2026 NASA COST AND SCHEDULE SYMPOSIUM

# POLARIS$^2$

PROGRAM OVERSIGHT & LOGIC ANALYSIS FOR RISK & INTEGRITY OF SCHEDULES

From forensic schedule analysis to an integrated program controls intelligence platform



**David J. Politte**

Scheduling SME — NASA Glenn Research Center

▶ FANFARE

CUI/ITAR-SAFE AI · EVERY NUMBER CITED

01 · ORIGIN

# Where this started

Preparing for a flagship-program Integrated Baseline Review meant the same manual loop, run by many people, every cycle — across schedules stretching back **more than a year** , never once evaluated by a seasoned forensic scheduler.

THE GRIND

TOOL 1
**Acumen Fuse**

TOOL 2
**SSI**

TOOL 3
**Microsoft Project**

TOOL 4
**Excel**

Export → Interpret each output → Verify by hand → Chase explainable outliers ↺

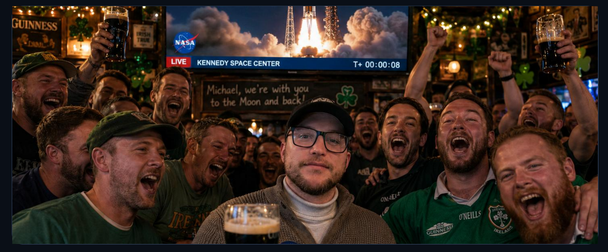
THE SPARK

Overloaded, I handed our newest scheduler a stack of Acumen trend charts — hours of Excel, I figured. Done in `under 20 minutes` . “How in the…?” The student became the teacher: **Michael Moran introduced me to AI.**

**Jason Nehez** taught me prompting and GitHub — and has endured hearing about my addiction ever since, which deserves an award all by itself.




MICHAEL MORAN




JASON NEHEZ

**Repetitive, time-consuming, counterproductive — it could be modeled, and automated.**

01 · ORIGIN

# What POLARIS² is

A data-sovereign, AI-augmented desktop application that turns native schedule files into traceable, decision-ready findings — **every analysis done 100% locally, or through CUI/ITAR NASA-approved AI models.**

LAW 1 · DATA SOVEREIGNTY

### Nothing about a schedule leaves the machine — unless you arm the approved gateway.

Built for CUI. A network-egress guard fails the build if an unapproved cloud client enters the runtime; the AI defaults local and fails closed when in doubt.

LAW 2 · FIDELITY OVER SPEED

### Numbers must match the reference tools.

Acumen Fuse, SSI, and Microsoft Project on the same inputs — parity is gate-locked in CI. A fast, wrong number is worthless in a forensic context.

Deterministic engine calculates · AI explains · Every claim cites file + UID + task

# Teaching AI to read schedules

THE WALL

**AI could not read the files the work lives in.**

`.mpp` is an undocumented binary container. `.xer` is dense proprietary text. Early sessions guessed — and guessing is disqualifying in forensics.

THE CLIMB

01 Learned the formats — field by field, calendar by calendar.

02 Native `.mpp` via an embedded reader + portable Java — **no Microsoft Project install required.**

03 `.xer` and MS Project XML parsed in pure Python — calendars, logic, constraints, baselines, resources, stable UIDs all retained.

Then the hard-won knowledge was committed as **project skills** — procedures the AI loads at the start of every session, so nothing is ever relearned.

`metric-parity` `render-verify` `cui-guard` `full-gate` `prove-able-to-fail` `ui-change` `session-close`

02 · BUILD

# The power is in the parent file

The goal was never to recreate SSI, Acumen Fuse, or Microsoft Project — one place and no license fee is convenient. **The goal is the data exports never give you.**

### THE EXPORT PATH

Exporting or converting drops data — fields truncated or corrupted, a hard cap on how many survive even when carefully mapped, calendars and logic detail left behind.

### THE NATIVE PATH

Parsing the native `.mpp` or `.xer` reads **everything that is there** — every field, calendar, constraint, baseline, resource, and UID — and the AI helps interpret it quickly and accurately.

### BURIED CONSTRAINTS SURFACE

The hard constraint quietly pinning Starlight's launch milestone rarely survives an export mapping — read natively, the pinned date masking negative float is caught.

### REWRITTEN HISTORY SHOWS

Baseline dates and recorded actuals live in the file with their edits — cross-version diffs catch actuals changed after the fact, which summary exports smooth over.

### CALENDAR TRUTH HOLDS

The imported work week, day lengths, and 100+ holiday exceptions drive every computed date — no silent default-calendar math shifting finishes.

**GET IT BACK OUT** Every underlying table exports to **Excel (.xlsx)** or **PDF** — any native MS Project or Primavera field, or any custom field, can be added and exported if it exists in the native file: `WBS · work order number · OBS · more`.

# Building with an AI pair

AI build sessions, large and small — **all state committed to git, never only in chat.**

## ~110

SESSIONS

One sitting of AI-assisted work with a defined goal — opened from the last handoff, closed by writing the next.

## 17

MILESTONES M1–M17

Planned capability blocks — parsers, engine, UI, AI — each done only when its tests and gates pass.

## 431

DECISION RECORDS

One committed document per architecture choice — what was decided, why, and what it replaced.

## 4,000+

TESTS · ~98% COVERAGE

Automated pass/fail checks run on every change — covering ~98% of the code, so a break surfaces immediately.

