Rearchitecting Enterprises from Apps to Agents

Why agentic AI is becoming the new surface, what enterprise foundations it needs, and how to make your solutions agent ready.

Ahmad Najjar

Managing Solution Architect · Capgemini · Norway

For the architects and tech leads who build what runs underneath

Ahmad Najjar

Managing Solution Architect · Capgemini
Oslo, Norway

EXPERIENCE & FOCUS AREAS

22+ **Microsoft Ecosystem**
Development, Databases, Collaboration, Documents

11+ **Azure**
Integration, Service Architecture, Functions

9+ **Power Platform**
Power Automate, Power Apps, AI Builder, Copilot Studio

5+ **AI**
Applied intelligence, Agentic AI, Architecture, Tools

Business Applications MVP · 10 Years
Microsoft Certified Trainer · 4 Years
FTRSA – Power Apps & Power Automate
Books Author & Reviewer

📘 **Microsoft Power Automate Cookbook**
O'Reilly · Author

COMMUNITY & SPEAKING

200+	50+	8+	10+
Sessions	Workshops	Events	Community

Enthusiastic Solution Architect by day · Passionate Developer at heart ☺

// HOW THE HOUR RUNS

Five acts, one question

What is an app when the agent becomes the surface?

01	**The shift**	From prediction engines to digital actors, and what an app becomes
02	**Agentic logic endpoints**	Agents as the new integration logic layer
03	**Information Architecture**	Knowledge models, context, contracts, retrieval, lineage, security
04	**Intent & agent readiness**	Designing around intent and making solutions agent ready
05	**Trust to scale**	Evals, human in the loop, attack surface, cost and latency

The shift

From prediction engines to digital actors. What happens to the word "app" when software starts to act?

// SETTING THE SCENE

Uncharted territory

01

No playbook

There are no concrete guidelines yet — the patterns are still being written, by us, in real time.

02

Mostly noise

Endless speculation and anticipation online. Some of it sensible; much of it not.

03

A genuine shift

Not just adopting a tool — a fundamental change in how we work every single day.

Uncharted territory means the playbook is being written now — separating signal from noise while the shift unfolds.

A new era of work! This is not technology adoption. It is a change in how we work, and it is unlike any shift before it...

// THE SHIFT

From prediction engines to digital actors

Three generations of AI in the enterprise, and where the human sits in each.

01 HUMAN ACTS

Predict

Classify, score, forecast

- Fraud scores, churn models, demand forecasts
- Output is a number or a label
- A human reads it and acts
- Deterministic pipelines around a model

02 HUMAN ACTS

Generate

Draft, summarise, answer

- Chat, copilots, RAG over documents
- Output is text or an artifact
- A human reviews it and acts
- The model is a function call

03 HUMAN SUPERVISES

Act

Plan, call tools, orchestrate

- Agents loop: perceive, reason, act, observe
- Output is an outcome in your systems
- A human sets goals and supervises
- The model is a runtime

The unit of value moved from an answer to an outcome. That is what forces the architecture to change.

// THE SHIFT

Every era redefined the app

The interface has been shrinking for thirty years. Agents take it to its logical end.

CLIENT / SERVER	WEB	MOBILE & CLOUD	SAAS & LOW CODE	AGENTIC
The app is a binary	**The app is a URL**	**The app is an API plus a screen**	**The app is a configured platform**	**The app is a capability an agent can reason with**
Installed, versioned, one screen per task	Pages and forms, the browser is the runtime	Backends for frontends, REST everywhere	Connectors, flows, canvases on shared data	Tools, knowledge, policy. The screen is optional.

Each era removed a layer of predefined interface. Agents remove the last one: the interface is negotiated at runtime.

Same lesson as on prem to cloud: the playbook is written by the people who start building while it is still uncertain.

// WHAT AN APP BECOMES

What an app becomes

Apps do not disappear. They lose the monopoly on being the surface.

	Traditional app	AI first solution
COMPOSITION	UI + logic + data, shipped together	Capabilities + knowledge + policy, exposed as contracts
INTERFACE	Predefined screens, forms and flows	Negotiated at runtime; screens optional or generated
WHO DRIVES	A human, clicking through a path we designed	An agent acting on an intent, with a human supervising
INTEGRATION	Point to point, mapped by developers	Reasoned over by agents, executed by workflows
CHANGE	A release	A new skill, a new tool, a new contract

An AI first solution is a set of capabilities, a body of governed knowledge, and a policy for how they may be used.

// A BALANCED VIEW

Where agents win, where apps hold

The honest answer to "are apps being replaced?" is: the surface is. The system of record is not.

AGENTIC ORCHESTRATION WINS WHEN

The work is open ended

- Multi step reasoning with judgement calls
- Tasks that span many systems and teams
- Ambiguous, conversational or changing intent
- Exceptions outnumber the happy path
- The cost of a wrong answer is recoverable

STRUCTURED APPLICATIONS HOLD WHEN

You need determinism

- High volume, repeatable transactions
- Hard transactional and audit guarantees
- Tight latency and cost ceilings
- Regulated steps that must be identical every time
- The system of record itself

A mature architecture uses both, deliberately. The mistake is forcing an agent into a job a form does better, or the reverse.

Agentic logic endpoints

From service endpoints with fixed interfaces to endpoints that reason, act and orchestrate. Agents as the new integration logic layer.

// DEFINITION

Service endpoint vs agentic logic endpoint

Same client server shape, fundamentally different contract.

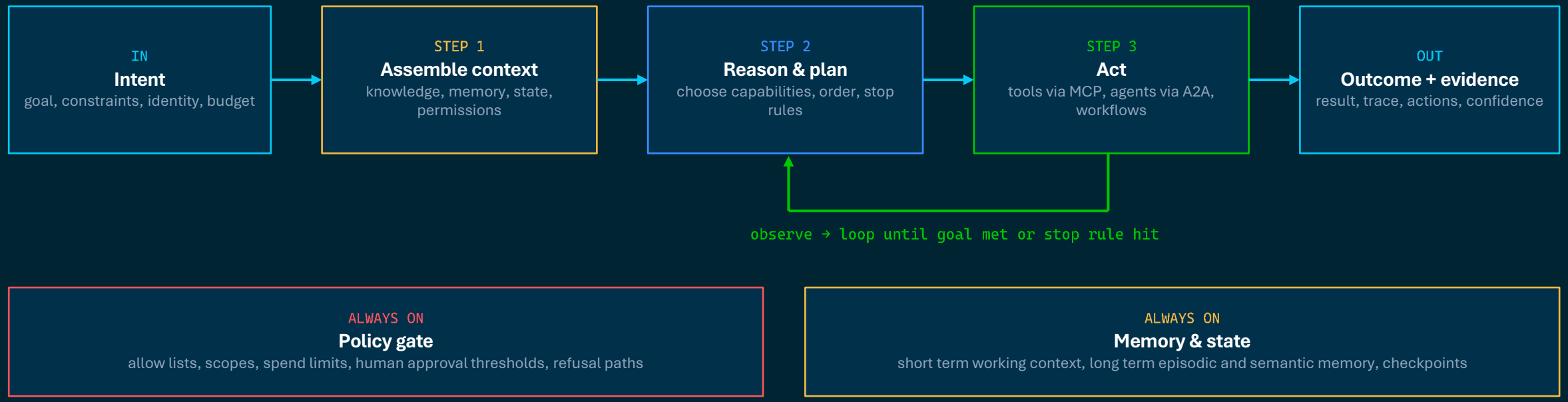
Dimension	Service endpoint	Agentic logic endpoint
Contract	Fixed schema, versioned, documented in OpenAPI	Intent plus constraints in; outcome plus evidence out
Behaviour	Deterministic: same input, same output	Reasoned: non deterministic within declared bounds
Discovery	Docs read by developers at design time	Descriptions and cards read by models at runtime
Composition	A developer wires the calls in code	The agent plans the calls, per request
Testing	Unit and integration tests, pass or fail	Evals, rubrics, red teaming, scored
Failure	Error codes and retries	Degraded outcome, escalation, human review
Change	A new version and a migration	A new tool, skill or policy; behaviour shifts

An agentic endpoint accepts a goal and returns an outcome with evidence. Everything else in this session is what makes that reliable.

