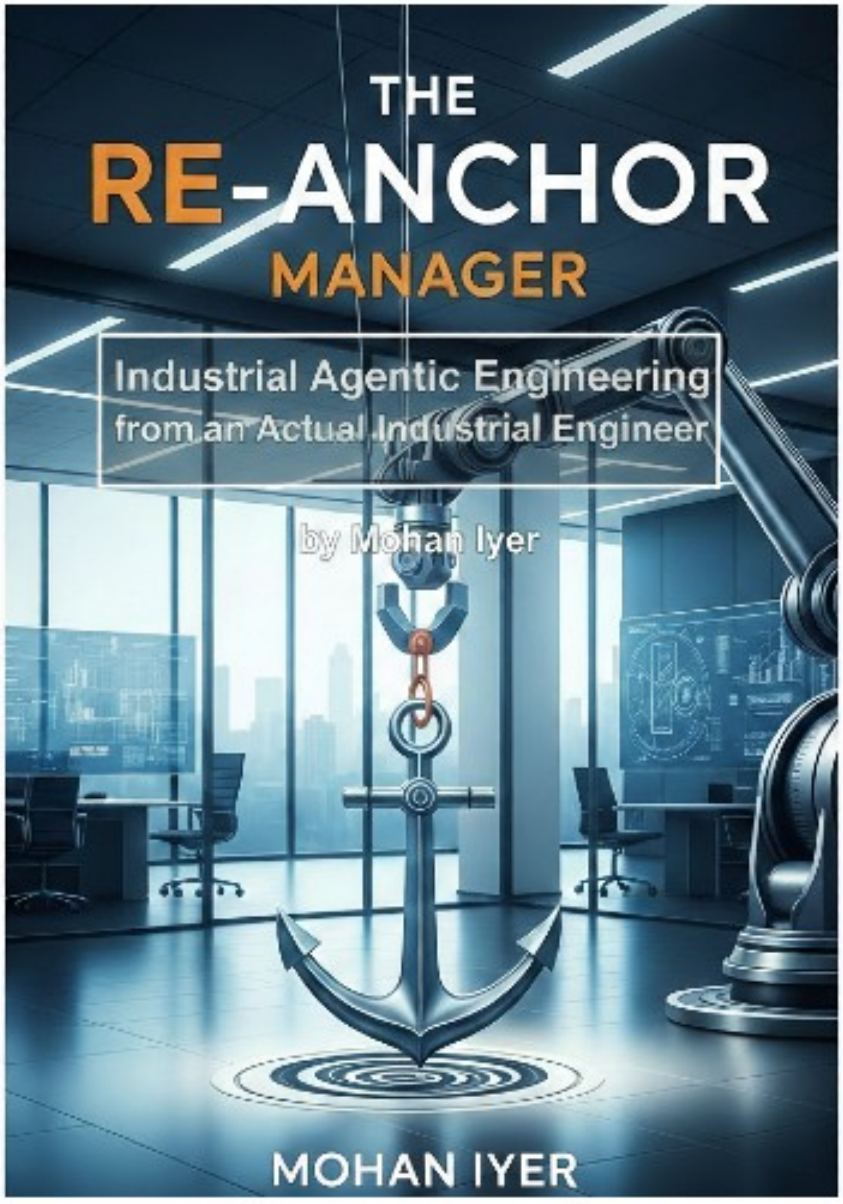
THE
RE-ANCHOR
MANAGER
Industrial Agentic Engineering
from an Actual Industrial Engineer
by Mohan Iyer
MOHAN IYER

Preface

I am a seventy-five-year-old industrial engineer with three decades in IT behind me. I spent fifty years across engineering and technology — factory floors, process-design rooms, and systems work — before I made this method my daily practice. When I started building software with large language models in December 2022, week one of ChatGPT's public release, I did not bring coding skill. I brought a different set of instincts: quality gates and shift-handover discipline. The kind of thing that keeps a plant running when the person who built it is asleep.