EVERY CHANGE PASSES

- lint + strict types
- security scans
- parity gate
- egress guard
- coverage gates

# Matching the AI to the job

CLAUDE CODE

### The builder

Ran the milestone sessions end to end — engine, importers, UI — committing state to git.

CLAUDE

### Narrative + review

Drafted documentation and the Mission Ops story; audited the builder's work.

CLAUDE DESIGN

### The design studio

Shaped the visual language and screens — and built this very deck.

CHATGPT-CODEX

### Second opinions

Targeted coding tasks and independent cross-checks of the others' output.

ALSO IN THE ROTATION Gemini · Grok · even Copilot — for the light work 🙄

MODELS AUDIT MODELS

Claude audited Codex output, and vice versa — **a different model catches different mistakes.**

EFFORT DIALED PER TASK

Frontier reasoning for architecture and audits; lighter, faster versions for routine edits.

SKILL · SESSION-CLOSE (HANDOFFS)

Every session ends by writing the handoff — decisions recorded, state rotated — so the next session, or the next model, starts exactly where the last one stopped.

SKILL · FULL-GATE (QUALITY CONTROL)

The complete pre-commit gate as an invoked procedure — lint, strict types, security, parity — plus a QC agent that re-runs it and fixes what it finds.

CHALLENGE EVERYTHING

"Assume everything you just said is wrong — **prove it.**" The AI authors pass/fail tests, runs them in a sandbox, and **only proven changes get pushed.**

# The build timeline

Oct–Nov 2024

**IBR prep**

the manual grind plants the idea

Nov 2024

**The experiment begins**

can AI read — and judge — a schedule?

Jan–Feb 2025

**Parsers + engine**

native .mpp, .xer, XML aboard; CPM + DCMA-14 solving locally

Jul 2026

**POLARIS² brand + Mission Ops UI**

12-chapter story, four themes

Aug 2026

**Launch sequence + hardening**

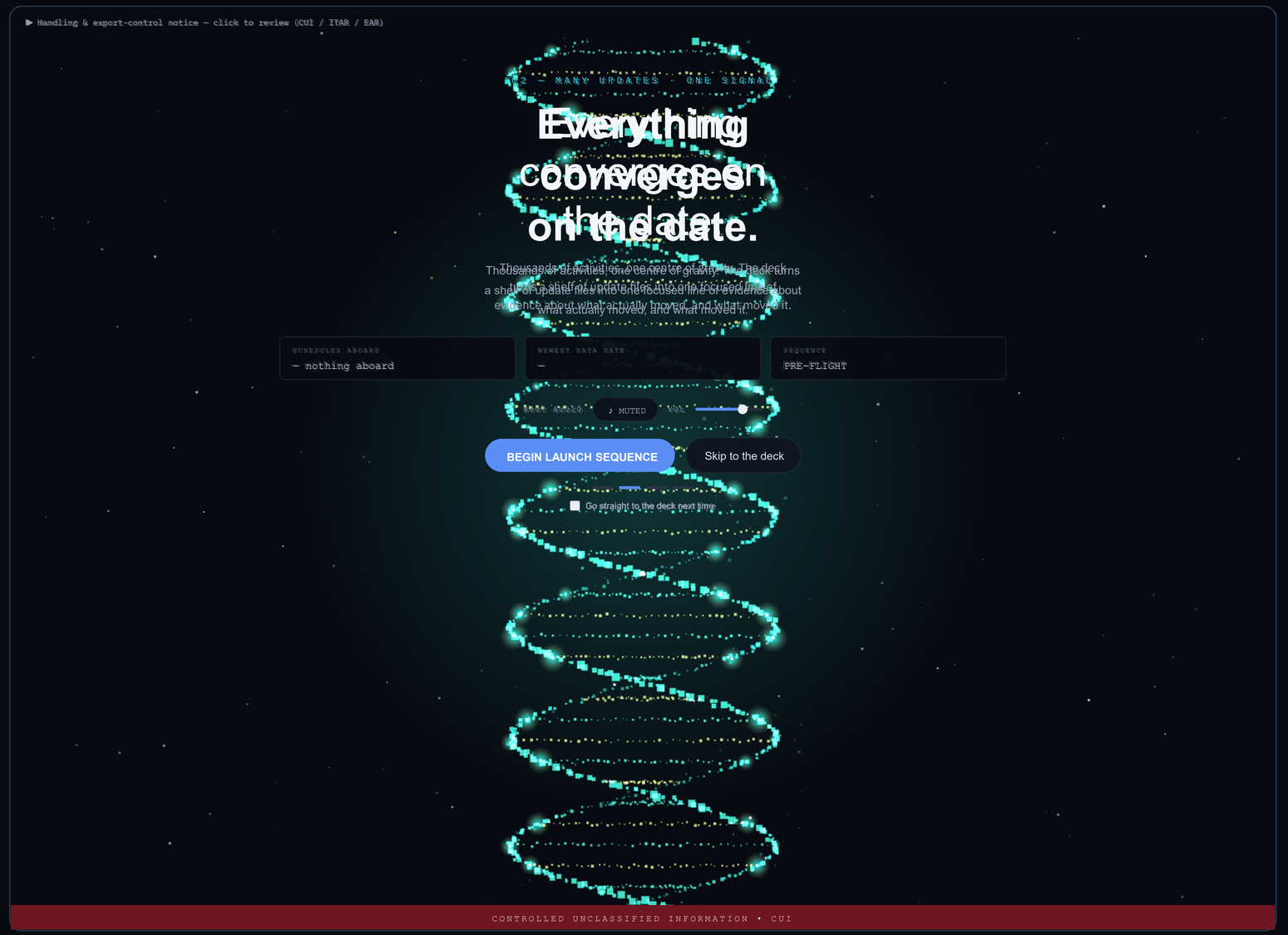
boot lightshow; suite passes 4,000 tests

