
Large Language Models for Business and Finance

Architecture, Failure Modes, and Defensible Analysis

Monarch Wadia · monarchwadia.com

SECTION 0: OVERVIEW

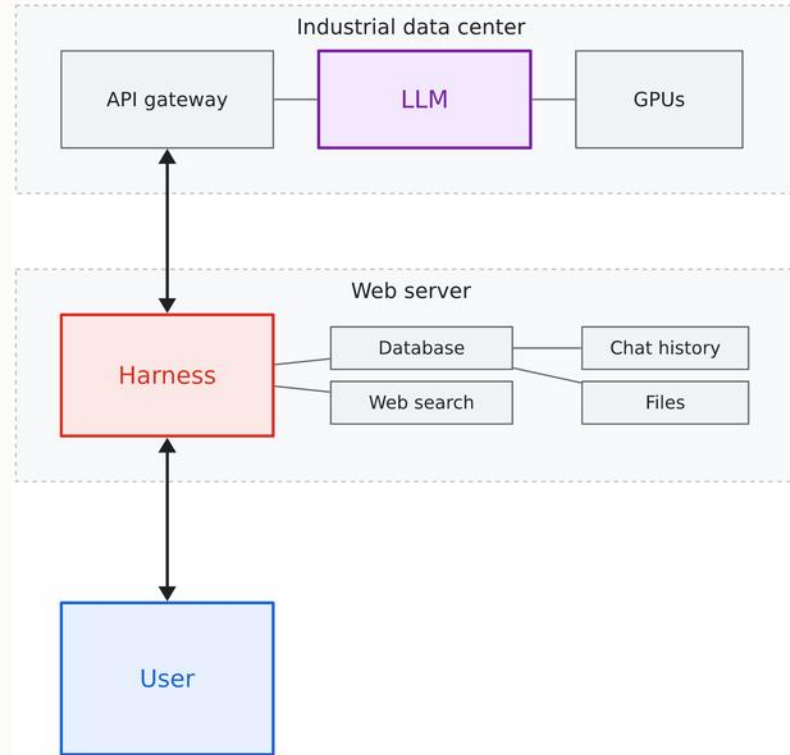
Architecture of Probabilistic Finance

Schedule: 10:00 to 10:15 | Foundations, Tools, and Auditable AI

Determinism Versus Probabilistic Fluency

- Finance runs on exact arithmetic and verifiable paper trails.
- Spreadsheets recalculate the same total every time because formulas are mechanical.
- Language models predict likely words, not audited facts.
- A fluent sentence easily hides an invented number or a dropped footnote.
- We pair word prediction with deterministic tools to survive audit.

Unbundling the Chat Interface



- Your chat screen packages three separate parts into one window.
- The prediction engine estimates word odds and holds zero memory.
- The application harness stores your past messages and sends them again on every turn.
- Managing risk requires taking these three parts apart.

One Running Specimen Across Every Chapter

- Every lesson anchors to one filing: the Acme Industrial Technologies 10-K.
- One revenue figure runs beside our work: \$4,280 million.
- Note 8 lists funded debt (\$2,800M), cash (\$360M), and the 3.50x covenant ceiling.
- We watch unguided models read, alter, and invent details from this text.
- Using one filing provides a stable benchmark for every test.

Three Lines of Defense in Financial AI

- First line: The analyst writes strict prompts, freezes source files, and forces tool use.
- Second line: The risk officer checks citations, calculation traces, and reconciliation tables.
- Third line: The firm enforces data boundaries and model risk governance standards.
- Software never carries legal or fiduciary liability under securities law.
- The human signing officer remains solely responsible for every published number.

SECTION 1: CYCLE 1

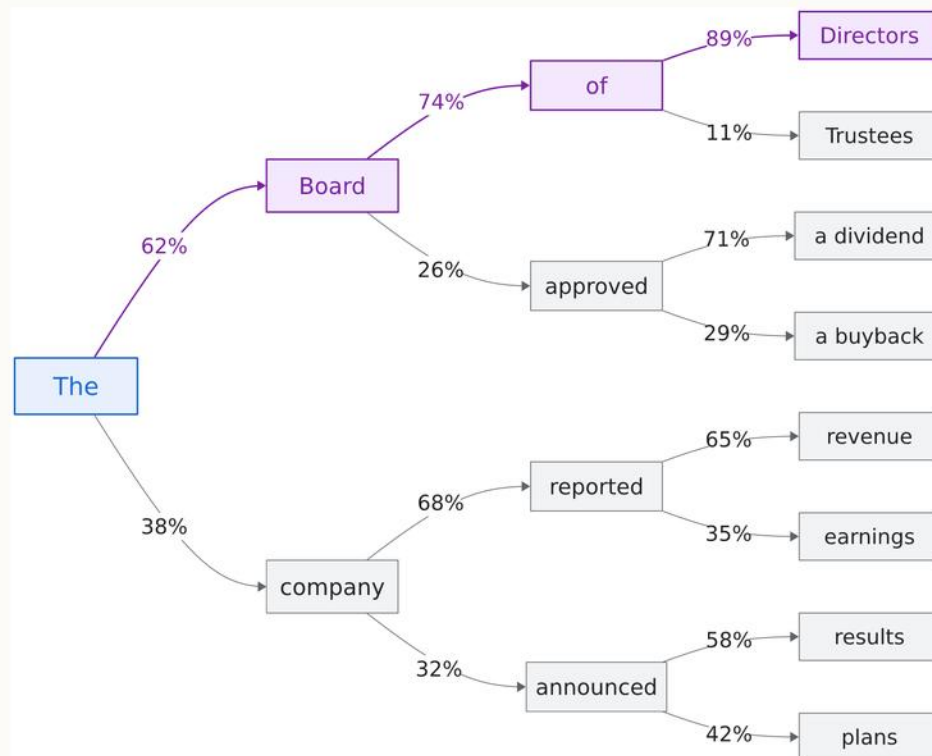
Foundations: Prediction and Variance

10:15 to 10:35 | Tokens, Logits, Sampling, and Hallucination Trees

Estimating Conditional Probabilities

- A language model does not look up records in a database.
- Text divides into token fragments that represent common word pieces.
- The neural network scores the probability of each candidate next token.
- Financial phrases appear frequently because training corpora contain millions of disclosures.
- Good grammar is statistical probability, not proof of financial reasoning.

Generation Without Financial Reasoning



- Each chosen word becomes context for the next token prediction.
- The loop runs step by step: calculate odds, sample a token, repeat.
- Standard syntax carries high odds: dividend reliably follows declared a quarterly.
- Flawless sentences can carry completely invented accounting figures.

Logits Determine Candidate Probabilities

- The model outputs raw scores called logits for every token in its vocabulary.
- A softmax formula converts raw logits into percentage odds that total 100%.
- High-scoring tokens match common patterns in financial text.
- Low-scoring tokens represent rare, specialized, or unexpected words.
- The model draws candidate tokens from this calculated distribution.