// ANATOMY

Anatomy of an agentic logic endpoint

What has to exist between "intent in" and "outcome out".



Evidence is not optional. Every outcome carries the context used, the plan chosen, the tools called and who approved what.

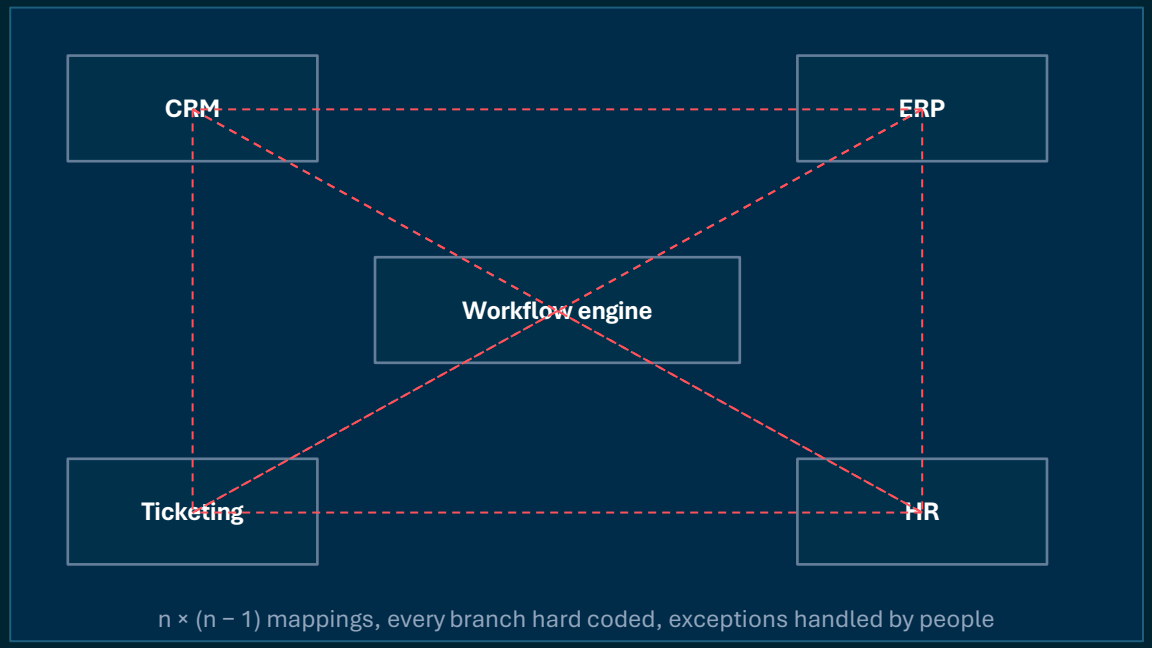
Intent → context → plan → act → outcome, inside a policy gate, with memory. That loop is the new "request handler".

// THE INTEGRATION LAYER

Agents as the integration logic layer

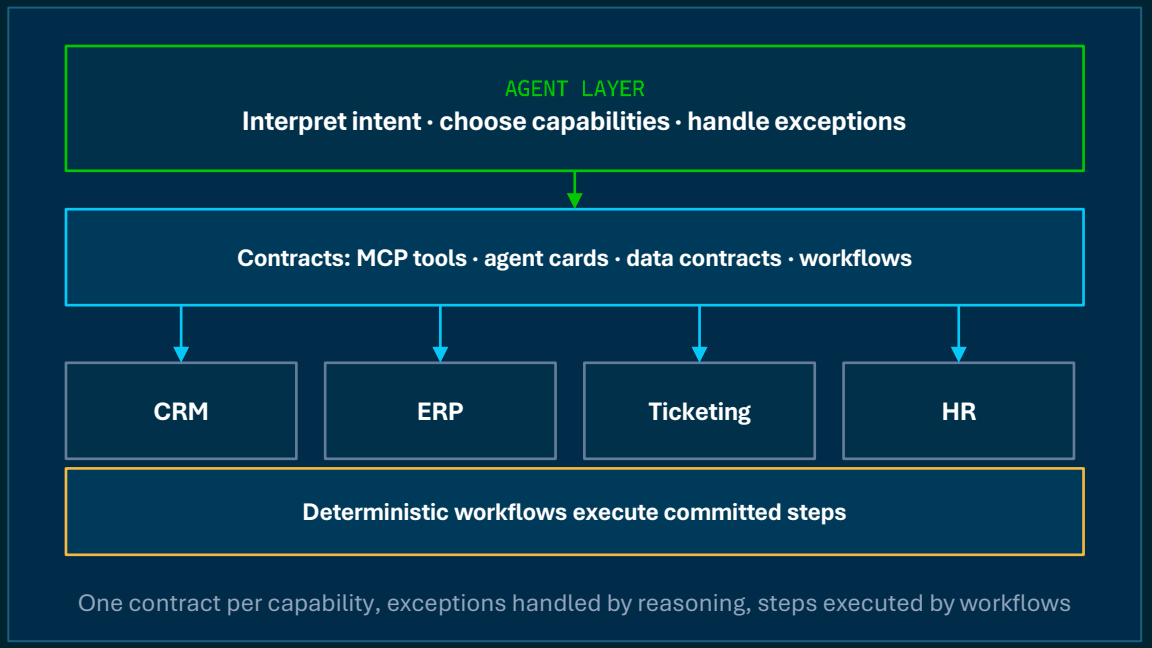
Integration logic moves from mapping code to reasoning over contracts.

BEFORE — LOGIC LIVES IN THE WIRING



n × (n − 1) mappings, every branch hard coded, exceptions handled by people

AFTER — LOGIC LIVES IN THE AGENT LAYER



One contract per capability, exceptions handled by reasoning, steps executed by workflows

The agent does not replace the integration. It replaces the part of the integration that encoded human judgement in code.

// THE DESIGN RULE

Deterministic core, agentic edge

The agent decides. The workflow executes. The system of record commits.

AGENTIC

Where judgement lives

- Interpret and classify intent
- Plan and choose capabilities
- Negotiate with other agents
- Handle exceptions and ambiguity
- Draft, summarise, recommend

DETERMINISTIC

Where commitment lives

- Validate against business rules
- Commit, post, pay, notify
- Enforce transactional guarantees
- Run high volume, low latency steps
- Produce the audit record

SHARED

What both depend on

- Contracts for every capability
- One identity model, propagated
- Policy: scopes, limits, approvals
- Observability and lineage
- Evals and regression guards

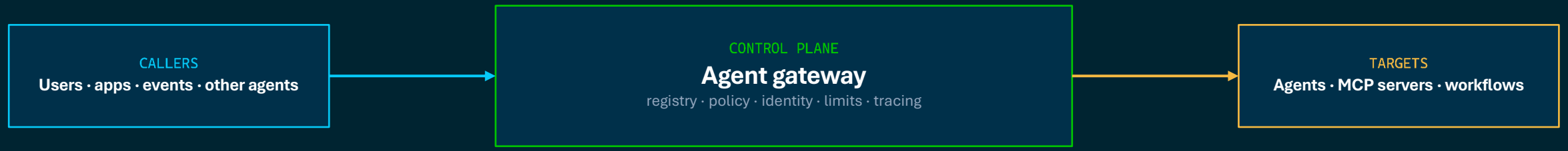
Never let an agent be the system of record. Let it be the system of judgement.