Aug 14, 2026

**NASA-approved AI gateway live**

ITAR-authorized models, fully logged

# The launch sequence




### PARTICLE LIGHTSHOW

Up to 15,000 particles, four scenes that morph into each other — one substance rearranging.

### SYNTHESIZED BOOT AUDIO

A generative WebAudio hum — no media file ships, no loop ever repeats.

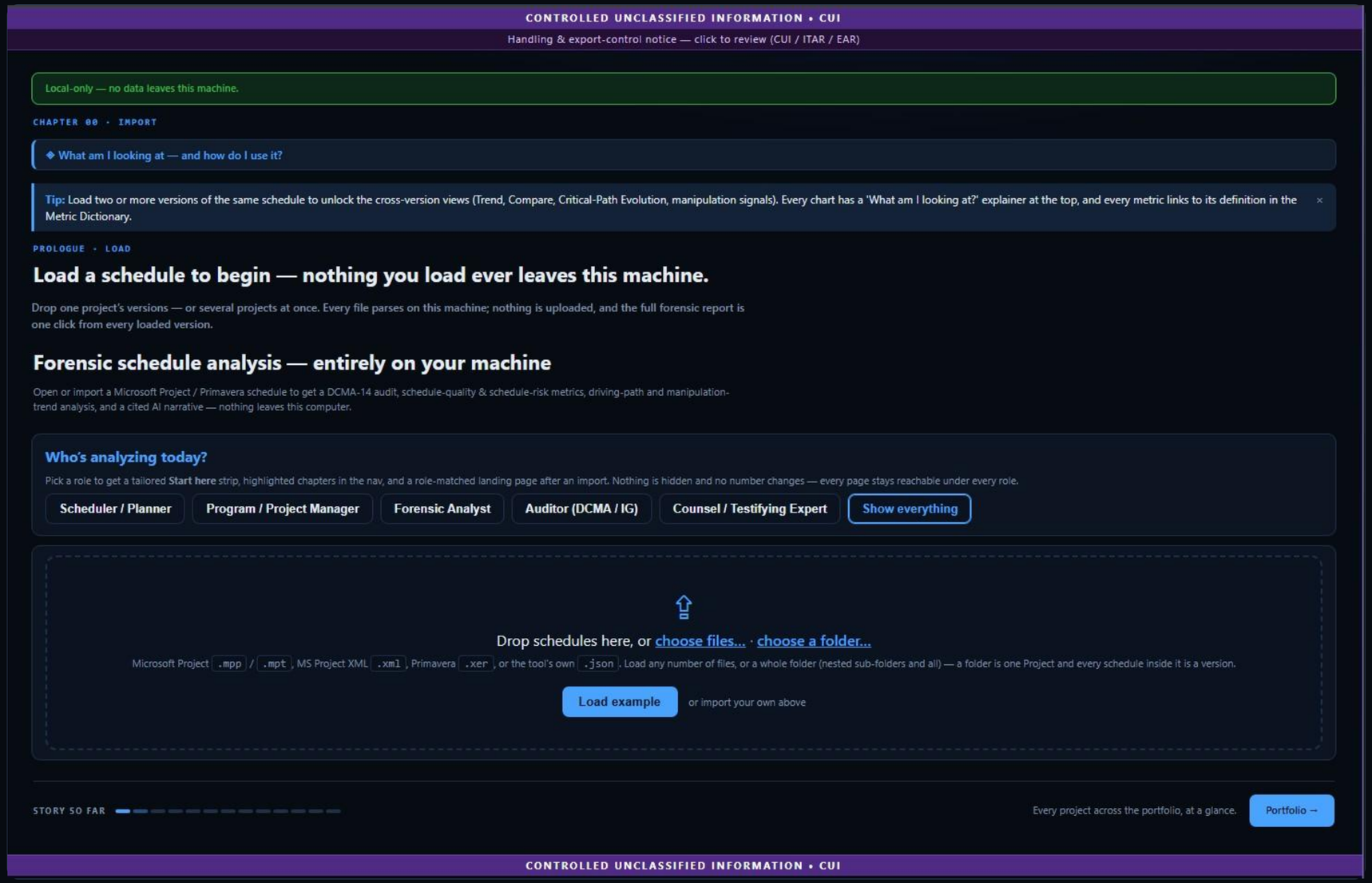
### HONEST TELEMETRY

Tiles read real session facts — an em dash when empty, never a fabricated figure.

### CUI CHROME, ALWAYS

Handling and export-control marking on every screen, including this one.

# Import — native files, no conversions




### `.mpp · .mpt · .xml · .xer · .json`

Native `.mpp` parses with no Microsoft Project installed.

### MANY FILES AT ONCE

Select a pile of schedule files in one drop — the program separates them into projects using the **Project Information the schedule author entered**.