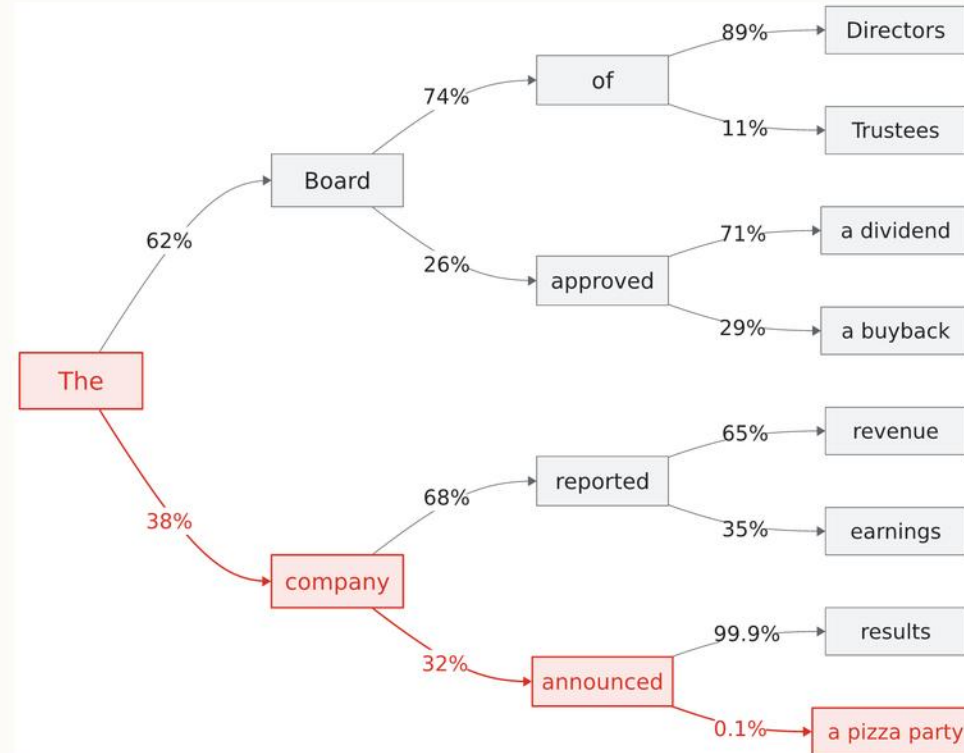
```
"The Board of Directors declared a quarterly..."  
--> dividend (88.4%)  
--> cash ( 7.6%)  
--> distribution ( 2.8%)  
--> share ( 0.9%)  
--> pizza ( 0.1%)
```

Temperature Controls Sampling Randomness



- Temperature scales logit differences before the model draws a token.
- High temperature flattens differences, letting unusual words land on the page.
- Low temperature steepens differences, concentrating odds on the top candidate.
- Zero temperature picks the single highest-scoring token every step.

Early Sampling Variations Branch and Diverge



- Text generation follows a branching tree of possible next words.
- An unexpected token choice early in a sentence changes all following odds.
- The machine never stops or backtracks when an odd word is drawn.
- Hallucinations are the natural mathematical result of probability branching.

Run-to-Run Variance Breaks Reconciliation

- Pasting the exact same prompt can yield different numbers on repeated runs.
- Multi-digit numbers split into multiple token fragments.
- Picking an alternative fragment in the middle changes the entire total.
- An operating margin can shift from 15.0% to 14.8% between runs.
- Financial systems cannot rely on stochastic word sampling for numbers.

Lab 1: Observing Variance on Identical Prompts

- Time budget: 7 minutes hands-on in your Goose workspace.
- Submit the self-contained prompt below without attaching outside files.
- Run the identical prompt three times in three clean sessions.
- Compare the exact revenue numbers, citations, and wording across the three runs.
- Notice how the model sounds equally confident on every run, even when numbers drift.

Document:

Acme Industrial Technologies (NYSE: AIT) reported fourth-quarter consolidated revenue of \$4,280 million, a 6% increase compared to \$4,038 million in the prior-year period. Operating income expanded to \$640 million from \$595 million, reflecting price realization across heavy equipment lines.

Instructions:

Summarize the fourth-quarter performance for Acme Industrial Technologies. State the total revenue and the year-over-year growth rate. Use two concise sentences.

Executive Rule: Probabilistic Text Is Never Math

- Never allow unguided text generation to serve as the source of balance sheet data.
- Treat every generated number as an unverified draft until checked against primary filings.
- Remember that model confidence reflects grammatical probability, not factual truth.
- Require deterministic tools whenever numbers must be calculated or verified.
- Put human review gates in front of every customer-facing or regulatory document.

Cycle 1 Checkpoint: Questions and Synthesis

- **Target Clock** : Concludes at **10:35 AM** .
- Key takeaway: Word prediction explains why models invent numbers with total confidence.
- Core rule: Check every extracted number against primary source text.
- Next block: Cycle 2 begins immediately at 10:35 AM.
- Questions on tokens, probability trees, or run-to-run drift?