This booklet is the operating manual I built for myself after the first hundred development sessions kept hitting the same wall. None of it is theory. Every technique here came out of a real session, a real failure, and a real fix. I am not going to teach you better prompts. I am going to show you the infrastructure that makes every prompt land in the right context — so the AI you work with tomorrow knows exactly what you built today.

The whole method rests on one humble object: a structured document I call a *re-anchor*. Write one at the end of a session. Feed it back at the start of the next. That is the entire idea. What follows is what I learned turning that idea into something that survives a hundred sessions.

A note on how this was made: this booklet was written using the method it describes — the AI held the structure, I held the judgement, and a re-anchor carried the state between sessions. The current template and the latest edition both live at **consensus-press-ai.com**. They are free. Take them, adapt them, and send your improvements back.

Chapter 1 — The Problem

AI-Izheimer's

Around session thirty of a long build, I asked the AI to keep refactoring a module we had worked on for two days. It answered by asking me what the module did. Not because my instruction was unclear — because it had no idea we had ever worked together. As far as it was concerned, I was a stranger who had just walked in off the street.

That is not a glitch. It is the architecture working exactly as designed. Every conversation with a language model lives in a bubble. When a session opens, the model receives your message and its instructions — nothing else. No access to yesterday. No running log of your decisions. When the session ends, the window is cleared. The next one starts empty.

I call this AI-Izheimer's: the structural amnesia that wipes operational context between sessions. The name is deliberate. Like the human condition it echoes, it does not dull the intelligence — the AI is as sharp in session one hundred as in session one. It dulls the *continuity*. In session one hundred, it has no idea sessions one through ninety-nine ever happened. This will not be patched by a bigger model. It is a property of how stateless models interact with stateful work.

What it costs

The cost is not abstract. It shows up in four places. **Time** — without a continuity system, the first fifteen to thirty minutes of every session went to re-explaining the project; over a hundred sessions that is more than a full working week producing zero forward progress. **Decision**

integrity — without a record, the same questions get re-debated, and worse, the AI quietly reverses choices neither of you remembers making; in one project a naming convention I set in session twelve was silently overwritten in session twenty-three, and I did not catch it until session thirty-one. **Quality** — an AI that does not know your constraints writes code that violates them, each violation small, compounding across dozens of sessions into a quality crisis. **Morale** — there is a particular exhaustion in explaining something for the fifth time to a collaborator who greets you each morning with the cheerfulness of a brand-new hire while you carry the weight of a hundred sessions of shared history.

These costs compound. Lost time means fewer decisions per session, which means slower progress, more sessions, more context to lose. The longer a project runs without a continuity mechanism, the more of each session is eaten by re-establishing what should already be known.

Why platform memory does not fix it

Both Claude and ChatGPT now offer memory features. They help. They do not solve this. Platform memory stores *knowledge* — your name, your language, your preferred style. What it cannot hold is the precise operational *state* of a complex, moving project: which decision was made in which session, which bug was fixed and how, what the current priority is after the last three sessions. The distinction is knowledge versus state. Knowledge is static and tells the AI who you are. State is dynamic and tells the AI where you are. You need both — but only state lets you pick up exactly where you left off. A re-anchor is the state document.

The practice started as a frustrated workaround. Late one night in May 2025, after a long session where the AI was visibly degrading, I typed something like: *I'm going to wipe the session and start fresh. Please create a re-anchor for this so the next one can pick up where we are.* That message had frustration in it. It also had the seed of everything here. From that night on, every session ended with a re-anchor and every session began by parsing one.

Chapter 2 — What a Re-Anchor Is

An anchor holds a vessel against the currents that would carry it away. A re-anchor puts the project back in position after the tide of a new session has moved it. Every AI session is a tide. The document returns you to the right coordinates.