### MANY FOLDERS AT ONCE

Load multiple folders together — every schedule inside a folder becomes **one project** , each file a version.

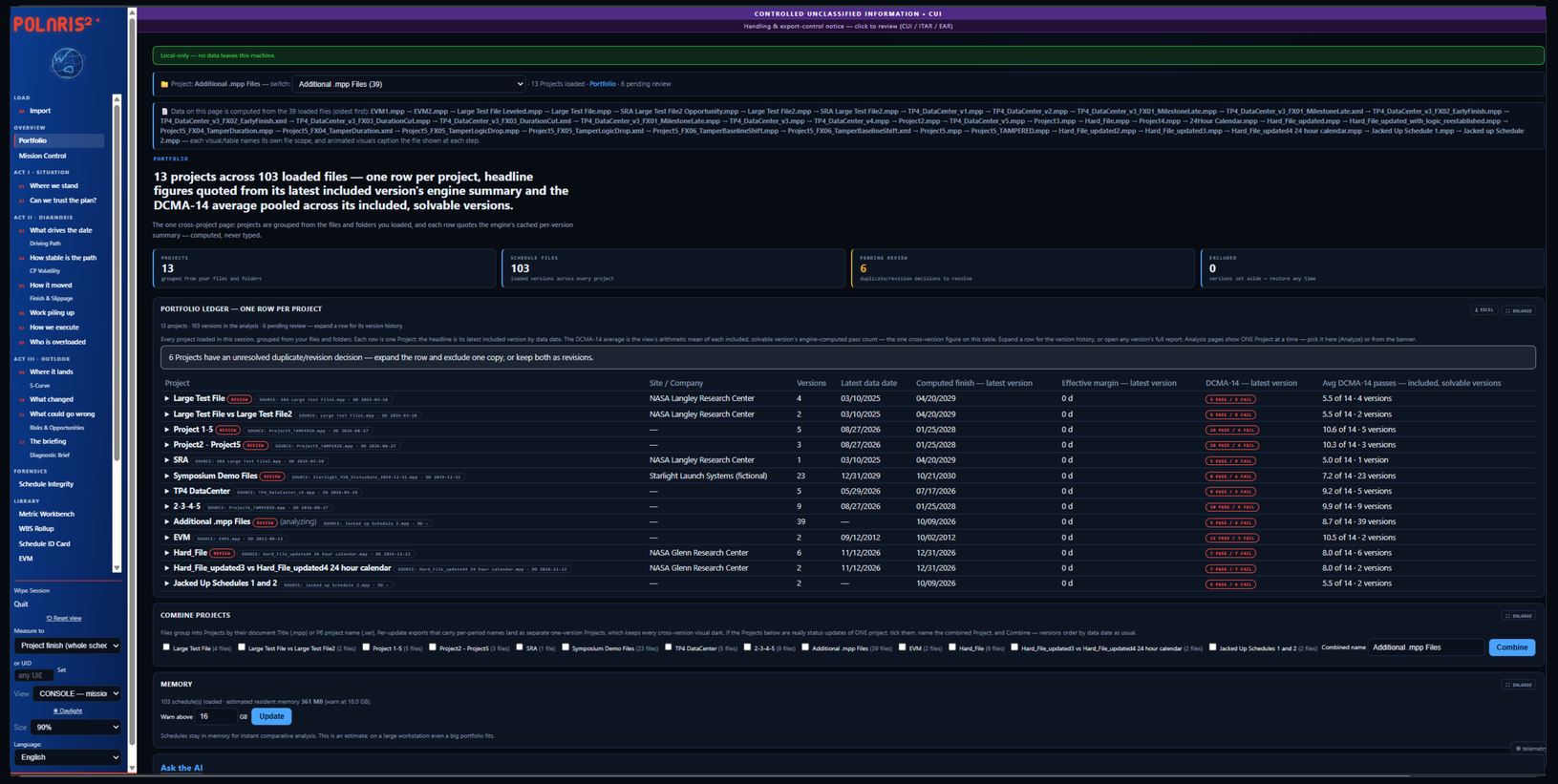
### ROLE-AWARE LANDING

Scheduler, PM, forensic analyst, auditor, counsel — emphasis changes, data never does.

### NO SILENT FAILURES

Every file parses, or the page names the file and the exact reason it did not.

# Portfolio — one row per project




## ALL ABOARD AT ONCE

13 projects · 103 schedule files — every project holding its own versions, uploaded in one pass.

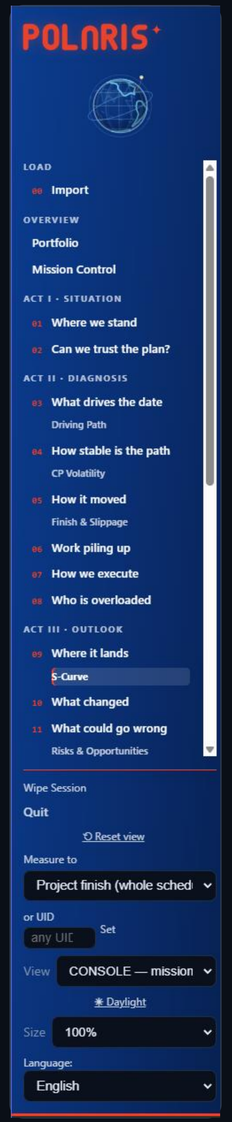
## POOLED QUALITY SCORE

DCMA-14 averaged across included versions: 7.2 of 14, latest 8 pass / 6 fail .