SECTION 2: CYCLE 2

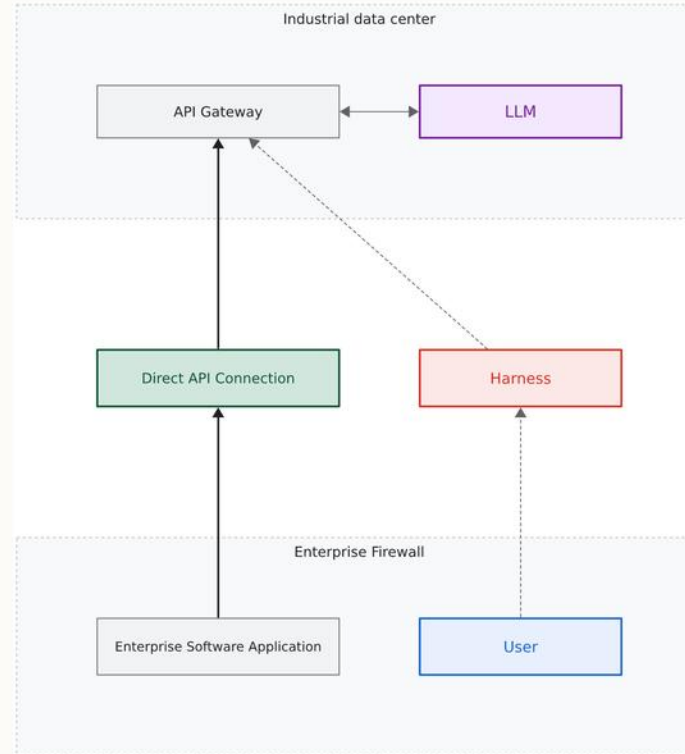
Multi-Turn: The Stateless Courier

10:35 to 10:55 | Statelessness, Quadratic Costs, and Attention Decay

Models Retain Zero Memory Across Requests

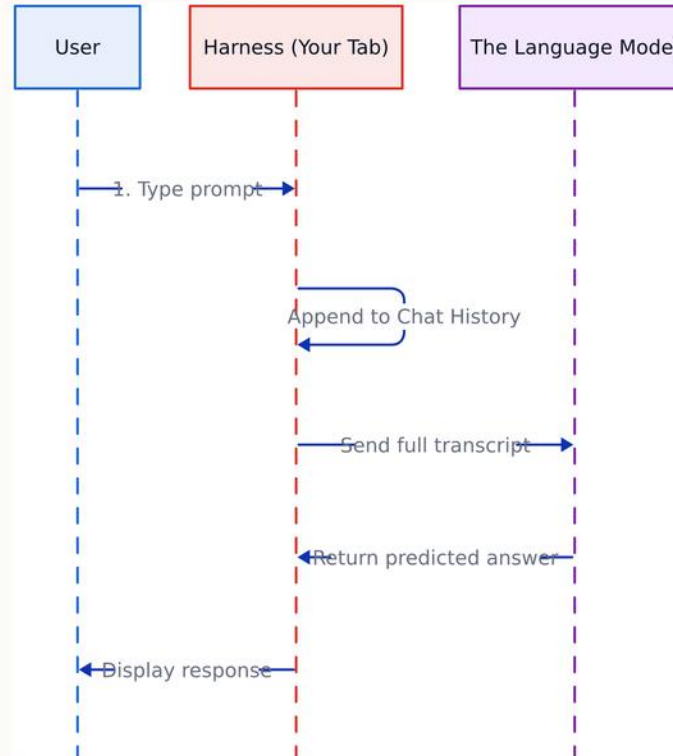
- Cloud language models are stateless HTTP services.
- The model keeps no memory of your earlier questions or its own previous answers.
- Each call runs in temporary server memory and vanishes as soon as the answer ships.
- Conversation history exists only because your laptop saves the transcript.
- Ongoing chat is an illusion created by your client application.

Harness Reships History on Every Turn



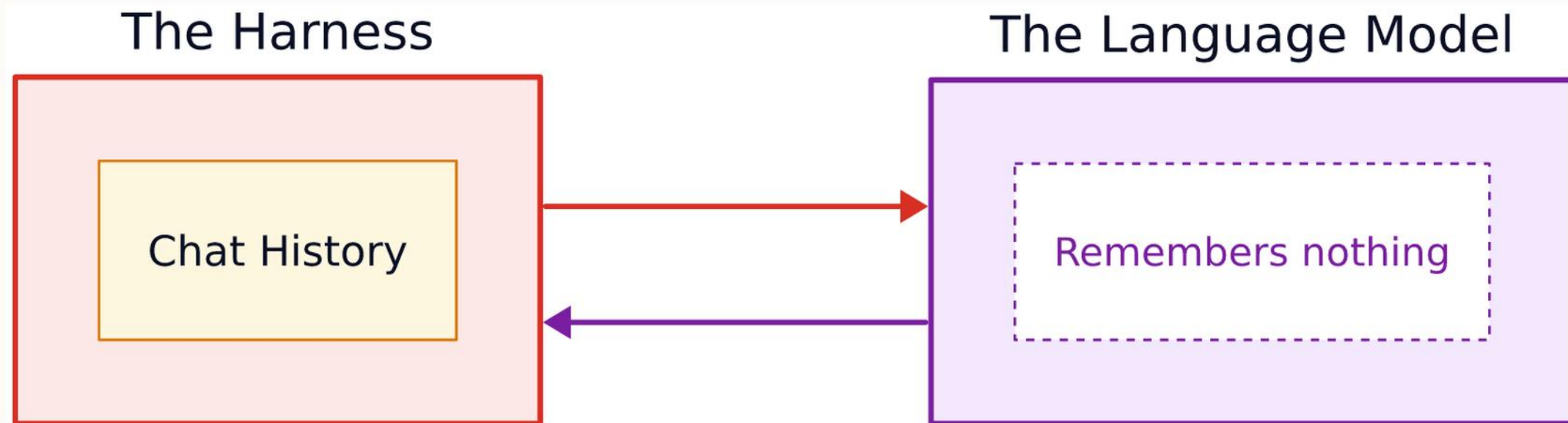
- When you type a follow-up question, your software bundles all prior messages.
- The growing list of messages travels across the network as a new HTTP request.
- The model reads the entire transcript from the beginning before writing the next word.
- Long chat sessions resend hundreds of redundant words on every keystroke.

Transcripts Flow in a Continuous Loop



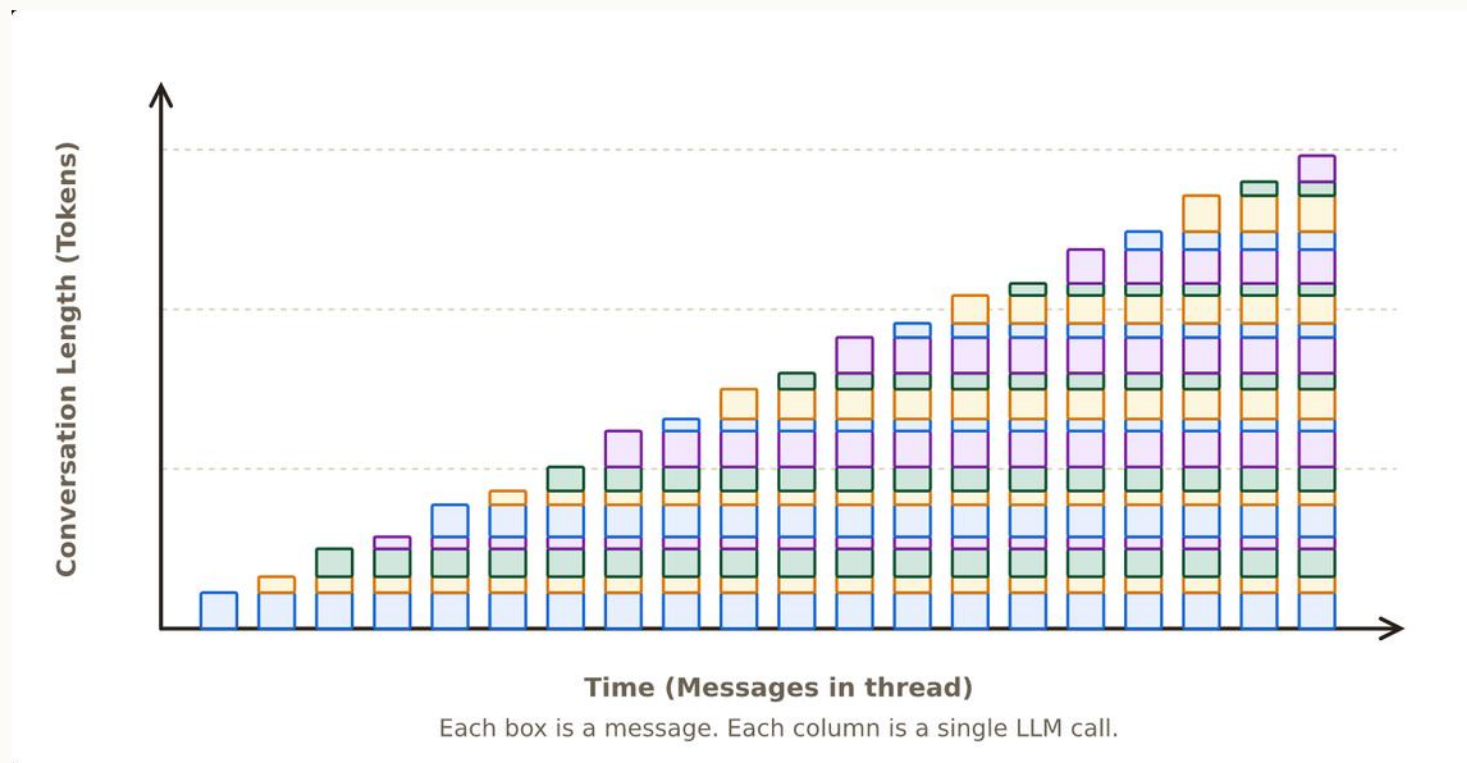
- Your software acts as a courier carrying messages back and forth.
- The transcript grows in your computer memory while the remote model stays blank.
- Early errors and discarded ideas remain trapped inside the transcript.
- Managing conversation state in local code is essential for clean work.

