Your Agent Is Confidently Wrong

Guardrails for Agentic AI in Real Engineering Work

Girish Konda

Principal Software Engineer, Microsoft

15 October 2026

Let's connect

go.girishkonda.com/conf42

The views expressed in this talk are my own and do not represent those of my employer, Microsoft Corporation.

Gate the claim, not the confidence

A recommendation needs a decision boundary before it becomes an action.

01

Agent

Proposes a claim

→

02

Read tool

Returns evidence

→

03

Evidence

Pins source + scope

→

04

Validator

Checks predicates

↓

Recommendation

Carries evidence IDs

←

Authorized action

Binds exact payload

←

Observation

Checks actual effect

↓

Refuse / re-read

Keep the reason

An approved coding goal is not blanket permission to push or authenticate.

Design model • A gate around recommendations, not a deployed control.

One change, five work surfaces

Teaching case: do the samples support changing the demo API's retry configuration?

Work surface	Decision	Required evidence
Incident triage	Is the signal explained?	Same service, sample, and time window
Telemetry	Did the error rate fall?	Errors AND requests; complete results
Code review	Does this change justify the claim?	Pinned diff, test, and stated limits
Runbook	May the next step proceed?	Current evidence, approval, read-back
Development sessions	May edit / test / auth / commit / push proceed?	Diff + tests + exact approval; current authority at this step

Illustrative fixture • demo-api and its retry policy are fictional. No production data or results.

A polished answer can still fail

The answer is well formed. None of its decision inputs is established.

ILLUSTRATIVE CANDIDATE — NOT AN OBSERVED AGENT QUOTE

```
claim: C1@r1
"Error rate fell 80%."
source_ids: []

basis: retry-policy / section 4
recommendation: change retries
status: confirmed
```

REJECT BEFORE RECOMMENDING

Unsupported number

No rows, denominator, or window.

Unresolved reference

The cited policy cannot be opened.

Unsupported status

No matching effect observation.

Illustrative fixture • The percentage, policy name, and candidate output are invented for teaching.

Resolve the reference, then its claim

A citation has to exist, match the version, and support this decision.

ILLUSTRATIVE SOURCE RESOLVER

```
resolve("retry-policy", "r7")
  -> NOT_FOUND

resolve("demo-runbook", "r7")
  -> FOUND: rollback steps

supports("error rate fell")
  -> NO
```

DIFFERENT FAILURES; DIFFERENT NEXT READS

No document

Ask for a resolvable source.

Wrong support

Rollback steps do not establish health.

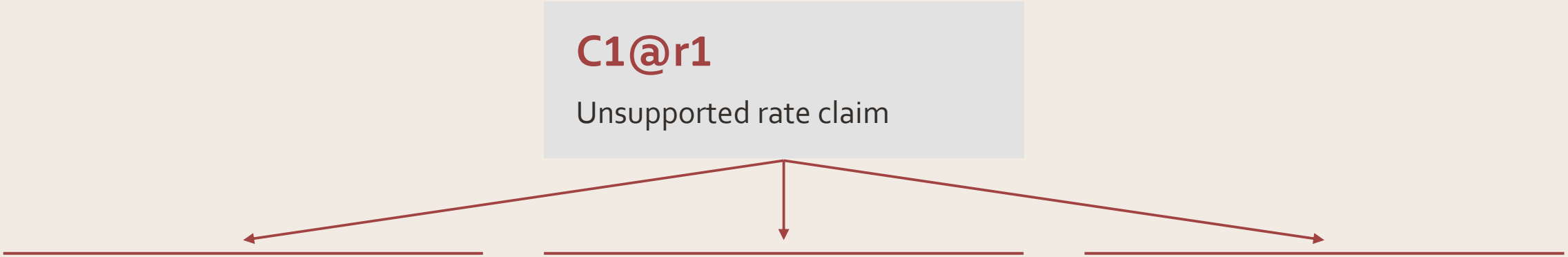
Versioned support

Pin the content, scope, and relevant passage.

Illustrative fixture • Existence is machine-checkable; relevance can still need expert review.

A correction can leave old facts alive

Changing the conversation does not update the artifacts that used it.



Illustrative failure trace • No invented dialogue is presented as an observed conversation.

A long session does not refresh a read

The summary may be new. The evidence can still be too old to use.

09:05Z	Data as of	Window ends 09:00–09:05Z
09:07Z	First decision	Age: 2 minutes Within the fixture bound
09:15Z	Later decision	Age: 10 minutes Re-read required

ILLUSTRATIVE FRESHNESS CHECK — BOUND IS DECISION-SPECIFIC

```
decision_time - data_as_of > 5 minutes  ->  stale
```

Illustrative times. Source: Evidence age at the point of use (IETF discussion).