Practical test: if a step must be identical every time and auditable, it is deterministic. If it needs judgement, it is agentic. If you cannot tell, it is deterministic until evals prove otherwise.

// THE AGENT GATEWAY

The agent gateway: an API gateway for reasoning

When agents are the integration layer, they need the same control plane APIs got fifteen years ago.



REGISTRY & DISCOVERY

Every agent and tool known, described, versioned, owned

IDENTITY & POLICY

Agent identity, on behalf of flows, scopes, allow lists, approval rules

LIMITS

Token and spend budgets, rate limits, loop and depth guards

TRACING

Every intent, plan, tool call and outcome correlated end to end

VERSIONING

Prompts, skills, tool schemas and models as versioned artifacts

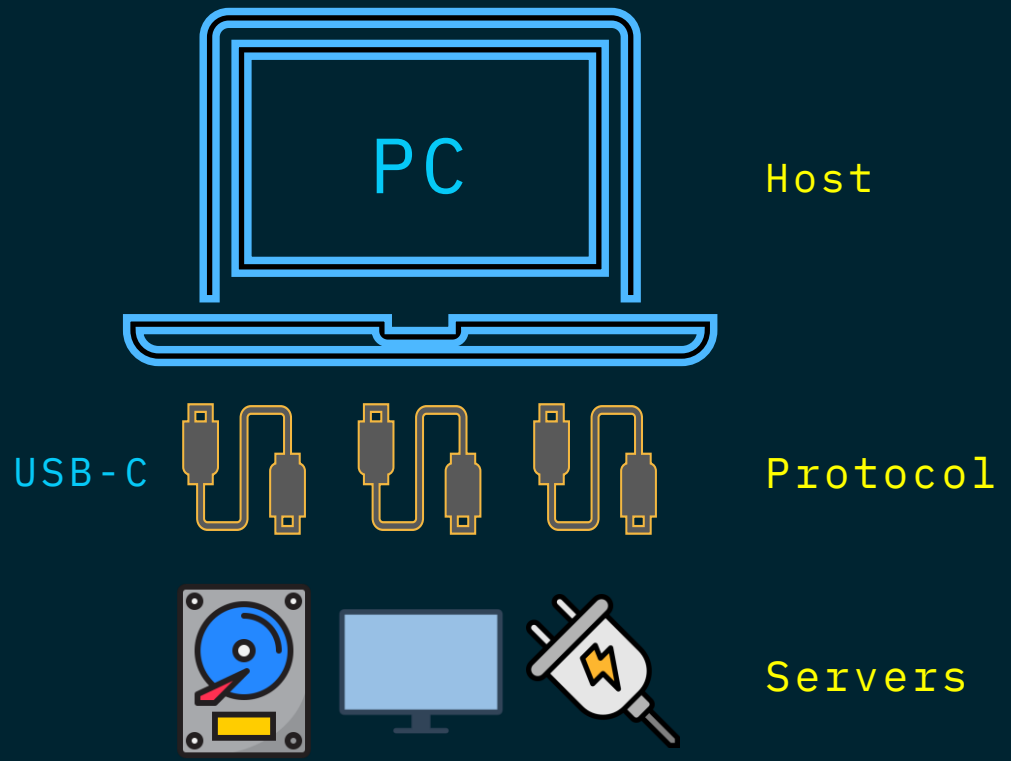
QUARANTINE

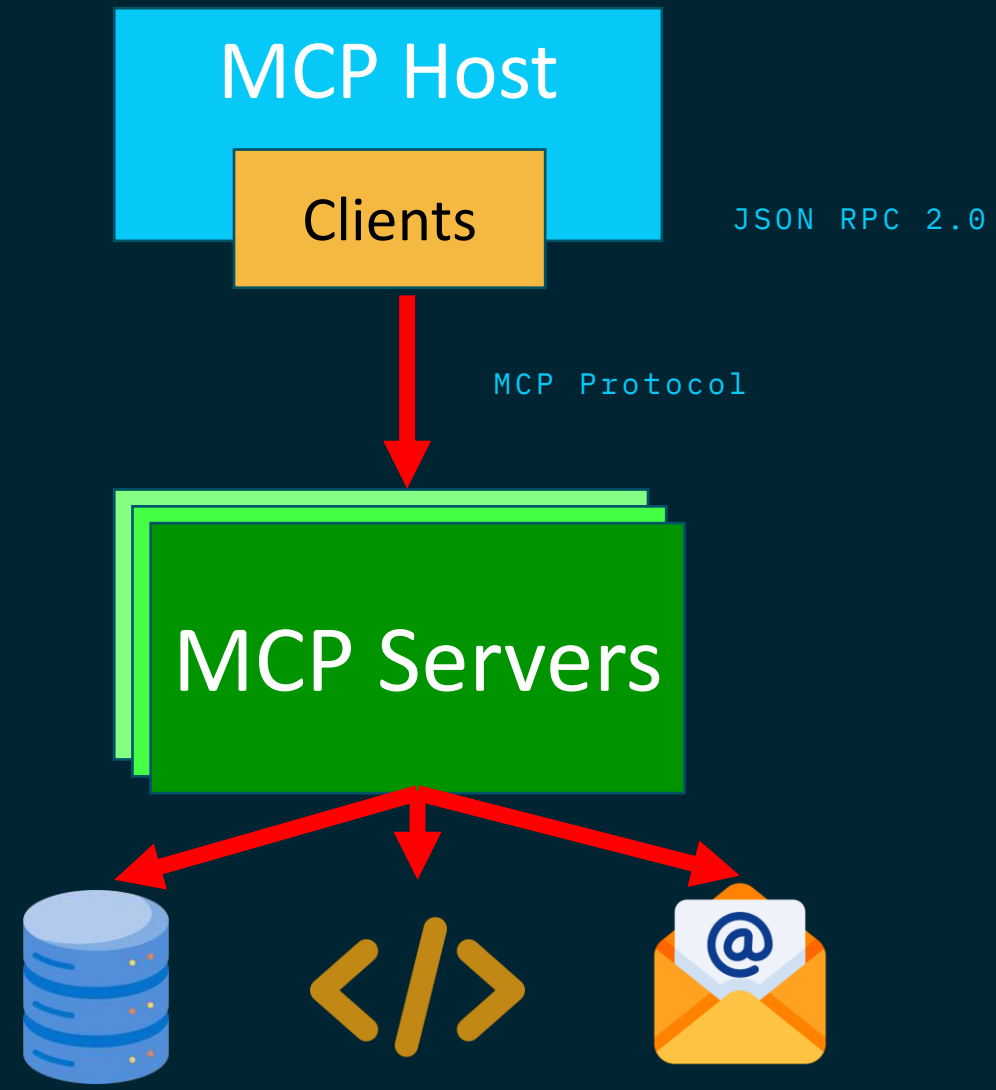
Kill switch and isolation for a misbehaving agent or server

Microsoft example: Agent 365 registry with Entra Agent ID for identity and inventory; API Management fronting MCP servers for policy, limits and tracing.