## BUILT TO HOLD MANY

Files group into projects automatically, one row each — cross-project comparison is in development.

# The 12-chapter story




## PROLOGUE · LOAD

00 Import — files, folders, roles

## OVERVIEW

Portfolio · Mission Control

## ACT I · SITUATION

01 Where we stand 02 Can we trust the plan?

## ACT II · DIAGNOSIS

03 What drives the date · 04 How stable is the path · 05 How it moved · 06 Work piling up · 07 How we execute · 08 Who is overloaded

## ACT III · OUTLOOK

09 Where it lands · 10 What changed · 11 What could go wrong · 12 The briefing

## INSIDE THE CHAPTERS

DCMA-14 audit · driving-path Gantt · trend · S-curve · Bow Wave / CEI · forecasts · cited AI

SIMULATED DATA · DRAWN LIVE

# The bow wave — twelve schedules, one growing pile

CONFIRMED · BOW WAVE

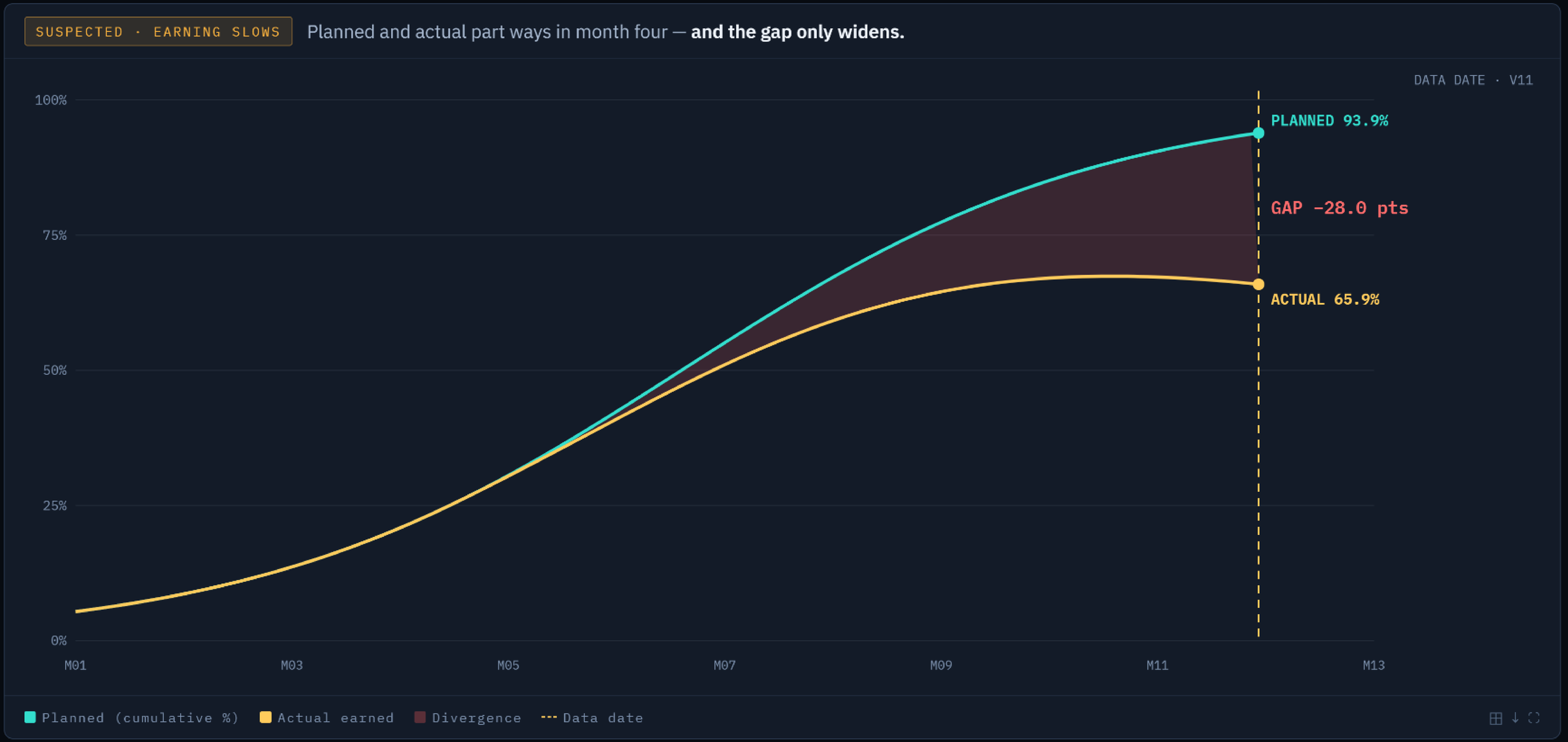
Unfinished work is pushed just past the data date every month — **the wave never breaks, it grows.**



# The diverging S-curve

SUSPECTED · EARNING SLOWS

Planned and actual part ways in month four — **and the gap only widens.**



# Starlight V05–V10 — the story on paper

## 0.990

SPI — READS ON-TRACK

Schedule Performance Index: earned progress ÷ planned progress. 1.0 means exactly on plan — 0.990 reads a hair behind, but healthy.