Give the claim an evidence record

Keep source revision, scope, clocks, and input state together.

ILLUSTRATIVE RECORD — SHORTENED FOR THE SLIDE

```
claim: C1@r2   evidence: E17
source: demo-read@r2
sha256: f946b6cb...3ef4466
scope: demo-api / retry-policy
samples: baseline + candidate
window: 09:00-09:05Z
as_of: 09:05Z  read_at: 09:06Z
evaluated_at: 09:07Z
state: established-negative
```

Example record, not a wire schema. Source: Konda -01; snapshots or references.

WHAT THIS LETS YOU CHECK

Open the same bytes

Source ID + immutable version.

Ask the same question

Service, revision, samples, and window.

Reconstruct the decision

A new read time does not make old evidence fresh.

Unknown is not false

A missing result, a negative finding, and an old value are different facts.

Input state	What was established	Possible next step
established-positive	Predicate checked; satisfied	Continue evaluation
established-negative	Predicate checked; not satisfied	Refuse on that finding
not-established	Required input unavailable	Re-read or route
stale	Known value is now too old	Refresh the input
not-applicable	Policy does not require it	Record why it is omitted

Source: Konda -01, section 6 — per-input states; local policy derives the decision.

Keep the failure reasons distinct

The same refusal can mean different failed facts—and different next checks.

Record	Meaning	Next check
Input: not-established	No usable probe value	Repair or repeat the read
Input: established-negative	Observed predicate failure	Review the actual finding
Input: stale	Value exceeds the age bound	Refresh at point of use
Effect: not-observed	No matching effect evidence yet	Read back by action ID
Effect: indeterminate	Effect cannot be determined	Reconcile; do not guess

Source: Konda -01, section 6. Effect examples are illustrative, not standardized states.

Put the gate outside the prompt

Recording an unavailable input is not enough; the consumer must enforce the refusal.

ILLUSTRATIVE PYTHON — REQUIRED CHECKS ARE CALLER-OWNED

```
POSITIVE = "established-positive"
if not required_checks:
    return Refuse("missing-checks")
for check in required_checks:
    if check.get("state") != POSITIVE:
        return Refuse(check["reason"])
return EligibleForHumanReview()
```

CHECK BEFORE EMIT

Source + scope + freshness

Resolve and validate the required inputs.

Arithmetic + contradictions

Recompute; stop on conflicting reads.

No recommendation on refusal

Return the reason and next read.

Illustrative local recommendation policy • Not a universal Internet-Draft requirement or a production control.

Recompute the actual rate

Same demo service, revision, and window. Different traffic volumes.

Sample	Errors	Requests	Error rate
Baseline	10	1,000	1.0%
Candidate	6	600	1.0%

Raw errors

10 → 6 (–40%)

Error rate

1.0% → 1.0% (no change)

Rate reduction

```
1 - (6/600) / (10/1000) = 0%
```

C1@r2: No rate reduction in this sample. Do not recommend the change.

Illustrative fixture E17 • Window 09:00–09:05Z. A point comparison, not statistical or causal proof.

Invalidate every dependent artifact

A durable correction changes what later consumers are allowed to use.

ILLUSTRATIVE CORRECTION RECORD

```
claim: C1
revision: 2
supersedes: C1@r1
reason: matched-rate calculation
replacement_evidence: E17
current_revision[C1]: 2

consume(C1@r1) -> REJECT
```

1 **Block r1 dependents**

↓

2 **Rebuild from C1@r2 + E17**

↓

3 **Re-check before the next use**

Example dependency graph. Source: Recording decision-time context (IETF discussion).

Approval binds to the exact action

Even a supported recommendation is not permission to perform a write.

ILLUSTRATIVE PERMITTED BRANCH — NOT EXECUTED

```
action_id: A7
operation: set_retry_limit
target: demo-api
scope: staging
value: 3 retries

grant binds:
  action ID + payload digest
  approver + expiry + single use
```

CHECK AGAIN AT THE WRITE BOUNDARY

Different scope?

Reject. Staging approval is not production approval.

Changed text or payload?

Get approval for the new artifact.

Already consumed or expired?

Reject. No carried-over permission.

Design guidance • Our worked telemetry case was refused; this shows the separate branch after evidence passes.

Resume restores context, not permission

The same conversation is not current authority to repeat a side effect.

ILLUSTRATIVE CONFIG RESUME — NO ACTION EXECUTED

```
resume_context: A7 / staging
grant: recheck current status
input_age: check at next use

revocation_check: unavailable
local_policy: refuse + signal
side_effect: not resubmitted
```

RECHECK BEFORE THE NEXT WRITE

Restore the context

Do not replay side effects as a resume step.

Check current authority

Participant, termination, and revocation are different checks.

Keep failure visible

Unknown is not permission. Signal the failed check.