// THE AGENT GATEWAY

The MCP Anatomy

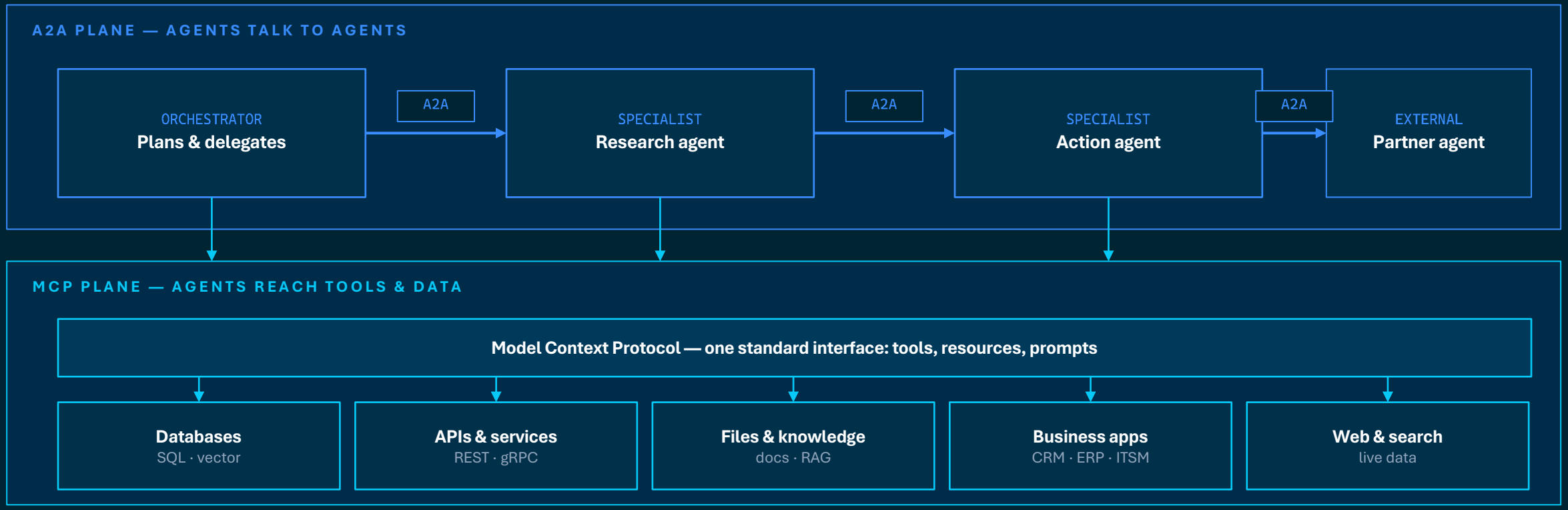




// PROTOCOLS

Two protocols, one backbone

A2A connects agents to agents. MCP connects agents to tools and data. Together they replace bespoke integration logic.

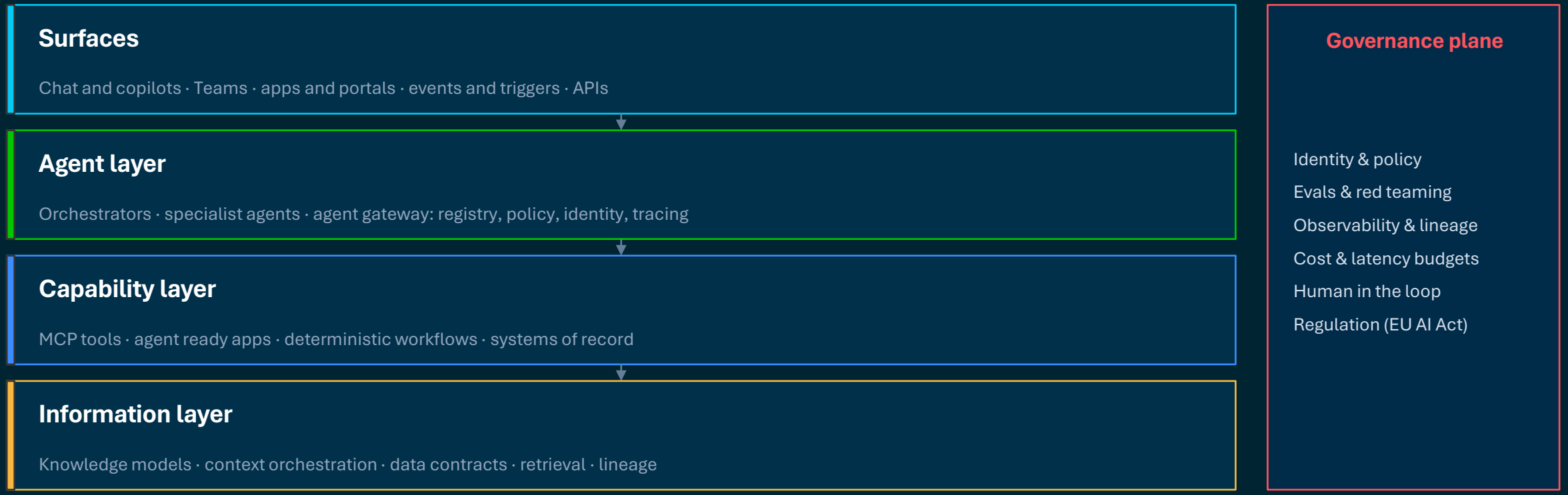


Rule of thumb: A2A for agents talking to agents, MCP for agents talking to tools and data. Bespoke connectors become the exception.

// THE MAP FOR THE REST OF THE SESSION

The reference architecture

Five layers, one governance plane. The map for the rest of the session.



Most enterprises have invested in the top two layers and are discovering the bottom two are where agents actually fail.

Information Architecture

The foundation agents reason over. Structured, governed, high quality information sources and flows, or unpredictable agents.

// WHY NOW

IA just got promoted to the front line

For years information architecture was a back office discipline. Agents moved it into the request path.

ABOVE THE WATERLINE

AI strategy · agents · copilots · demos

Legacy systems
models nobody can explain, interfaces nobody documented

Data pipelines
inconsistent, undocumented, well meaning

Integration debt
point to point connections and fragile handoffs

Undocumented meaning
the same word means three things in three systems

Leadership sees the tip. Architects see what is underneath.

THE OLD FAILURE MODE

Bad IA gave you a slow report

Someone noticed, someone fixed the query, the business carried on.

THE NEW FAILURE MODE

Bad IA gives you a confident, fluent, hallucinating agent

It acts on the wrong definition of "customer" at 2 a.m. and nobody notices until the audit.

THE QUIET PART

An agent cannot find answers out of thin air, unless it is hallucinating. The systems and the data have to exist first, and now they have to be built agent ready.

// SIX PILLARS

Six pillars of agent grade information architecture

Every one of these is a design decision you own. Skip one and it finds you in production.

PILLAR 01

Knowledge models

Shared definitions, entities and relationships agents reason over

PILLAR 02

Context orchestration

How context is assembled per request, with what budget and priority

PILLAR 03

Data contracts

Owned, versioned promises about schema, meaning, quality and access

PILLAR 04

Retrieval design

How the right facts surface fast, precisely, and permission trimmed

PILLAR 05

Lineage & provenance

Where data came from and what an agent did with it, replayable

PILLAR 06

Security

Identity propagation, classification, isolation of untrusted content

Without these, agentic behaviour becomes unpredictable, unscalable, or misaligned with business outcomes.

// PILLAR 01 · KNOWLEDGE MODELS

Agents reason over meaning, not tables

If Sales, Service and Finance disagree on what a "customer" is, so will your agents.

Glossary

One definition per business term, with an owner. "Active customer", "open case", "net revenue".

Entity model / ontology

Customer, order, case, asset, and how they relate. The map an agent navigates when it plans.

Semantic layer

Metrics, measures and business rules defined once and reused by every agent and report.