## −356 days

COMPUTED FINISH, PULLED IN ACROSS THE SERIES

How far the finish date the network computes moved across the six versions — nearly a year earlier, on paper.

## +1.9 pts

FINAL GAP — FIRST VERSION EVER AHEAD OF PLAN

Actual % complete minus planned % complete at V10's own data date — the first time the project ever reads ahead of plan.

COMPLETION VS PLAN, AT EACH VERSION'S OWN DATA DATE




Flat for four updates — then the entire improvement lands in `one final step`.

THE POINTS SCALE

1 point = 1 percentage point of total project completion. Each gap is `actual % complete − planned % complete`, measured at that version's own data date. `V10: 97.7 − 95.8 = +1.9 ahead`

04 · CASE STUDY

# What manipulation looks like

The classic levers — each one buys schedule **on paper** without buying any work.

### 01 SHORTENED DURATIONS

Remaining work keeps getting faster with no added scope, staff, or shifts — recovery that exists only in the file.

### 02 LOGIC EDITS

Links deleted, resequenced, or padded with leads — the critical path is steered around the trouble instead of through it.

### 03 HARD CONSTRAINTS

Pinned dates override the network math — the negative float the logic would expose never gets to show.

### 04 RE-BASELINING

The yardstick moves — variance resets to zero and the slip disappears from every report that follows.

### 05 REWRITTEN ACTUALS

Recorded history changes after the fact — the one thing that should never move, moved.

### 06 QUIET SCOPE DELETION

Activities vanish — test and verification first — so the finish stops slipping because the work stopped existing.

**Any one edit can be legitimate. The tell is the pattern — the same levers, every update, always in the schedule's favor.**

# The manipulation fingerprint

How to read it: `5 of 5` means the lever was pulled in **all five version-to-version steps**, V05 → V10 — and never once in the other direction.

### 5 of 5

**Remaining durations cut in every update** — recovery claimed without added people, shifts, or scope.

### 5 of 5

**Planned work hours edited in every update** — the effort was shrunk so the dates would fit.

### 5 of 5

**Cost changed in every update** — the budget re-shaped to keep the shorter schedule believable.

### 5 of 5

**Resource assignments moved in every update** — loading rearranged to justify the compression.

### −383 days

**Nearly all finish improvement lands in the two most heavily edited updates.** Real recovery tracks steady progress — improvement that clusters at the edit points came from the file, not the floor.

### 24 + 9

**24 baseline dates re-set in the “good news” update; 9 activities’ recorded actuals rewritten.** Move the yardstick and variance resets; rewrite history and the trend means nothing.

### Deleted: V&V

**The removed scope is verification & validation.** The finish stops slipping because the proving work stopped existing — the risk stayed with the project.

**“A schedule that only ever gets shorter is being steered.”** — POLARIS² AI narrative, demo session

# The AI's verdict — in its own words

## THE QUESTION, AS TYPED

When looking at all schedules is there a trend of what appears to be intentional schedule manipulation and if so explain why?  Put your response in the form of an email directed to upper level program management.

## THE ANSWER — AN EMAIL TO LEADERSHIP

Leadership,

**Bottom line up front:** Yes. Looking across all six loaded versions as one series, there is a consistent, repeating pattern of edits that a forensic reviewer would flag as *possible* intentional schedule manipulation. I want to be precise about my language: the tool detects and counts *what changed* between versions; it does not — and I cannot yet — prove *why* it changed. But the shape of the changes is not what an honestly maturing schedule normally looks like, and it warrants a documented explanation from the planning team before we rely on the current finish date.

Here is why I reach that conclusion.

### SHOWN VS INFERRED

Facts computed from the files are labeled as shown; judgment is labeled as inferred. Intent is never asserted.

### THREE FORECASTS DISAGREE

**As-scheduled** — 2030-06-20

The finish stored in the file — the dates exactly as the scheduler entered them.

**CPM logic** — 2030-09-19

The finish the network itself computes from durations and logic links — what the plan supports if the rest goes per plan.

**Earned schedule** — 2030-12-04

The finish projected from the pace actually demonstrated so far — keep earning schedule this slowly, and this is where you land.

They should roughly agree — a stored date **earlier than the logic** means constraints are pinning it.

### FULLY TRACEABLE

Every number in the email traces to a specific version and activity ID — and it ends with recommended actions.

# Ask the AI — reading the S-curve

## THE EMAIL, AS GENERATED — TREND TABLE INCLUDED

**Subject: Starlight V05–V10 — What the S-Curve Trend Is (and Isn't) Telling Us**

Dear Program Leadership,

You asked for a plain-language read on what the S-curve trend across our six loaded schedule versions says about how the program is progressing. Here is the high-level picture, followed by the caveat that materially changes how you should interpret it.

**The headline trend (what the data literally shows)**

Reading each version's cumulative completion against its own plan, at its own data date:

| Version | Data date | Actual | Planned | Gap |
|---|---|---|---|---|
| V05 | 2028-06-30 | 85.2% | 85.7% | −0.5 |
| V06 | 2028-12-29 | 89.5% | 90.0% | −0.5 |
| V07 | 2029-03-30 | 90.9% | 91.0% | −0.1 |
| V08 | 2029-06-29 | 92.4% | 92.5% | −0.1 |
| V09 | 2029-09-28 | 93.9% | 94.0% | −0.1 |
| V10 | 2029-12-31 | 97.7% | 95.8% | **+1.9** |

Taken at face value, the gap improved from −0.5 points to +1.9 points — a 2.4-point swing toward, and then past, the plan. No step ever got worse. The current SPI of 0.990 and a forecast finish 38 workdays ahead of baseline are consistent with that "we're on plan, slightly ahead" story.