Memory Resides in the Host Application



- The split between model inference and application memory is absolute.
- The host software owns the transcript, login credentials, and file paths.
- The model sees only the text payload passed to its API endpoint.
- Closing your browser tab or resetting your session wipes all active context.

Token Usage Scales Quadratically in Chat



- In multi-turn chat, you pay for previous tokens again on every turn.
- A 10-turn conversation creates a quadratic cumulative token curve.
- API bills explode when analysts leave long chat threads open for days.
- Unmonitored transcript growth creates serious cost and speed problems.

Attention Decays in Long Transcripts



- Context windows have a maximum capacity fixed by model architecture.
- Attention does not spread evenly across giant transcripts.
- Models pay close attention to the start and end of text, but fade in the middle.
- Footnotes and debt covenants buried in the middle of long chat history get overlooked.

Lab 2: Testing Precision in Long Transcripts

- Time budget: 8 minutes hands-on in Goose.
- Send the five prompts below one turn at a time.
- Do not paste all five prompts at once; let conversation history build up.
- In Turn 5, check whether the model still respects the currency rule from Turn 1.
- Watch your session token count jump after Turn 5.

Turn 1: Note that Acme Industrial Technologies reports in USD, but European sales occur in EUR.

Turn 2: For Acme Industrial Technologies in Q4, what was total reported revenue?

Turn 3: What was operating income for the fourth quarter, and what drove margin expansion?

Turn 4: Did Acme disclose any supply chain friction in European markets?

Turn 5: Reconcile European currency exposure **using** our Turn 1 note.

Executive Rule: Fresh Sessions for New Tasks

- Use one clean session per analytical task.
- Close and restart chat sessions when switching companies, deals, or fiscal years.
- Stop old assumptions from leaking into new valuations.
- Track team token usage to catch runaway chat threads early.
- Summarize long background files before pasting them into new prompts.

Cycle 2 Checkpoint: Questions and Coffee Break

- **SCHEDULE : COFFEE BREAK: 10:55 AM TO 11:10 AM (15 MINUTES)**
- **Target Clock** : Concludes at **10:55 AM** .
- Take a quick break to step away and stretch.
- Next block: Cycle 3 begins promptly at 11:10 AM (Grounded Context & Filings).
- Key takeaway: The remote server forgets everything; your laptop re-sends history at growing cost.
- Questions before we break on statelessness, token costs, or attention decay?

SECTION 3: CYCLE 3

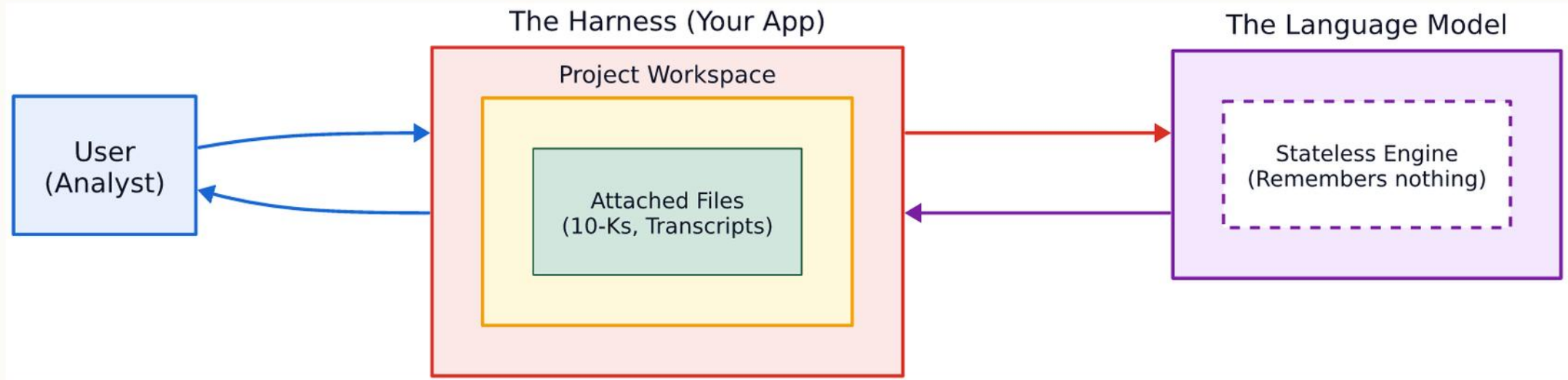
Grounded Context: Projects and Knowledge

11:10 to 11:35 | Project Boundaries, Primary Filings, and Information Perimeters

Pre-Training Weights Cannot Replace Filings

- Training data comes from old web text frozen on a past date.
- Foundation models do not know what was filed on corporate registries this morning.
- Training text often mixes up company targets with audited historical results.
- Asking a model to recall balance sheet data from memory leads to date errors.
- Defensible work requires feeding the model the exact source filing.

Project Boundaries Isolate Reference Files



- A project workspace draws a clear boundary around model work.
- Analysts attach primary source files: annual reports, credit agreements, and notes.
- Prompts instruct the model to answer only using the attached files.
- The project boundary acts as an information barrier for confidential deal data.