Knowledge graph (optional)

Relationships across systems for multi hop questions: "which suppliers affect this order?"

WHY IT MATTERS FOR AGENTS

- Tool descriptions and prompts inherit the glossary
- The entity model bounds what an agent may plan over
- Ambiguity in the model becomes divergence in behaviour

MICROSOFT EXAMPLES

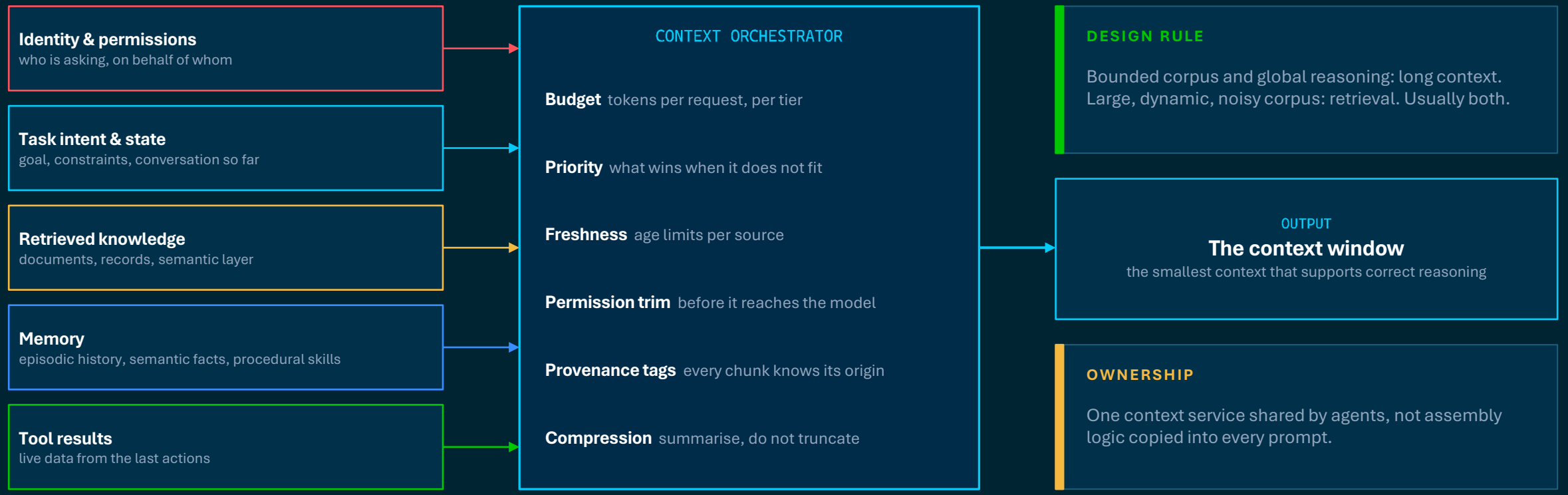
- Dataverse tables and relationships as the entity model
- Fabric and Power BI semantic models as the metric layer
- Purview business glossary for owned definitions

The semantic layer is the thing agents should reason over. Raw tables are for the deterministic core.

// PILLAR 02 · CONTEXT ORCHESTRATION

Context is assembled, not found

Every request gets a context built to a budget from many sources. Someone has to own that assembly logic.



Bigger is not always better. Precision retrieval beats raw capacity, and attention dilutes long before the window fills.

// PILLAR 03 · DATA CONTRACTS

Contracts are what agents read

A data contract is an owned, versioned promise. Agents cannot ask a colleague what a column really means.

OWNER

a named team accountable for the dataset or capability

SCHEMA

fields, types, keys, required and optional

SEMANTICS

what each field means, in glossary terms, with examples

QUALITY SLOS

freshness, completeness, accuracy, and what happens on breach

ACCESS POLICY

who may read or write, classification, retention

CHANGE PROCESS

versioning, deprecation windows, consumer notification

DESCRIPTIONS ARE ROUTING LOGIC

Not documentation. Routing logic.

- A tool description tells the model when and how to call it. Vague description, wrong call.
- A schema without semantics is a guess an agent will make for you.
- Side effects must be declared: read only, idempotent, or mutating.
- Consumers of a contract include agents you have not built yet.

THE SAME DISCIPLINE, THREE SHAPES

Data contract
a dataset or a semantic model

Tool contract
an MCP tool: name, description, input and output schema, side effects

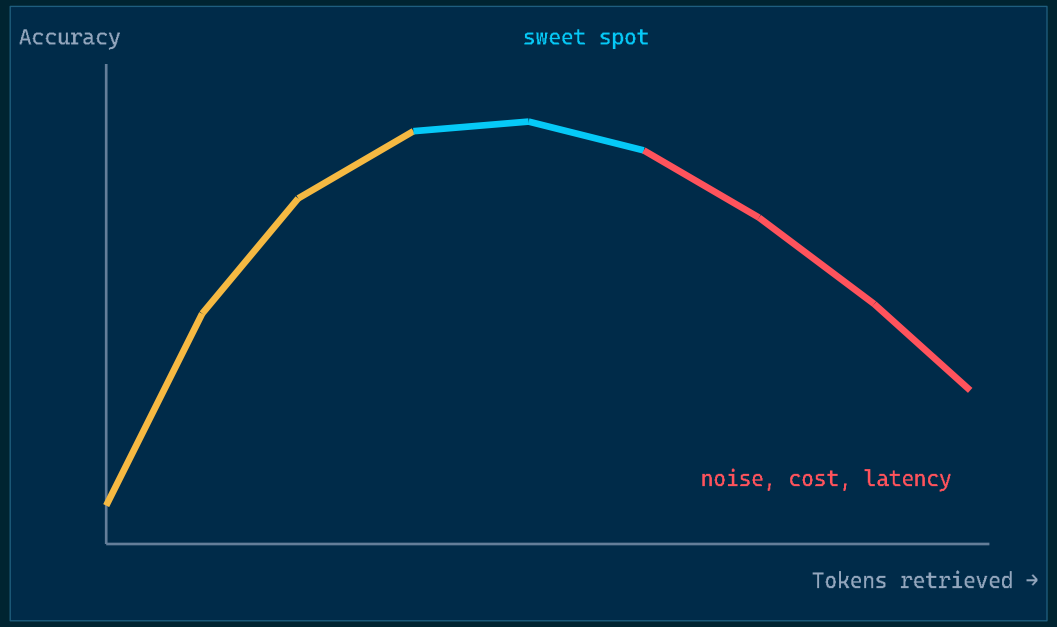
Agent contract
an A2A agent card: skills, modalities, auth, limits

Every contract you skip becomes a guess an agent makes on your behalf, in production, with your identity.

// PILLAR 04 · RETRIEVAL DESIGN

Retrieval: precision over volume

Not everything should be dumped into the context window. Intentionality matters.



Chunking chunk size sets the precision ceiling; too big is noisy, too small breaks meaning

Hybrid search + rerank keyword plus semantic, then a reranker; beats either alone

Metadata filters entity, date, source, classification, before similarity

Permission trimming at retrieval time, with the caller's identity, never after

Retrieval evals measure recall and precision per corpus; treat drift as a defect

Bounded corpus, global reasoning	Long context
Enterprise scale, dynamic, noisy	RAG
Cross document gap analysis	Long context
High precision search	RAG

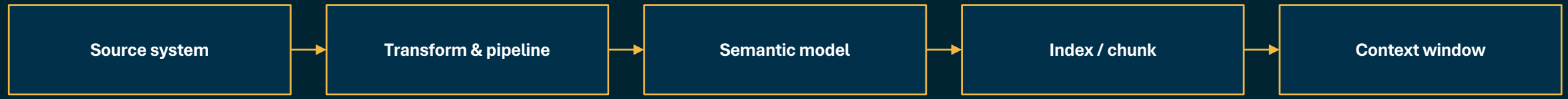
Retrieval selects the right subset; long context gives the model room to reason over it. Design both, measure both.