If AI-lzheimer's is the disease, the re-anchor is the de-amnesiac. It gives a forgetful model something it cannot hold on its own: *enduring memory* — not memory of every word, but of everything that matters. Concretely, a re-anchor is a structured document that captures the complete operational state of a session — what was built, what was decided, what failed, what was learned, and what comes next. At the start of the next session you feed it to the AI as its first input. The AI reads it, reconstructs its understanding, and signals it is ready to continue. That is the whole concept. The power is not in the idea — it is in the discipline of doing it every single session.

It helps to be clear about what a re-anchor is *not*. It is **not a transcript** — it records the outcome of a conversation, not the conversation; a three-hour session might be twenty thousand words of back-and-forth, while the re-anchor is a few hundred words of distilled state. It is **not a summary** — a summary tries to be shorter; a re-anchor tries to be *precise*, keeping what matters for next session and dropping what does not. It is **not a journal** — a journal records what happened; a re-anchor records what matters for what happens next. It is **not a prompt** — it is a factual statement of what is true about your project, read the way a

colleague reads a briefing. And it is **not permanent** — each session produces a new one that supersedes the last, while previous re-anchors become the archive.

My first re-anchor was a mess — copy-pasted fragments, a typo in the word itself, written by a tired engineer at midnight. Over about five weeks it grew, one failure at a time, into a structured format. Somewhere in there it stopped being a thing I wrote and became a system I operated. Early re-anchors were prose, and they worked poorly — the AI skimmed a paragraph, took what it thought mattered, and ignored the rest. A field like `blocker: payment processing pending` is unambiguous. A sentence like *we should probably look into payments at some point* is easy to overlook. You do not have to use YAML; Markdown or JSON work too. What you need is structure. Prose lets the AI interpret; structure forces it to parse.

The one question a good re-anchor answers: *if the AI wakes tomorrow with no memory of today, what does it need to know to continue without losing ground?* Answer that at every close, feed it back at every open, do it for fifty sessions, and you will have something most AI developers have never had — continuity.

Chapter 3 — The Session Protocol and Hydration

A re-anchor is a document. A protocol is a discipline. The document is useless without the discipline. Three phases, every session, no exceptions: open, work, close.

Open — with one line

My sessions open with a single line that names the project, the session number, and what to load. Something like:

> *ProjectName-54. Hydrate with memory, instructions, and the two most recent re-anchors.*

That one line triggers the whole opening phase. The project name and number orient the AI in the chain. *Memory* and *instructions* are whatever persistent layers your platform offers — the stable facts and standing rules you have saved. And the **two** most recent re-anchors, not one, are deliberate: one captures the current state; two give a sense of trajectory — where things were heading, what changed, which decisions are stable. I arrived at two by trial and error; one was often too little, three consumed context for diminishing returns.

Then I require the AI to *parse*, not just read — to actively reconstruct current state, open issues, settled decisions, and next steps. To enforce it, I make the AI emit a signal before any work begins: session number, project name, and its understanding of current status. If the signal is wrong, I correct it before a single task starts. Thirty seconds here saves thirty minutes of misaligned work later. The hardest rule is the simplest: **no productive work until the opening phase is complete.** Every time I have skipped this in a hurry, I have regretted it.

Hydration

There is a specific word for feeding context into a fresh session: *hydration*. Reading means the AI received the text. Hydrating means it absorbed, parsed, and reconstructed the state from that text. A student who read the textbook has done something different from one who can pass the exam. Hydration is the exam.

Hydration fails in four characteristic ways, and you must check for them because the AI will not tell you. **Stale** hydration — it loaded an old re-anchor and reconstructs a state behind reality;

the mechanical fix is to verify the session number matches what you expect. **Partial** hydration — it parsed but missed sections, usually because the document is too long; in my experience re-anchors under about three hundred lines parse reliably, and above four hundred partial loss becomes likely. **Hallucinated** hydration — it signals success but has filled gaps with assumptions; the most dangerous mode because it is invisible, and the reason the signal must include verifiable facts. **Context overflow** — the re-anchor plus supporting docs crowd out room for work, solved by summarising older sessions more aggressively.

