck, productivity, unit cost.

7.3 Boundary, workflow, and better questions

> The AI Assistant answers only about the **current** Format Prompt, drawn CYCLONE network, and **last** simulation or sensitivity results. It may **propose** Format Prompt edits; you must **Apply**, **Draw Model**, and **Simulate** for changes to take effect.

Workflow: build and simulate a model you understand → ask focused questions → if a new prompt is proposed, Apply → Draw → Simulate → compare numbers. Do not accept a quantitative claim without a run.

WEAKER	STRONGER
“Make it better.”	“Which resource has the highest idle % after this run?”
“Optimize everything.”	“Propose trucks = 8; keep loader = 1; I will re-simulate.”
“What is CYCLONE?” with an empty model	Draw Example 1 first, then “Explain this model’s resource cycles.”

Chapter 8 — Limits, deploy, and integrity

Teaching studios need guardrails so a shared host stays fair and numbers stay interpretable.

8.1 Simulation and sensitivity caps

PARAMETER	DEFAULT	HARD LIMIT	NOTES
Max cycles	100	**500**	Teaching cap; the UI clamps higher values
Seed	12345	—	Reproducibility; dice for alternate paths
Time unit	minutes	—	Stated in the Format Prompt header
SA resources	—	**5**	Extra ranges ignored with a note
SA combinations	—	**~150**	Step increases; not a silent mid-axis cut

8.2 Performance, AI, and deploy

Sensitivity prefers a **Web Worker**; fallback is the main thread with the same engine. Single Simulate runs on the main thread—appropriate for teaching sizes.

AI limits: about 30 Assistant requests / hour / IP; about 20 AI DSL draft requests / hour / IP; compact CONTEXT only; replies short. Without an API key, the local English helper still works. These limits do **not** change CYCLONE rules; they protect hosting and API cost.

Source of truth: **GitHub** `main` auto-deploys to **Vercel** at neo-cyclone.vercel.app. Teaching does not require sign-in. Always cite the footer **version** after a deploy.

8.3 Citing Neo-CYCLONE in homework or papers

Include:

1. Product name and version (footer).
2. URL of the live app.
3. Operation name, seed, and max cycles.
4. Whether results came from Simulation only or also Sensitivity.
5. Optional: software DOI if your course requires a formal software citation.

Chapter 9 — FAQ from real studio use

Drawing and the model diagram

Q: I selected an Example but nothing drew.

A: Click **Draw Model**. Examples only paste the prompt.

Q: Why does the diagram not match the book figure exactly?

A: Same CYCLONE *logic*; Neo-CYCLONE teaching *notation* (Chapter 4). Black and gold arrows and some shapes are intentional—not a bug.

Q: What do gold dashed arrows mean?

A: Resource **return** to a **home QUEUE**. Solid black means forward work (including into staging before a COMBI).

Q: COMBI versus NORMAL looks wrong.

A: COMBI only when **two or more resources** must meet to start the task. Solo work → NORMAL.

Q: Where is the production counter?

A: Golf-flag node. Set with `Counter after: TaskName`. Multiple flags are allowed.

Q: GEN and CON look like random triangles.

A: ▽ GEN multiplies units on arrival; △ CON gathers $n \rightarrow 1$. Prefer inline form on the cycle. Not required in every model.

Simulation and results

Q: Why is Simulate not useful yet?

A: Draw a model you trust first. Numbers without a truthful diagram are noise.

Q: My classmate got different productivity.

A: Compare **seed**, **max cycles**, and whether the Format Prompt is identical—including durations and branches.

Q: What is the steady-state line?

A: Teaching guide: productivity within about **5%** over at least **10** cycles—old-gold dashed on the units/hour chart. Quote that band, not the first spike.

Q: Sensitivity tab is empty.

A: Add a `Sensitivity:` block (Examples 5–6). Without ranges there is nothing to sweep.

Q: Where is cost?

A: Only if the prompt has `Cost:` rates (USD per resource-hour). Then the Cost Report shows totals and **unit cost**.

Prompt, language, AI, and access

Q: Excel file name looks ugly.

A: Add `Operation: ShortName` after `#` notes, before resource cycles.

Q: Can I write the prompt in Indonesian?

A: `#` notes may be any language. Keep **network keywords**, task names used in counters, and distribution keywords in **English** for reliable parsing.

Q: The AI proposed a prompt but numbers did not change.

A: Click **Apply**, then **Draw Model**, then **Simulate**. Nothing silent updates the engine.

Q: Is sign-in required?

A: No for teaching use.

Q: Where is the bibliography?

A: **References** at the **end** of this manual, after the Epilogue.