// PILLAR 05 · LINEAGE & PROVENANCE

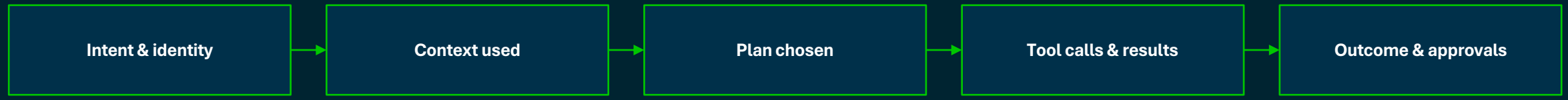
Two chains you must be able to replay

Data lineage tells you where a fact came from. Action lineage tells you what an agent did with it.

DATA LINEAGE — THE CHAIN INTO THE CONTEXT



ACTION LINEAGE — THE CHAIN OUT OF THE AGENT



DEBUGGING

Why did it do that? Without both chains, every incident is an argument with a probability distribution.

COMPLIANCE

The EU AI Act expects traceability and human oversight for high risk use. Lineage is how you prove it.

TRUST

An outcome with evidence can be reviewed, sampled and scored. An outcome without it can only be believed.

Microsoft examples: Purview for data lineage and classification; Foundry tracing and Copilot Studio analytics for action traces; Agent 365 for cross agent visibility.

// PILLAR 06 · SECURITY

Security lives in the information layer too

Permission trim at retrieval, not at answer. Labels follow outputs. Untrusted content never becomes instructions.

IDENTITY PROPAGATION

Agents act on behalf of a person or a service identity. Every retrieval and every tool call carries that identity. No shared super user.

TRIM BEFORE THE MODEL

Filter by permission at retrieval and tool time. Asking the model to "not mention" what the user cannot see is not a control.

LABELS FOLLOW OUTPUTS

Sensitivity classification travels from source to chunk to context to answer to downstream action.

ISOLATE UNTRUSTED CONTENT

Documents, emails and web pages are data, never instructions. Structural separation, not a polite prompt.

SECRETS OUT OF PROMPTS

Credentials live in a vault and are injected at the tool boundary. The model never sees them.

DLP ON ACTIONS

The same data loss prevention you apply to people applies to what agents send, post and export.

Microsoft examples: Entra for identity and on behalf of flows; Purview sensitivity labels and DLP that apply to Copilot Studio agents and AI outputs.

// THE INFORMATION LAYER, END TO END

The information layer, end to end

Vendor neutral pattern, Microsoft products as the worked example.

	PATTERN	MICROSOFT EXAMPLE
Consumers	Agents · copilots · apps · reports	M365 Copilot · Copilot Studio · Foundry agents · Power BI
Context & retrieval	Context orchestrator · retrieval services · memory stores	Azure AI Search · Foundry · Dataverse knowledge · Graph
Semantic & knowledge	Glossary · entity model · semantic layer · knowledge graph	Purview glossary · Dataverse · Fabric semantic models
Governed storage	Lakehouse · operational stores · document stores	OneLake · Fabric · Dataverse · SharePoint
Ingestion & contracts	Pipelines · data contracts · quality SLOs	Fabric pipelines · Dataflows · Purview data quality
Sources	Operational databases · SaaS applications · files & documents · events	Dynamics 365 · SQL · SAP · M365 · third party

Security & governance

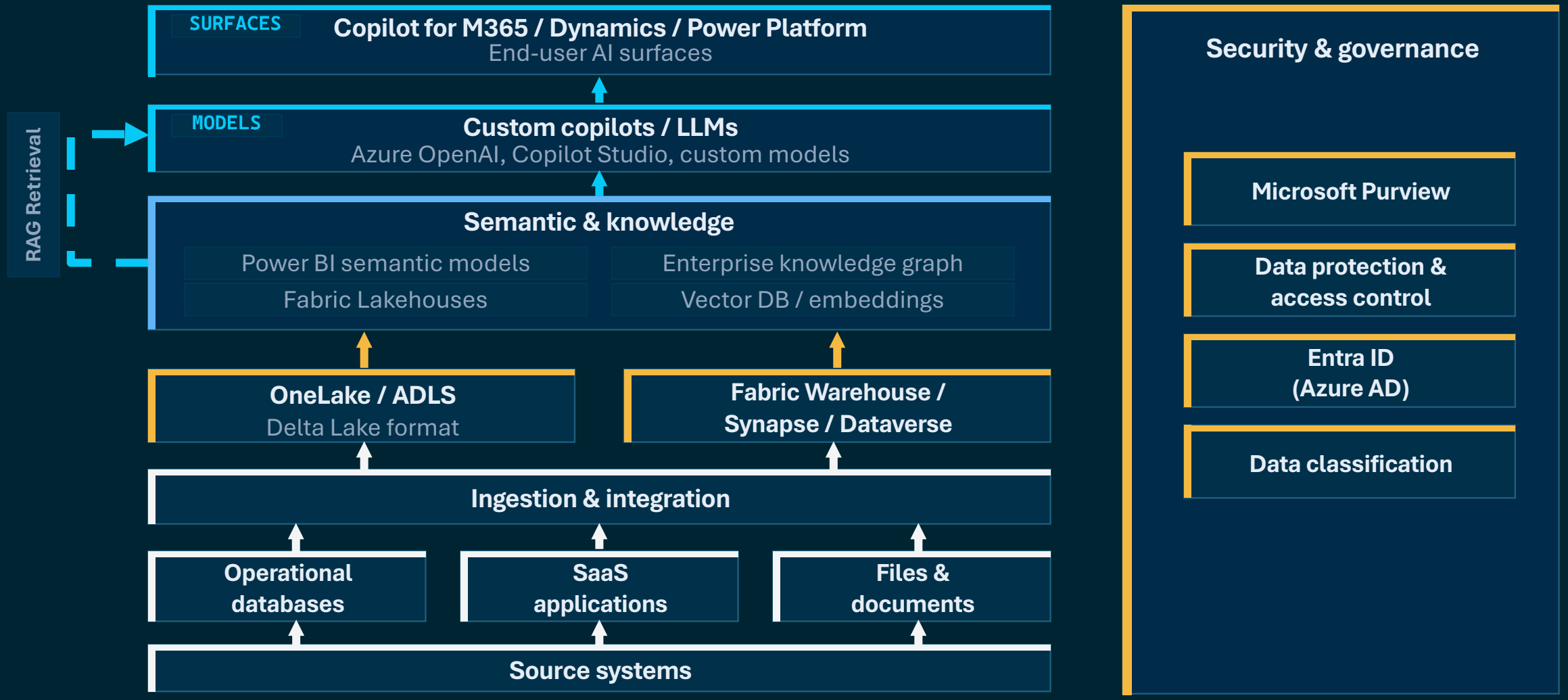
Identity & access
Classification & labels
DLP & retention
Lineage & catalogue
Audit & monitoring

Entra · Purview · Defender

The layers turn scattered infrastructure into reusable, governed capability. Agents plug in at the top; contracts hold at every boundary.

// AN EXAMPLE INFORMATION ARCHITECTURE

An example information architecture



Designing around intent

AI first solutions, and a way to make your existing and new systems agent ready before the agents arrive.

// INTENT

Intent is the new interface

We used to design paths. Now we design capabilities and let the path be negotiated.