Long Filings Challenge Attention Windows

- Modern models advertise context windows over one million tokens.
- Dumping a full 100-page annual report into one prompt strains attention.
- Complex tables, multi-column pages, and small footnotes test retrieval limits.
- Models easily spot headline revenue but miss restructuring notes below the table.
- A large context window does not guarantee that every footnote is read.

Vector Search Chunks and Loses Footnotes

- Retrieval systems split large PDFs into small text snippets stored in databases.
- Vector search retrieves snippets that match the wording of your question.
- Footnotes frequently get separated from the tables they explain.
- Chopping up a PDF breaks table rows and cross-references.
- Analysts must check if their search tool reads whole filings or lossy text pieces.

Direct Ingestion Preserves Context

- Feeding clean, whole sections to the model keeps accounting context intact.
- Supplying full financial statements keeps notes linked to their line items.
- Markdown and text tables keep column headers and currency labels clear.
- Telling prompts to quote exact page lines makes human verification fast.
- Clean document ingestion forms the bedrock of reliable extraction.

Lab 3: Grounding Queries in Primary Filings

- Time budget: 8 minutes hands-on in Goose.
- Paste the text below into Goose as a single self-contained prompt.
- The prompt includes the exact filing excerpt from Note 8 directly in the text.
- Check if the model extracts the 3.50x leverage covenant ceiling accurately.
- Verify that the model cites the exact Note coordinates from the prompt.

Document: Note 8 (Debt Covenants and Credit Facilities), Acme Industrial Technologies 10-K:

Under the Senior Credit Facility, consolidated net funded debt (funded debt less cash and equivalents) to EBITDA shall not exceed 3.50 to 1.00 at the close of any fiscal quarter. Consolidated funded debt stood at \$2,800 million against cash and equivalents of \$360 million. Consolidated EBITDA for the trailing twelve months reached \$1,220 million.

Instructions:

Identify the restrictive debt covenant governing Acme Industrial Technologies. State the covenant ceiling and current net debt balance.

If any figure is not explicitly disclosed in the document above, output 'NOT FOUND'.

Executive Rule: Primary Documents Must Be Frozen

- Record the SHA-256 hash of every filing used in an analysis memo.
- Stop analysts from relying on public web search for audited figures.
- Keep proprietary deal memos and models inside isolated corporate networks.
- Confirm that enterprise AI agreements guarantee Zero Data Retention for your files.
- Audit team folders to ensure files do not leak across deal perimeters.

Cycle 3 Checkpoint: Questions and Grounding Review

- **Target Clock** : Concludes at **11:35 AM** .
- Key takeaway: Never rely on public model memory for numbers; ground in frozen filings.
- Core rule: Whole document sections protect footnotes better than fragmented vector search.
- Next block: Cycle 4 begins immediately at 11:35 AM (Tools & Math Delegation).
- Questions on project boundaries, RAG limitations, or Zero Data Retention?

SECTION 4: CYCLE 4

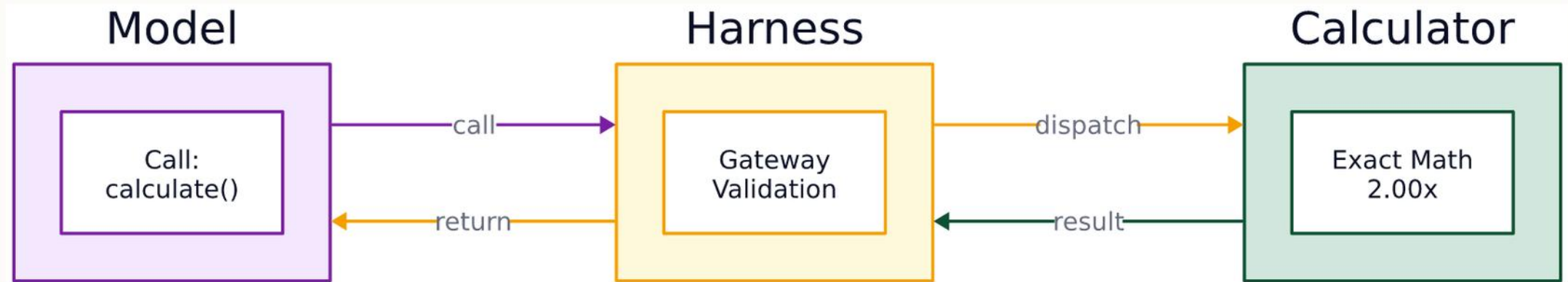
Tools and Connectors: External Execution

11:35 to 12:05 | MCP Tools, Compounding Decay 0.98^n , and the Four Checks

Language Models Cannot Execute Reliable Arithmetic

- Language models predict sequences of words; they do not calculate numbers.
- Multiplication, compound growth rates, and currency exchange require exact code.
- Ask a model to calculate debt-to-EBITDA, and it guesses a plausible number.
- High training frequency helps with basic addition, but multi-digit math often fails.
- Regulated finance systems must never let models compute math in their heads.

Model Emits Arguments, Runtime Runs Code



- Tool calling connects text prediction engines to exact software programs.
- The model notices when a prompt requires a calculation.
- It outputs structured parameters listing the tool name and numbers.
- The verified mathematical answer returns to the model to complete the memo.