Three quick checks catch almost everything, in under thirty seconds combined: verify the signal (correct session number, project, status); probe a decision (*what was decision D-14 and why?* — if it invents, hydration was partial or hallucinated); and check blockers against the re-anchor's list. I run the signal check every session and the others when the work depends on recent state.

Work, then close

During the running phase the rules are few but non-negotiable. Honour settled decisions — the AI must not silently reopen or contradict a choice in the re-anchor; if it thinks one should be revisited, it says so and waits. Flag new decisions as they happen, each with an identifier and a rationale. Track issues in real time. And know when to stop: an AI session degrades after about two to three hours as the window fills and attention to earlier material fades. When the AI starts contradicting something agreed an hour ago, that is the signal to close cleanly — a fresh session with a good re-anchor beats a long one fighting its own haze.

The closing phase determines the next session's success. Create the re-anchor *while the context is still live* — defer it and you have already lost the thing it was meant to preserve. Let the AI draft it; it has perfect recall of the current window and you do not. Then verify before closing: read it, confirm the achievements match what actually happened, the decisions carry their rationale, the blocker list is complete. I have caught errors at this step — an issue marked resolved when it was only worked around — that would have poisoned the next session. Finally, archive, never delete. The numbered files accumulate into a navigable history. The most common mistake is loving the format and skipping the protocol. The format can evolve; the protocol does not change.

Chapter 4 — The Re-Anchor Format

Let me show you what evolution looks like. One project's re-anchor for an early session ran four hundred and thirteen lines: a narrative paragraph up top, achievements listed flat, a section for files referenced. Twelve days later the same project's re-anchor was two hundred and eighty-three lines — thirty per cent shorter but denser. The narrative was gone. Achievements were grouped by workstream, each with a status and specific outcomes. The metadata linked explicitly back to the previous re-anchor. The format did not just shrink. It got smarter — from prose to structure, from implicit to explicit, from narrative to data.

A mature re-anchor has a consistent set of sections, so the AI always knows where to look. The foundation is seven: **identity** (session number, project, date, and the filename of the previous re-anchor, creating a traceable chain); **achievements** (specific and verifiable — not *worked on auth* but *implemented token auth with fifteen-minute expiry, tested, deployed to staging*); **decisions** (each with an identifier and a rationale — the section that prevents drift, because the AI cannot unknowingly contradict a choice it can see); **documents created**; **current status** (a present-tense snapshot, often with metrics); **blockers** (each with severity

and an owner; a blocker in three consecutive re-anchors is a pattern demanding a systemic fix); and **next steps** (the handover — what to start on, what must be true first).

Five qualities separate a good re-anchor from a bad one: density over length, specificity over generality (*deployed v2.3.1 with 27 tests passing*, not *made progress*), structure over prose, forward-looking over backward-looking, and verifiable over aspirational. And the format is a living thing — add sections when a failure mode demands one, drop sections that do not carry their weight. The only hard rule is consistency.

Four sections I added after the format kept failing me

The seven sections above are the foundation. The last three months taught me four more, each born from a specific recurring failure. These are the additions in the current public template at **consensus-press-ai.com**, and they are the most useful upgrades I have made in a year.

Behavioural invariants

The finding behind this is uncomfortable: as the volume of instructions grows, adherence degrades *uniformly*. A long, comprehensive rule document behaves like a human onboarding manual — read once, then drifted from by mid-session. Four minimal constraints, loaded first and re-read before every major action, outperform twenty paragraphs read once. So I put four invariants at the top of the re-anchor: **B1 — Assume nothing** (state assumptions before acting; if ambiguous, halt and ask); **B2 — Minimum scope** (take the smallest action that closes the gap; no speculative additions); **B3 — Surgical changes only** (touch only what the gap requires; no adjacent rewrites); **B4 — Evidence gate** (no *done*, *passed*, or *shipped* claim is valid without named proof — naming the action is not evidence). These are not context; they are execution constraints, re-read before any closure. On a self-detected violation: halt, name which invariant broke, surface it, and do not resume until cleared.