INTERFACE BY DESIGN — A PATH

Every step, field and branch decided up front. Change the process, change the screens.

INTERFACE BY NEGOTIATION — AN INTENT

"Onboard this supplier under the standard terms by Friday." The path is composed at runtime from what the systems expose.

DESIGN CAPABILITIES, NOT SCREENS

Each business action becomes a named, described, contracted capability. The screen is one client among many.

UX FOR PEOPLE, AX FOR AGENTS

Agent experience: descriptions, schemas, error messages and docs written for a model to read at runtime.

GENERATIVE AND OPTIONAL UI

The interface can be rendered on demand for the step that needs a human: an approval, a choice, a correction.

INTENT NEEDS BOUNDARIES

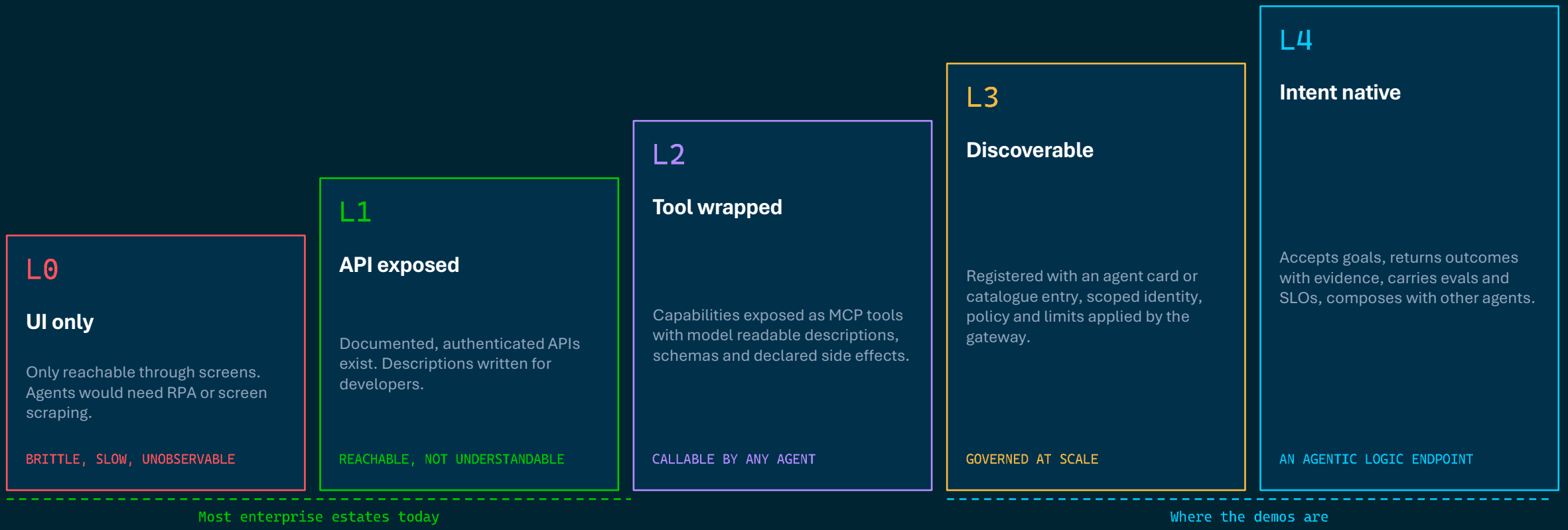
Constraints, budgets and stop rules are part of the request. "Do it" without limits is not an intent, it is a liability.

An AI first solution exposes what it can do and what it knows, and lets the intent decide how they combine.

// THE AGENT READINESS LADDER

The agent readiness ladder

Score every system you own. Most enterprises sit at level 1 and think they are at level 3.



Moving a system from L1 to L2 is a few weeks of contract work. It unlocks every agent you will ever build against it.

// AGENT READY, IN PRACTICE

Making an existing solution agent ready

Wrap, describe, harden, register. Then let the strangler pattern do its work.

01 EXPOSE CAPABILITIES, NOT SCREENS

Name the business actions the app performs and expose each as a tool. "Create supplier", not "open the supplier form".

02 DESCRIPTIONS AS ROUTING LOGIC

When to use it, when not to, what it needs, what it returns. Use glossary terms. Include examples.

03 DECLARE AND HARDEN SIDE EFFECTS

Read only, idempotent or mutating. Idempotency keys on writes, compensating actions for the rest.

04 RETURN OUTCOMES WITH EVIDENCE

Structured results, record identifiers, what changed, what did not, and why. Errors a model can act on.

05 PROPAGATE IDENTITY

On behalf of flows, scoped tokens, no service account with god mode. The tool enforces the caller's permissions.

06 REGISTER AND EVALUATE

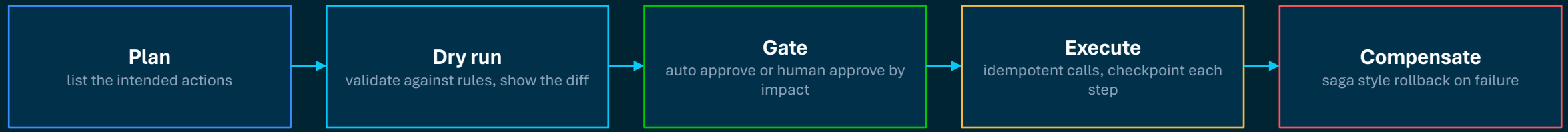
Publish the contract to the registry, attach an owner, add evals for the top intents, monitor the calls.

Strangler pattern for agents: wrap the legacy capability as a tool today, replace the implementation behind the contract tomorrow.

// WHEN AGENTS MUTATE STATE

Transactional integrity when agents act

A non deterministic planner calling mutating tools is a distributed transaction with a creative coordinator.



IDEMPOTENCY KEYS

Every mutating tool accepts a key. A retried or repeated call is safe by construction, because agents retry.

COMPENSATING ACTIONS

Every write has an undo, and the undo is a tool too. Cancel order, reverse posting, revoke access.

IMPACT TIERS

Classify actions: reversible, costly, irreversible. Tier decides the gate: auto, review after, approve before.

CHECKPOINTS & BLAST RADIUS

Persist state after each risky step; isolate agent lanes so one bad plan cannot cascade across systems.

The agent may decide. The workflow executes. The system of record commits. Each layer keeps its own guarantees.

// AI FIRST SOLUTIONS

Anatomy of an AI first solution

Capabilities, knowledge, procedure and policy, with surfaces on top. Design each part explicitly.

WHAT IT CAN DO

Capabilities

Tools with contracts, deterministic workflows for committed steps, agents it can delegate to.

WHAT IT KNOWS

Knowledge

Semantic layer, retrieval over governed sources, episodic memory of past interactions.

HOW IT DOES THINGS

Procedure

Skills: portable procedural knowledge, versioned and reviewed like code. The "how to" layer.

WHAT IT MAY DO

Policy

Scopes, spend limits, approval thresholds, refusal paths, evals that gate release.

Surfaces: chat · Teams · a generated screen for the human step · an event trigger · an API for other agents

MEMORY MAPS ONTO A FAMILIAR HUMAN SPLIT

Semantic — facts
knowledge bases, RAG

Episodic — experiences
conversation and action history

Procedural — skills
skill files, reviewed like dependencies

A solution is AI first when an agent can use it without a person in the loop, and a person can still step in at any point.

// SCALE

Agent sprawl is the new shadow IT

Agents will be created exponentially across business units and platforms. The operating model has to exist before they do.