Chapter 10 — Notes for instructors and peer mentors

With many concurrent learners, small conventions prevent chaos.

10.1 Baseline and exercise patterns

Agree once for the whole class:

- Example **1. Earthmoving** (or a course-specific prompt),
- Seed **12345**,
- Max cycles **100** (or 200 if you prefer longer settling).

Everyone's first Process Report should match. Then change **one** thing per exercise.

PATTERN	ASK STUDENTS	WHAT THEY SUBMIT
See waste	Run baseline; screenshot idleness	Which resource waits? Why?
Read the diagram	Label QUEUE / COMBI / NORMAL / COUNTER / gold return	Screenshot + five labels
Fleet change	Apply trucks 3 vs 8; same seed	Unit cost + productivity table
Branch	Example 2	Effect of the breakdown path
Shared resource	Example 4; swap priorities	Who starves?
Sensitivity	Examples 5–6	Best unit-cost combo + a caution

10.2 AI policy and a light rubric

Allow the Assistant for **explanation** and **prompt drafts**. Require **Apply** → **Draw** → **Simulate** before any graded claim. Remind labs that a shared NAT may share one IP against rate limits.

CRITERION	STRONG LOOKS LIKE
Model truth	Diagram matches the narrative (Chapter 4 checklist)
Waste literacy	Idle/busy and bottleneck—not only total production
Reproducibility	Seed, cycles, and version cited
Decision	Unit cost or SA-informed recommendation
Integrity	Engine results primary; AI optional

No install is required: a modern browser and the Manual in the header are enough. After a deploy, hard refresh if styles look missing.

Epilogue

When the diagram is clean and the numbers are stable, you have done what Halpin asked of a generation of students: **see the operation**. AI can speed the typing; it cannot replace that seeing.

Keep one run with seed **12345** as a shared baseline. Change one idea at a time—fleet, duration, branch, or priority—and ask what happened to **idleness** and **unit cost**. That discipline matters more than any single feature.

If the on-screen model looks slightly different from a photocopy of a 1992 figure, return to Chapter 4: honor the **logic**, learn this studio's **legend**, then run the engine.

Thank you for using Neo-CYCLONE with care.

References

Selected literature on **CYCLONE**, **MicroCYCLONE**, and applications by **Daniel W. Halpin**, students, and collaborators. Placed **last** so teaching chapters stay in front.

Foundations

1. **Halpin, D. W.** (1973). Ph.D. dissertation, University of Illinois at Urbana–Champaign.
2. **Halpin, D. W.** (1977). "CYCLONE: Method for Modeling of Job Site Processes." *Journal of the Construction Division*, ASCE, 103(3), 489–499.
3. **Halpin, D. W., & Riggs, L. S.** (1992). *Planning and Analysis of Construction Operations*. Wiley.

MicroCYCLONE

1. **Lluch, J., & Halpin, D. W.** (1982). *Journal of the Construction Division*, ASCE, 108(1), 129–145.
2. **Halpin, D. W.** (1990–1992). MicroCYCLONE user and system manuals (Purdue / Learning Systems).

DISCO

1. **Huang, R.-Y., & Halpin, D. W.** (1993–1995). DISCO-related papers (ISARC; *Microcomputers in Civil Engineering*; *Journal of Construction Engineering and Management*).
2. **Huang, R.-Y.** (1994). Ph.D., Purdue University (advisor: Halpin).

PROSIDYC · COST · WebCYCLONE

1. **Halpin, D. W., & Martinez, L.-H.** (1999). PROSIDYC. *Winter Simulation Conference*.
2. **Cheng, T.-M., et al.** (2000). COST. *17th ISARC*.
3. **Halpin, D. W., Jen, H., & Kim, J.** (2003). WebCYCLONE. *Winter Simulation Conference*.

Related lineage

1. Work in the Halpin circle and peers: AbouRizk & Halpin; Hijazi; Lutz; Gonzalez-Quevedo; Abraham; project-level CYCLONE studies; AbouRizk et al. (2011) and related synthesis.

2. Related systems students may meet later: **UM-CYCLONE** (Ioannou), **STROBOSCOPE** (Martinez), **Simphony / Simphony.NET** (AbouRizk et al.).

How Neo-CYCLONE relates

Neo-CYCLONE does **not** claim to supersede research simulators. It is **AI-Assisted Construction Operation Simulation**—a studio for first principles: flow, idleness, cyclic networks, transparent discrete-event logic, and responsible AI beside that engine. Diagram notation follows this product's standard (Chapter 4; `NOTATION_STANDARD.md`) while remaining in Halpin's tradition.

AI-Assisted Construction Operation Simulation · Manual v1.7.2