**Why I am not comfortable reporting that as good news**

## WHY IT WON'T CALL THAT GOOD NEWS

The trend is arithmetically real but analytically fragile, for three connected reasons:

1. **Each version is scored against its own baseline — and the baseline was moved.** For the first five versions the gap sat essentially flat at roughly −0.1 to −0.5 points: we were tracking plan, never pulling ahead. Then in the final update (V09→V10) the gap suddenly jumps to +1.9. That same update contains a signal that **24 activities had their baseline dates changed**. When you re-baseline and then measure "actual vs. plan," the gap can improve without a single extra day of work being done. The near-vertical +3.8-point jump in *actual* completion in one quarter (93.9% → 97.7%), against a modest planned move (94.0% → 95.8%), is the fingerprint of a plan being re-drawn, not of a work surge.

2. **The finish date pulled in by nearly a year in the same step — driven by edits, not progress.** The network's computed finish moved 356 days earlier across the series, with −238 days of that concentrated in the final V09→V10 update. Our counterfactual test is decisive here: reverting the specific V09→V10 edits (duration cuts restored, removed logic links restored on the launch-campaign chain — UIDs 659–664) shows the schedule time was **removed by the edits, not earned by execution**. In that same step, **4 activities were deleted** from the critical path region (Independent V&V Audit, Umbilical Mate & Verification, Post-Flight Anomaly Assessment, Pad Safing & Refurbishment).

3. **This is a pattern, not a one-off.** Across all five version-to-version steps, four signals fire in **every single step**: cost changed, resource assignments edited, durations shortened, and work hours changed (5 of 5 each). Earlier we also saw actual dates and actual work being reduced (V07→V08, 9 activities each), a clamping constraint added to the launch milestone, and activity deletions in the very first step. A gap that "improves" while durations are repeatedly shortened and scope is repeatedly deleted is not the same thing as a project getting healthier.

### TREND, TRANSLATED

States what the data literally shows — gap −0.5 → +1.9, no step ever worse — before judging it.

### THE CAVEAT THAT MATTERS

The whole improvement lands in the re-baselined final cycle — "measured against a plan that was just moved," not proof of recovery.

### THE VALUE

An hour of chart-reading compressed into a leadership-ready email — ending with three concrete asks, every figure traceable.

# Ask the AI — judging schedule integrity

## THE PATTERN, COUNTED

**1. The same "date-improving" edits recur in every single update — that is the definition of a pattern.**

Across the 5 consecutive comparisons, 31 signals fired, and signals appeared in **5 of 5** updates. Four edit types recurred in an unbroken run across **all five** periods:

- Duration shortened — 5 of 5
- Work hours changed — 5 of 5
- Cost changed — 5 of 5
- Resource assignment edited — 5 of 5

A one-off edit is housekeeping. The same four levers being pulled every reporting period, without exception, is the repetition a reviewer calls a pattern. Of particular note: **durations were shortened in every update and never routinely lengthened.** Real schedules encounter both good and bad news; a schedule that only ever gets shorter is being *steered*.

## THE RECOMMENDED ACTIONS

**Recommended actions (before the next customer submission)**

1. **Freeze the current finish narrative.** Do not report V10's "38 days ahead" or "+1.9 points ahead of plan" externally until items 2–4 are cleared.
2. **Demand the audit trail for the Step 3 actuals rewrite** (UIDs 5–8 + 5 others): who changed reported actual dates and erased actual work, on what authority, with what basis.
3. **Produce the re-baseline authorization for Step 5** (the 24 baseline-date changes) and confirm it followed a controlled re-baseline, not an in-cycle edit.
4. **Justify every deletion** (9 activities across Steps 1 and 5) as authorized descope, with special scrutiny on the test/V&V items — confirm the work is genuinely gone from program scope, not just from the

### A PATTERN, NOT A HUNCH

31 signals across five comparisons; the same four levers pulled in every update — repetition a reviewer can defend.

### LANGUAGE BUILT FOR THE RECORD

"Detects and counts what changed — does not prove why." Intent stays with humans; innocent explanations are acknowledged.

### THE VALUE

Findings arrive as four defensible asks — audit trail, authorization, justification — not accusations. Ready for a program review.

# Hardening the results

PARITY WITH THE REFERENCES

## 108 / 108

SSI driving-path checks match exactly; Acumen Fuse v8.11.0 metrics reproduced, with every residual named and pinned. The parity suite is a locked CI gate.

SEEDED-DEFECT TESTING

### Plant known issues. Demand the tool finds them.

We authored schedule files with deliberate, documented problems. Every miss meant train, correct, and re-run until the catch was reliable.

PROVE IT CAN FAIL

### Every new guard is broken on purpose — once.

A check that has never failed proves nothing. New tests are run against a deliberately broken build before they count.

- Honest progress raises no false flags — verified on clean update histories.

# Hundreds of test files, both directions

Validating not just the numbers POLARIS² produces — **how it derives them.**

Author files with known truth → Run SSI + Acumen Fuse → Run POLARIS² → Compare, line by line

### BUILT TO PASS

Clean, honest schedules must sail through with **zero false flags** — an alarm that cries wolf is as useless as one that sleeps.