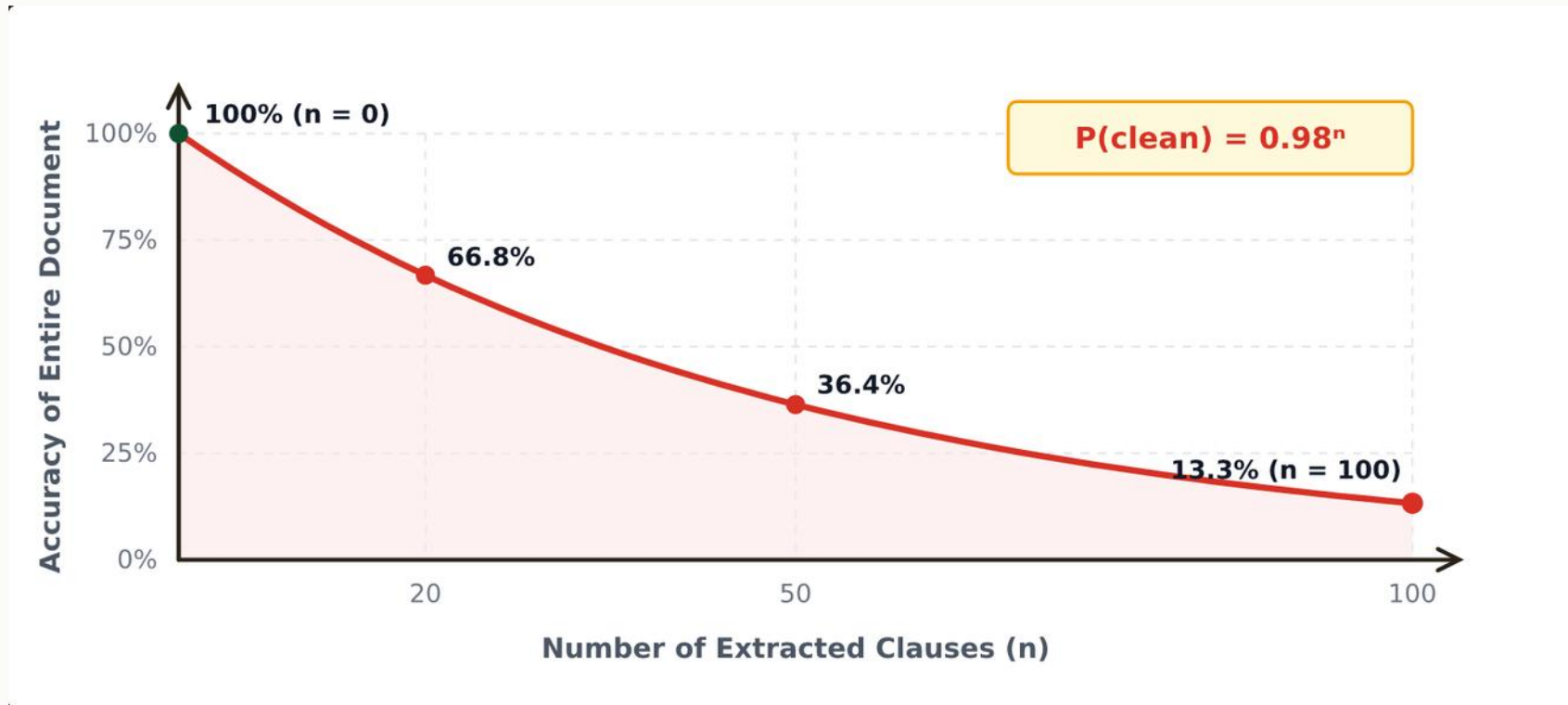
Model Context Protocol Standardizes Tools

- The Model Context Protocol provides an open standard to connect models to tools.
- MCP servers expose standard tools: calculators, SQL databases, and market feeds.
- The protocol standardizes tool descriptions, parameter checks, and security rules.
- Local runtimes enforce permissions, logins, and argument safety.
- Standard tools let firms switch model vendors without rewriting tool integrations.

Sequential Steps Compound Small Error Rates

- High accuracy on single questions does not guarantee success on multi-step tasks.
- A 98% factual accuracy score sounds reassuring in sales demos.
- Real credit reviews require dozens of sequential extractions and calculations.
- Error odds compound across each step in an analytical chain.
- An unverified pipeline with dozens of steps is mathematically doomed to fail.

Compounding Error Decays Overall Accuracy



- If an extraction step achieves 98% accuracy, it carries a 2% error rate.
- Across 50 steps, overall clean accuracy decays to 36.4% (0.98^{50}).
- Across 100 steps, clean accuracy drops to 13.3% (0.98^{100}).
- Verification gates are mandatory to prevent compound failure in production.

Four Checks for Defensible Output

1. Citations



2. Dates



3. Schema



4. Caveats



- Deploy four mandatory quality gates before publishing any AI draft.
- Provenance Check: Match every number to an exact quote and page coordinate.
- Arithmetic Delegation Check: Verify that all calculations ran in external code.
- Negative Confirmation Check: Actively search for absent covenants or missing caveats.

Provenance and Delegation Stop Inventions

- The Provenance Check requires unbroken lineage from output text to source pages.
- If an extracted number lacks an exact document citation, reject the draft.
- The Arithmetic Delegation Check verifies that an external calculator ran.
- Flag any memo showing financial ratios without matching code execution logs.
- Automated filters catch fabricated numbers before analysts spend time reviewing drafts.

Sanity Checks and Negative Confirmation

- Models often mix up currency units: thousands, millions, and billions.
- The Unit Sanity Check confirms scale and flags sudden magnitude jumps.
- Negative Confirmation forces the model to state clearly what disclosures do NOT appear.
- It explicitly looks for missed debt defaults, lawsuits, or accounting restatements.
- Verifying the absence of adverse items protects firms from costly blind spots.

Lab 4: Delegating Math to a Python Tool

- Time budget: 8 minutes hands-on in Goose.
- Make sure the Goose developer code execution tool is turned on.
- Send the prompt below; it contains all numbers needed for the calculation.
- Require the agent to write and execute Python code rather than guessing in text.
- Check that the final answer displays the Python code and exact terminal output.

Data:

Acme Industrial Technologies Financial Figures (FY2025):

Funded debt: \$2,800 million.

Cash and cash equivalents: \$360 million.

Trailing twelve-month EBITDA: \$1,220 million.

Note 8 Covenant Ceiling: 3.50x net leverage.

Instructions:

Calculate Acme's net leverage ratio using Python:

```
net_debt = 2800 - 360
```

```
net_leverage = net_debt / 1220
```

MANDATORY EXECUTION RULE:

Do not calculate the ratio in narrative text.

Write and execute a Python script in your sandbox to compute the ratio.

State whether Acme remains in compliance with its 3.50x ceiling.

Display the executed script and exact stdout terminal output.

Executive Rule: Reject Weights-Based Math

- Demand an architecture review of any commercial financial AI tool.
- Ask vendor engineers if math happens inside model weights or in sandboxed code.
- Reject any platform that demonstrates financial calculations without code logs.
- Make sure external tool connectors include strict parameter checks and limits.
- Require all financial metrics to pass provenance and unit checks.