PLATFORM TEAM OWNS

The shared foundation

- Agent gateway, registry and identity
- Context service and retrieval
- Semantic layer and contracts
- Eval harness and red teaming
- Cost and telemetry dashboards
- Reference patterns and skills

BUSINESS UNITS OWN

The agents themselves

- Purpose, scope and success measures
- Domain knowledge and skills
- Approval rules for their actions
- The named owner and the budget
- Retirement when value stops

EVERY AGENT CARRIES

A passport

- Identity and owner
- Purpose and allowed intents
- Data and tools it may reach
- Impact tier and approval policy
- Evals it must pass to publish
- Cost ceiling and telemetry

// THE MICROSOFT STACK AS A WORKED EXAMPLE

Mapped to the Microsoft stack

Buy, extend, build. Choose the layer by the job, not the hype.

Surfaces	Microsoft 365 Copilot · Teams · Power Apps · Dynamics 365
Agent layer	Copilot Studio (extend) · Foundry Agent Service (build) · Agent 365 as the gateway and registry · Entra Agent ID
Capability layer	MCP servers behind API Management · 1,500+ connectors · Power Automate and Logic Apps for deterministic steps · Dataverse as system of record
Information layer	Fabric and OneLake · Dataverse · Azure AI Search · Purview glossary, lineage and classification · Microsoft Graph
Governance plane	Entra identity and conditional access · Purview labels and DLP · Defender for AI · Foundry evaluations and tracing

BUY
Microsoft 365 Copilot
ready to use, grounded in Graph, no build

EXTEND
Copilot Studio
low code agents for a process, your data and rules

BUILD
Microsoft Foundry
pro code models, orchestration, evals, at scale

Trust to scale

Non determinism is a design property, not a bug. Evals, human in the loop, the attack surface, and cost as an architecture input.

// TESTING & EVALUATION

How do you test something non deterministic?

You stop asking "does it pass" and start asking "how well, how often, and is it getting worse".

EVALS

Define what good looks like first

- Scored runs: inputs, expected outcomes, rubrics
- Build the eval set before the agent
- Evals are the contract tests of an agentic endpoint
- A prompt, skill or model change without a green eval run is an unreviewed deploy

RED TEAMING

Attack your own agent

- Prompt injection through documents and tool results
- Goal hijacking and scope creep
- Sensitive data leakage across agents
- Run known attack patterns systematically, and test the refusal paths

MONITORING IN PRODUCTION

Production is a live eval loop

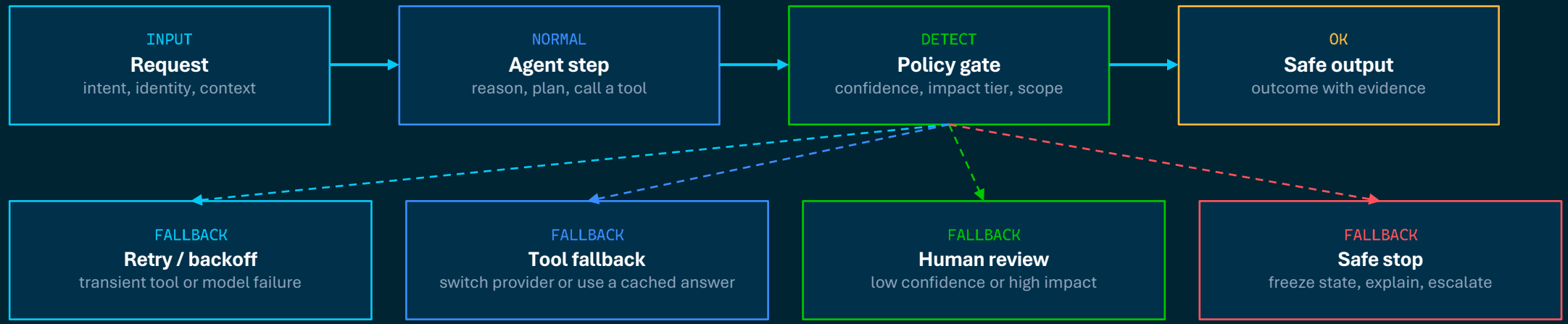
- Log intents, context, tool calls, outcomes, latency, cost
- Sample real traffic and score it
- Detect drift before users feel it
- Feed failures back into the eval set

Publish an SLO per agentic endpoint: quality score, latency, cost per outcome, escalation rate. Treat a breach like an outage.

// HUMAN IN THE LOOP

Human in the loop is an architecture, not a checkbox

Design the escape routes before the agent needs them.



Checkpoint state after every risky hop, so any route can resume or roll back

APPROVE BEFORE

irreversible or costly actions

REVIEW AFTER

reversible actions, sampled

AUDIT SAMPLE

low impact, high volume

The human is a first class route in the graph, with the same tracing, state and SLOs as any tool call.

// THE ATTACK SURFACE

Agents and protocols widen the attack surface

We hardened against SQL injection. Now we harden against prompt injection, tool poisoning and agent impersonation.

WHAT CHANGES

New surface, familiar mindset

- Prompt injection, including transitive injection across agents via A2A
- Tool poisoning through malicious or tampered tool descriptions
- Agent impersonation, agent card tampering and replay
- Merged trust boundaries: instructions, data and tool calls in one context
- Recursive delegation loops and runaway spend

MITIGATE TODAY

Defence in depth, protocol aware

- Least privilege agent identities; scoped, short lived tokens
- Signed agent cards, mTLS, PKI machine identities
- Allow list MCP servers; scan them like any dependency
- Isolate untrusted content from instructions; verify outputs
- Loop, depth and spend guards at the gateway

The protocols are hardening fast. Your controls today: identity, isolation, allow lists, verification, limits. Same list as the last twenty years.

// COST & LATENCY AS DESIGN INPUTS

Token cost and latency are architecture decisions

An agentic endpoint that is correct but slow and expensive will be switched off by finance, not by security.

01 MODEL SELECTION

Match the model to the task

Small models for classification, extraction and routing. Reserve frontier models for synthesis, judgement and planning. Route by task, not by habit.

02 LATENCY BUDGET

Design for the clock

Parallelise independent steps, set per hop SLAs, stream partial results, cache retrieval and prompts. Every agent hop is a network hop with a thinking tax.

03 COST CONTROL

Instrument first

Measure tokens per outcome by agent, tool and user. Cap retrieval and loops. Set spend ceilings in the gateway. Report cost per outcome next to value per outcome.

The unit of economics is cost per outcome, not cost per token. Design the endpoint so that number is visible from day one.

Value side, from the field: time and labour saved, error and rework reduction, cycle time, revenue impact, retired legacy spend. Define when to say yes, and when to say no or not yet.

// CLOSING — THE ARCHITECT'S CHECKLIST

The architect's checklist

Six questions to ask of any solution before you call it agent ready. Photograph this one.

01 JUDGEMENT VS COMMITMENT

Which decisions are agentic, which steps stay deterministic, and where is the line written down?

02 CONTRACTS

Does every capability have an owned contract: schema, semantics, side effects, access, change process?

03 KNOWLEDGE MODEL

What semantic layer do agents reason over, and who owns the definition of "customer"?

04 CONTEXT

Who assembles context, with what budget, priority, freshness and permission trimming?

05 LINEAGE

Can you replay any agent action: intent, context used, plan, tool calls, outcome, approvals?

06 READINESS & SCALE

What level of the ladder is each system, what moves it up, and who owns every agent that exists?

If you cannot answer one of these, that is where your first agent will fail. Fix the foundation before you scale the surface.

// CLOSING — THREE LINES TO TAKE HOME

Agents are the new surface.

Information architecture is the new foundation.

Intent is the new interface.

AI will accelerate building the infrastructure that powers the next AI leap. The foundation you design this year is what somebody's agent reasons over next year.

Thank you

Questions?