### BUILT TO FAIL

Files with planted defects must be caught — **every miss meant train, correct, re-run** until the catch was reliable.

A SAMPLE OF WHAT THE BATTERY COVERS

- dangling logic
- mixed duration types in one schedule
- project schedules
- task schedules
- resource schedules
- + many, many more

# Local models to the NASA gateway

ACT ONE · LOCAL (OLLAMA)

`llama3.2:3b` → `llama3.1:8b` → `qwen2.5:14b` → `qwen2.5:32b` → `llama3.1:70b`

Private by construction — the model lives on the workstation. The ceiling: model size vs the machine. Bigger brains, slower and still limited.

→

OPERATOR-ARMED

ACT TWO · NASA-APPROVED GATEWAY

**ITAR-authorized frontier models — Claude, GPT, Gemini — through the agency's approved API.**

`claude-opus-4.8-itar` · `claude-sonnet-5-itar` · `gpt-5.1-itar` · `gemini-2.5-pro-itar` · `gemini-3.5-flash-itar` + more

- Never a fallback — selected, acknowledged, and armed by the operator each launch
- Endpoint allowlist-pinned in code; anything else refuses to construct
- Every transmission logged locally — a hash and byte count, never the text
- Warning banner on every page while armed

**Stronger models sharpened the interpretation — the math never moved.**

Deterministic engine before, deterministic engine after.

# Lessons learned

### 01 Turn fixes into skills

Prose reminders decay; committed procedures don’t. Every hard-won fix — parsing rules, gate sequences, session habits — became a skill the AI reloads at the start of every session, so nothing is relearned.

### 02 Prove a test can fail

A guard once sat green for months while checking nothing. Now every new test is run against a deliberately broken build first — a check that has never failed proves nothing.

### 03 CUI guardrails from day one

Egress guards, allowlist-pinned endpoints, redacted logs — sovereignty is architecture, not a setting. Retrofitting it later is a rebuild, not a patch.

### 04 File formats are learnable

The .mpp/.xer wall at the start of this story is now just an import. Learn the format once, commit the knowledge, and it stays learned — in the repo, not in anyone’s head.

### 05 Challenge every answer

“Assume everything you just said is wrong — prove it.” The AI authors pass/fail tests, runs them in a sandbox, and only proven changes get pushed. Confidence is not evidence.

### 06 Commit state, not chat

A session that isn’t written down never happened. Decisions live in git as ADRs; every session ends with a handoff — so any model, any day, can pick up exactly where the last one stopped.

### 07 Match the model to the task

Frontier reasoning for architecture and audits; lighter, faster versions for routine edits. And let models audit each other — different models catch different mistakes.

### 08 References, not vibes

The only proof a number is right is matching the reference on the same input — 108/108 SSI checks, Acumen Fuse metrics reproduced, and the parity suite locked as a CI gate.

06 · FORWARD

# Still a work in progress

| | |
|---|---|
| DONE | The hardest part is behind us — native parsers, the deterministic engine, reference parity, the approved AI gateway. |
| IN VALIDATION | Some of what is built still has to be tested and validated before it counts. |

## After that, the only constraint left is my lack of imagination.

# Scaling and the road ahead

● IN DEVELOPMENT

## Compare projects, not just versions of one.

The tool was never restricted to one project at a time — the portfolio already holds many. The only thing that has prevented this development so far is **time and resources** .

PLANNED

· Cost + performance join the schedule: IPMR / IPMDAR, EV, EAC behavior

· Integration with existing NASA capabilities such as SMART

· One connected view, re-cut by role — executive, manager, analyst

· Project → site → program rollups with traceable lineage

THE AGENCY-SCALE VISION — EVALUATE ALL OF NASA'S SCHEDULES (A SAMPLE, NOT THE LIST)

HOW LONG THINGS REALLY TAKE

Average actual duration versus plan, by project type — a grounded reference for the next estimate.

WHERE TROUBLE STRIKES

Where issues typically occurred against baselines across the lifecycle — so reviews look where history says to look.

HEALTH ↔ OUTCOME

The correlation between a healthy schedule — DCMA-14 and NASA schedule-health standards — and the success of the finished project.

EARLY WARNING, EARLIER

A library of manipulation and slip fingerprints from past programs — so the patterns are recognized several updates sooner on the next one.

And beyond: new baselines scored against historical realism at approval, cross-center benchmarking, automated IBR prep packages, a living lessons-learned library — **the list keeps growing.**

“Its practical ceiling is defined less by a single tool or dataset than by the time and imagination available to connect NASA's information responsibly .”

# Live demo — the flight plan

01 **The launch sequence, live** lightshow + synthesized hum

02 **Import a stack of .mpp files** a folder becomes a project

03 **Walk the 12-chapter spine** import → briefing

04 **Trend + manipulation signals on Starlight** the fingerprint, live

05 **Bow Wave / CEI auto-play** watch slipped work push right

06 **Ask the AI — a cited answer, on the spot** your question, if time allows

# POLARIS 2

# Questions

**David J. Politte**

Scheduling SME — NASA Glenn Research Center

david.j.politte@nasa.gov · (757) 777-4585

github.com/polittdj/Schedule-Manipulation-Analysis-Tool-Experiment