Cycle 4 Checkpoint: Questions and Tool Execution Review

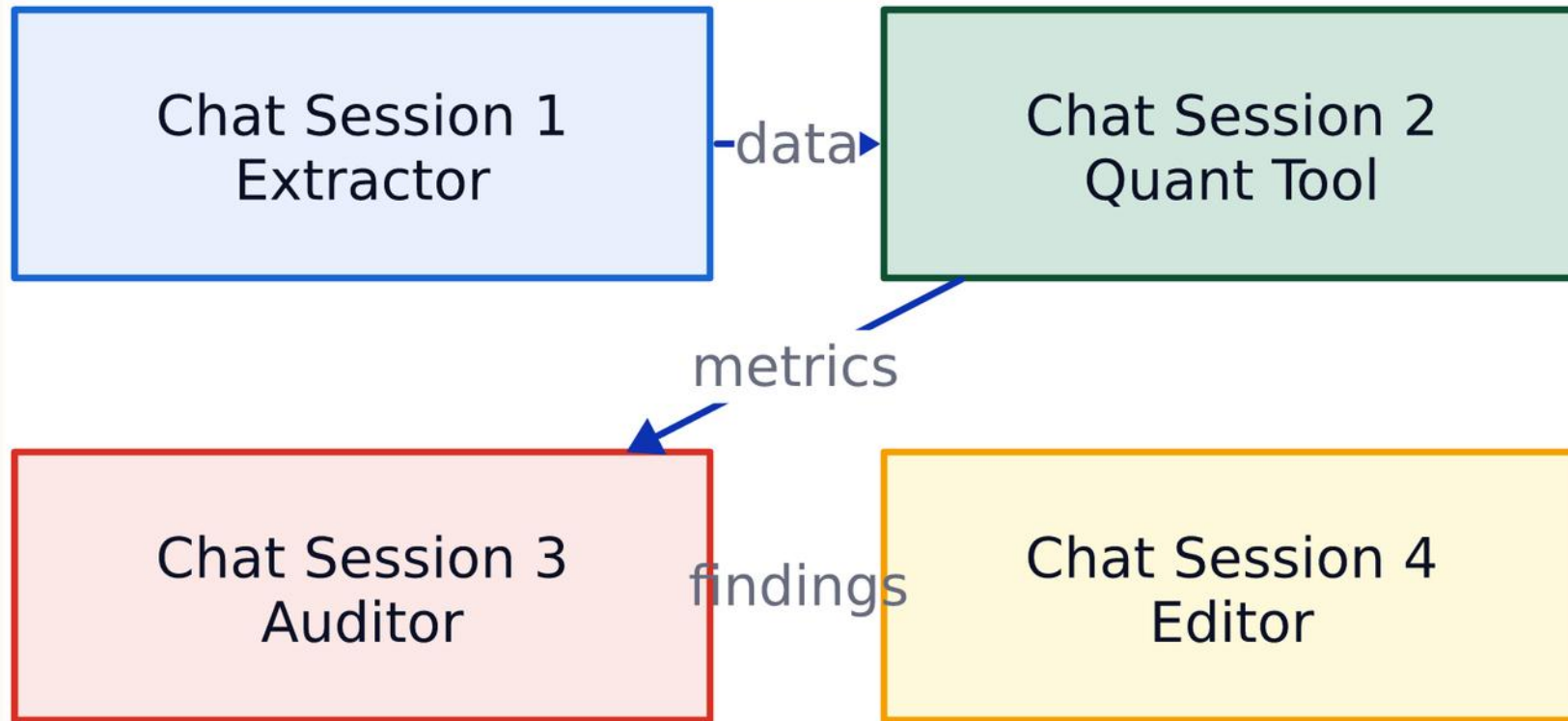
- **Target Clock** : Concludes at **12:05 PM** .
- Key takeaway: Models read and draft; deterministic code calculates. Never allow mental math.
- Core rule: Apply the Four Checks (Provenance, Arithmetic, Units, Negative Confirmation).
- Next block: Cycle 5 begins immediately at 12:05 PM (Sub-Agents & Governance).
- Questions on tool calling, Python execution, or error compounding?

SECTION 5: CYCLE 5

Sub-Agents: Division of Labor

12:05 to 12:30 | Chat Sessions, Immutable Lineage, and Audit Dossiers

Agents Are Distinct Chat Sessions



- An autonomous agent is not a sentient being; it is code orchestrating prompts.
- Multi-agent pipelines execute distinct prompts across clean, separate chat sessions.
- Each session carries a specialized prompt, isolated context, and specific tools.
- Data moves between sessions as structured tables or text files.

Specialized Roles Outperform Mega-Prompts

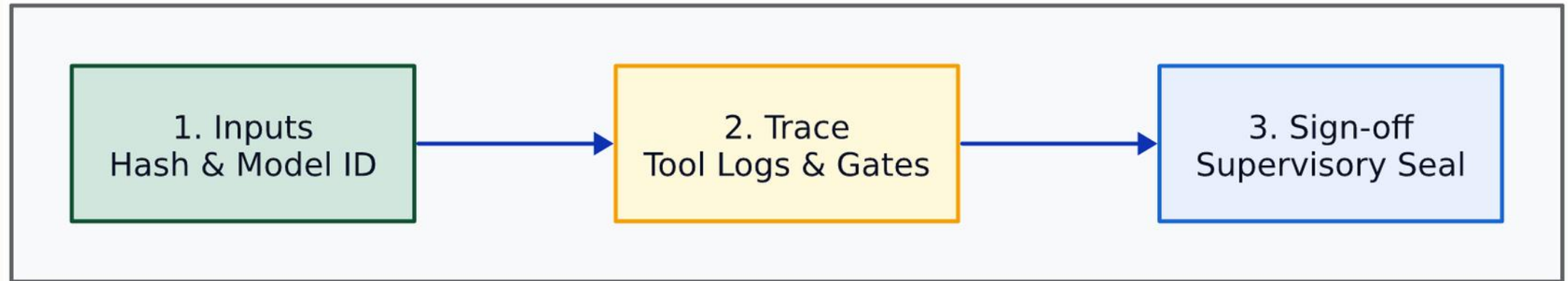
- Single prompts asking for extraction, math, risk, and drafting collapse under load.
- Attention scatters across conflicting instructions, raising error rates.
- Splitting tasks across specialized roles mirrors professional organizational design.
- Each role focuses on one job without distraction.
- Multi-session pipelines achieve higher precision than all-in-one prompts.

Four-Session Pipeline: From Data to Memo

- Session 1: The Extractor parses primary filings and pulls raw numbers with citations.
- Session 2: The Calculator ingests extracted numbers and runs exact Python scripts.
- Session 3: The Reviewer audits outputs against source files using the Four Checks.
- Session 4: The Editor shapes verified figures into an executive investment briefing.
- No single session has permission to skip intermediate verification gates.

Defensive Systems Require Audit Dossiers

Audit Dossier



- Regulated institutions must prove analytical lineage to supervisory authorities.
- Re-running prompts years later cannot reproduce the same text because generation is stochastic.
- Defensibility requires saving an immutable audit dossier when the memo is published.
- The dossier packages frozen input files, execution logs, and supervisory signatures.

Four Elements of an Audit Dossier

- Element 1: Input Hash Manifest saving SHA-256 signatures of files, prompts, and models.
- Element 2: Execution Trace recording external tool payloads, Python code, and output logs.
- Element 3: Reconciliation Table linking every stated metric to exact document coordinates.
- Element 4: Supervisory Attestation capturing the timestamped signature of the human analyst.
- This four-part package establishes unbroken chain-of-custody for every financial figure.

Human Reviewer Retains Legal Liability

- Artificial intelligence algorithms cannot sign contracts or hold fiduciary duties.
- Securities law places legal liability squarely on corporate officers and analysts.
- An analyst cannot defend an inaccurate filing by blaming automated software.
- AI speeds up drafting; human professionals verify and take responsibility.
- Professional judgment and supervisory sign-off are legal obligations, not formalities.

Lab 5: Two-Session Extraction and Cross-Check

- Time budget: 8 minutes hands-on in Goose.
- Open two separate chat sessions in your Goose workspace.
- In Session 1, run the Extractor prompt below.
- Copy the structured JSON output into Session 2 and run the Reviewer prompt.
- Have the Reviewer verify citations and check whether leverage violates the 3.50x ceiling.