Deprecated facts

The slowest, most expensive failure is *stale-state contamination* — the AI citing something that used to be true. The fix is a section listing facts that have changed, loaded *first*, before any other state is read. Each entry records what was true, why it is now false, what replaced it, and when. Once this section is loaded, citing a deprecated fact is a governance violation. Marking the dead facts before reading the live ones stops the AI building on yesterday's truth.

Artefact registry

This is the source-of-truth packet: the list of files that are authoritative for the project — permanent governance documents that never expire, canonical specifications that load before any work, session artefacts that expire when superseded, and deprecated artefacts that must never be loaded again. The rule that makes it powerful: if session state ever conflicts with a registered artefact, *the artefact wins* — and if a canonical file is missing or version-mismatched, halt and flag rather than proceed on stale ground.

The rehydration acceptance test

Loading a document is not the same as loading it correctly. This is the integrity certificate for the session. Before any work, the AI answers a short fixed set of questions from memory, then verifies each against the loaded re-anchor — the project and its one-line goal, the highest-priority blocker, the most recent locked decisions, the canonical source-of-truth file, the exact first action, and the four behavioural invariants. Any mismatch halts that question and triggers

a re-read; two or more halts, the AI rehydrates from the top. A session that skips this test has no hydration guarantee — only the *feeling* of one, which is exactly the hallucinated-hydration trap.

A final detail that seems trivial and is not: name your files consistently. I use session number, project name, session number — `42_myproject_42.yaml`. The repetition makes a file identifiable from the name alone and sorts naturally, and it lets me say *parse re-anchor forty-two* and have the AI know exactly what to load. The full, current template — with all of these sections laid out and commented — is free at **consensus-press-ai.com**. Start from it rather than rebuilding it from this description.

Chapter 5 — Getting Started in Fifteen Minutes

The fastest way to use this booklet

You do not have to read this cover to cover. Because the whole method is about working with an AI, you can let an AI do the reading for you. Drag this file into a chat window with any capable model and give it one instruction:

> *Step me through this manual. Do not deviate from the contents of the manual.*

The model will walk you through it section by section, answer your questions against the text, and — if you ask — help you write your first re-anchor on the spot. It is the method demonstrating itself: structured content, loaded into context, driving the session. Read on if you prefer to do it by hand; the steps are below.

By hand

Step 1 — Create your first re-anchor. At the end of your next session, ask the AI for a structured summary with five sections: session metadata (number, date, project), achievements (what was actually completed), decisions (what was chosen and why), blockers (what is preventing progress), and next session (what to do next). Ask for YAML or Markdown. Save it as `01_myproject_01.yaml`. Then review it: do the achievements match what was *completed*, not attempted? Do the decisions include rationale?

Step 2 — Use it to open the next session. Upload the re-anchor as project knowledge (in a Claude Project) or paste it at the very start. Open with the single line from Chapter 3 — project, number, and what to load — then tell the AI to parse it and report its understanding. Check three things: correct session number, project name, current priority. If any is wrong, correct it before proceeding. Ten seconds of verification saves an hour. It is the cheapest quality gate in the method.

Step 3 — Build the habit. The protocol is a habit, not a feature: open (load, verify the signal), work (record decisions as they happen), close (produce the re-anchor, name the session, save). For the first five sessions it feels like overhead. By session ten it feels natural. By session twenty you will not imagine working without it. The key is making the closing ritual non-negotiable — the temptation to skip it is strongest when you are tired at session end, which is exactly when the context is freshest and the re-anchor cheapest to produce.

Step 4 — Use what your platform offers. If your platform has persistent layers — saved facts and standing rules that survive between sessions — use them: stable facts that rarely change in the memory layer, standing rules you want enforced every time in the instructions

layer, and the session-by-session state in your re-anchors. The principle is universal: separate knowledge from enforcement from state.

A few mistakes to avoid, all of which I have made: writing re-anchors in prose (the AI interprets prose and parses structure); skipping re-anchors on short sessions (an unrecorded decision is a forgotten one — a five-line re-anchor still needs to exist); making them too long (promote stable information out of the re-anchor); and skipping the hydration check (you are then trusting the AI to have parsed correctly with no evidence — misplaced trust often enough to hurt over a long project). If you want a head start, the full template is free at **consensus-press-ai.com**.

Chapter 6 — What Comes Next

This method was built for a world where AI agents forget everything between sessions. That world is changing — memory features are improving, context windows are growing. Does that make re-anchors obsolete? No, for two reasons that have run through this whole booklet.

First, platform memory is general-purpose: it stores *this person works in Python*. A re-anchor stores *this session completed the first sprint; twenty-eight checks ran, all green; blocker BLK-01 is resolved; next priority is to curate the evidence.* The former is biography; the latter is a shift-handover log. Second, a bigger context window does not create *intent*. A re-anchor is not just compressed history — it is a curated statement of what still matters. Dump raw history into a vast window and you delegate that curation to the model, which has no basis for deciding what is operationally critical to your project; it attends to what is statistically interesting, not to what matters. The parallel to industrial engineering is exact: factories got smarter and machines got more capable, and shift-handover logs did not become obsolete — they became *more* important, because the systems they governed grew more complex.

So: start with one re-anchor. End every session by writing it. Begin every session by feeding it back. The system grows from there — one session at a time, one failure at a time, one fix at a time. It did for me, across more than three thousand conversations. It will for you.

The re-anchor is one idea in the quiver, and there is more coming. I am writing a nine-part essay series on a related theme — why *fluency* in an AI's output is not the same as *authority* over it, and the disciplines that close the gap between sounding right and being right. The current template, the latest edition of this guide, and the series all live at **consensus-press-ai.com**. If you adapt the practice for your own work, I would like to hear how.

A Compact Glossary

AI-Izheimer's — the structural amnesia that wipes operational context between AI sessions. It does not dull the intelligence; it dulls the continuity.

Re-anchor — a structured document written at session close, capturing state, decisions, blockers, and next steps. The de-amnesiac: it gives a stateless AI enduring memory across sessions that have none of their own.

Enduring memory — the benefit a re-anchor delivers. Not recall of every word, but reliable carry-forward of everything that matters for continuing the work.

Hydration — feeding the AI a state document at session start and *verifying* it has reconstructed the context. Reading is receiving the text; hydrating is passing the exam.

Session protocol — the three-phase discipline applied every session: open (load and verify), work (honour and record decisions), close (write, verify, save the re-anchor).

Behavioural invariants (B1–B4) — four minimal execution constraints loaded first and re-read before every major action: assume nothing, minimum scope, surgical changes only, evidence gate.

Deprecated facts — a section listing facts that have changed, loaded first so the AI marks the dead truth before reading the live state. Citing one after loading is a governance violation.

Artefact registry — the source-of-truth packet listing authoritative files. If session state conflicts with a registered artefact, the artefact wins.

Rehydration acceptance test — a short fixed set of questions the AI answers from memory and verifies against the re-anchor before any work. The integrity certificate for the session.

About the Author

Mohan Iyer is an industrial engineer — M.Tech in Industrial Engineering, and an MBA — with fifty years across engineering, technology, and management, three decades of them in information technology. He began working with large language models in December 2022, within the first week of ChatGPT's public launch, and has since accumulated more than three thousand conversations across multiple platforms. His approach draws on decades of industrial-engineering and IT discipline: structured processes, measurable outcomes, and the conviction that quality comes from systems, not luck.

He lives in Auckland, New Zealand. The free template, the latest edition of this guide, and more on the work it grew out of are at **consensus-press-ai.com**. For questions or to share how you have adapted the practice, write to mohan@pixels.net.nz.

The re-anchor method is free to use and adapt. If it helped, claps and shares are genuinely appreciated — fifty is the cap, just saying. Send your improvements back; the language and the practice belong to the people who use them.