Session 1 (Extractor):

Document: [Excerpt from Acme Industrial Technologies 10-K, Note 8:

Under the Senior Credit Facility, consolidated net funded debt to EBITDA shall not exceed 3.50x. Consolidated funded debt stood at \$2,800 million against cash of \$360 million. Trailing EBITDA reached \$1,220 million. Consolidated fourth-quarter revenue totaled \$4,280 million.]

Instructions:

Extract the metrics into a strict JSON object: company, revenue_millions, funded_debt_millions, cash_millions, trailing_ebitda_millions, covenant_ceiling.

Session 2 (Reviewer):

Audit the extracted JSON payload against Note 8.

Confirm whether net debt (\$2,440M) divided by EBITDA (\$1,220M) satisfies the 3.50x covenant.

Output an itemized verification log with PASS or FAIL status.

Executive Rule: Automated Reports Need Lineage

- Require audit dossiers for all publications that use artificial intelligence.
- Store prompt records, model version identifiers, and tool logs in compliance archives.
- Require dual-analyst sign-off on AI work that informs investment committee decisions.
- Train compliance officers to inspect code execution logs and reconciliation tables.
- Build audit defensibility into your software workflows before deployment.

Morning Conclusion: Questions and Working Lunch

- **SCHEDULE : LUNCH BREAK: 12:30 PM TO 1:15 PM (45 MINUTES)**
- **Target Clock** : Concludes at **12:30 PM** .
- Break for lunch and step away from your screens.
- Next block: Afternoon Creative Session starts promptly at 1:15 PM.
- Morning complete: From token logits to multi-session pipelines and audit dossiers.
- Questions or discussions on institutional governance and model risk before lunch?

SECTION 6: AFTERNOON

Creative Session: Investment Committee

1:15 to 4:00 | Hands-On Challenge, Briefing Deliverables, and Synthesis

The Afternoon Mission: Investment Briefing

- Work in teams of two to three analysts in the BMO Finance Lab.
- Perform a thorough financial review using Goose, OpenRouter, and company filings.
- Build a complete two-page investment committee briefing on Acme Industrial Technologies.
- Apply the tools, file boundaries, and multi-session checks learned this morning.
- Produce an unbroken audit dossier proving the exact lineage of every number.

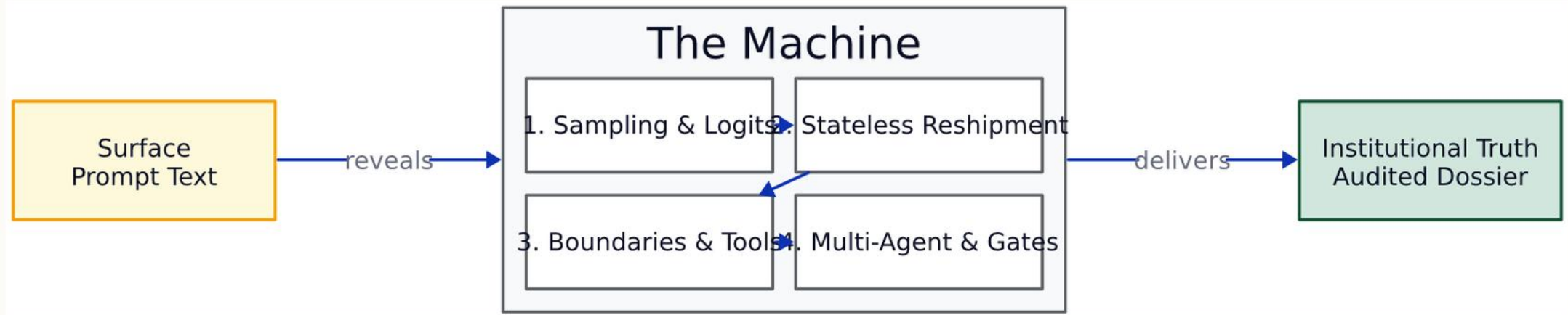
Three Project Tracks for Analysis

- Track A: Financial Health & Debt Covenant Analysis (Maturities, Python interest coverage, covenants).
- Track B: Capital Structure & Net Leverage Audit (Cash reconciliation, debt ratios, covenant headroom).
- Track C: Executive Commentary & Disclosure Reality Check (CEO letter tone vs segment financials).
- Select one track per team and configure your Goose workspace to begin analysis.
- Every track requires deterministic code calculations and verified document citations.

Deliverable Requirements and Rubric

- Requirement 1: Executive Investment Briefing summarizing findings, risks, and conclusions.
- Requirement 2: Reconciliation Table linking every numerical claim to exact filing lines.
- Requirement 3: Tool Execution Trace showing deterministic Python calculator logs.
- Requirement 4: Signed Supervisory Attestation certifying that all numbers were independently verified.
- Upload completed deliverables to the shared lab folder before the final share-out.

The Re-Examined Prompt



- This morning, a prompt looked like simple text typed into a chat window.
- We now recognize the prompt as the outer layer of a multi-part mechanical system.
- It drives external tool executions, multi-session pipelines, and audit dossiers.
- Defensible finance replaces superficial prompt tips with disciplined systems engineering.

Writing Your LinkedIn Project Summary

- Use your Goose agent to draft a professional summary of your afternoon project.
- Highlight your hands-on mastery of agentic workflows, deterministic tools, and audit controls.
- Explain the difference between consumer chatbots and governed enterprise systems.
- Emphasize your ability to audit financial models, stop hallucinations, and defend source lineage.
- Frame your AI skills around governance, reliability, and institutional risk management.

Draft a concise, professional LinkedIn post describing my work today at Schulich:

- Built an auditable financial analysis pipeline on corporate 10-K filings.
- Replaced unguided chatbot arithmetic with deterministic Python execution via MCP.
- Implemented multi-session division of labor (Extractor/Reviewer) and audit dossiers.
- Aligned pipeline controls with institutional model risk management principles.
- Focused on institutional defensibility and eliminating compounding error in finance.

Keep the tone professional, analytical, and free of marketing buzzwords.

Five Invariants for Production Financial AI

- 1: Language models estimate token odds; they never perform authoritative calculations.
- 2: Conversation state lives in application memory; start fresh sessions to control cost and drift.
- 3: Context boundaries must remain frozen; isolate primary filings within immutable project perimeters.
- 4: Math requires external runtimes; delegate arithmetic to deterministic Python engines.
- 5: Accountability never transfers to software; require cryptographic audit dossiers and human sign-off.

Conclusion: Defensible Financial Analysis

- You now hold the mental models to evaluate artificial intelligence with complete clarity.
- You understand how models generate text, why they fail, and how tools give them exact hands.
- You can design multi-session workflows that stop context decay and compounding errors.
- You carry practical frameworks to cut through vendor hype and evaluate software purchases.
- Apply these principles to build financial systems that survive scrutiny and protect your firm.