Source: B1-7 resumption requirement (merged text, not a filed revision); resumption discussion.

Submitted is not confirmed

Keep evaluation, operation status, and observed effect on separate axes.

ILLUSTRATIVE ACTION RECORD

```
action_id: A7
evaluation: evaluated
decision: permit
operation: accepted
effect: not-observed

read_back(action_id="A7")
  -> matching effect evidence
  -> confirmed
```

Example record. Source: Gilda -02, section 5.4 — evaluation status is separate from decision.

WHAT EACH SIGNAL ACTUALLY SAYS

Submitted / accepted

The request was sent or admitted.

Executed

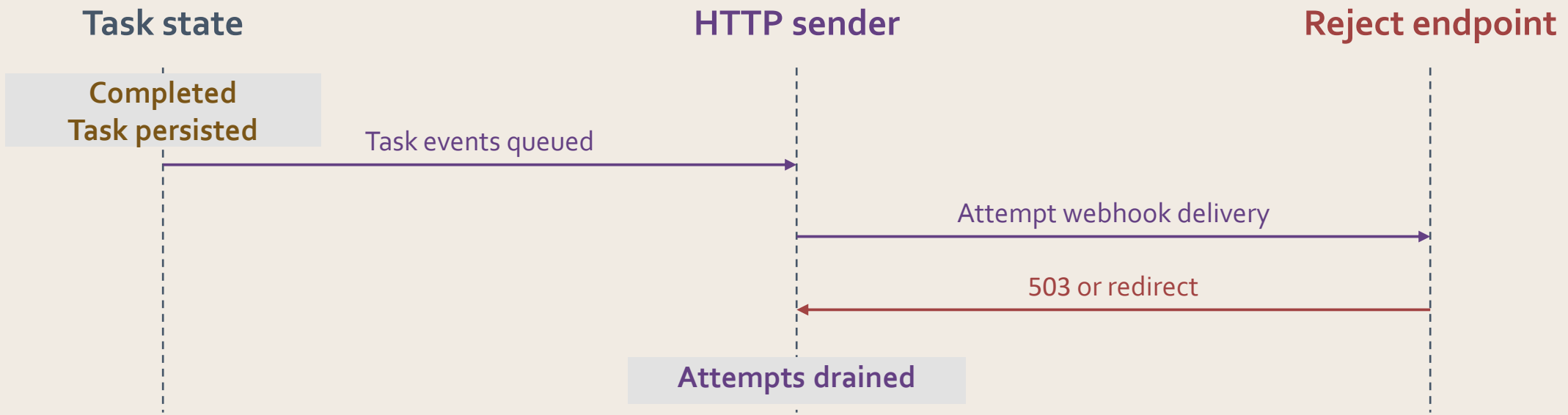
A component reports its own work.

Confirmed effect

A separate observation matches the intended result.

Completed task ≠ accepted notification

Public regression: delivery failure must not rewrite task state or delete registration.



Task	Delivery	Notifications	Registration
Completed	**Attempts drained**	**0 accepted**	**Retained**

Fixture: two attempts per event; redirects get one.

Source: Public PR #527 @ 5d6ca99 · 9 Oct 2026 · source-inspected, not run here; open, not shipped.

Assert the boundary, not the story

The regression is narrow. Keep the implementation limits next to it.

PUBLIC TEST EXCERPT — LINE BREAKS ADJUSTED

```
await DrainAsync(host);
Assert.Equal(redirect ? 3 : 6, attempts);
Assert.Empty(_received);
Assert.Equal(TaskState.Completed,
  (await client.GetTaskAsync(
    new() { Id = task.Id })).Status.State);
Assert.Single(await host.Services
  .GetRequiredService<IPushNotificationStore>()
  .GetAllAsync(task.Id));
```

WHAT THIS DOES NOT ESTABLISH

No atomic outbox

State and queue are separate stores.

No guaranteed receipt

Finite attempts can end without delivery.

No restart durability

The documented queue is in memory.

Source: PR #527 test; delivery/storage limits at 5d6ca99. Source-inspected, not run here.

Name the control you actually have

Instruction, human review, and code enforcement are different controls.

Control	What it can do	What it cannot prove
Prompt policy	Request inline sources and checks	That a skipped check blocks output
Human review	Judge relevance and approve scope	That every consumer honors the check
Executable gate	Reject unsupported structured input	That the source itself is truthful
Effect observation	Confirm a matching observed result	A wider outcome it did not observe

Design guidance • Not a before/after study or a claim that all four controls are deployed.

Replace effort with an exit condition

Judge a control by the invariant it checks, not by how much activity it creates.

Avoid relying on	Use instead	Stop / escalate when
"Remember the rule"	Caller-owned check before use	A required input is unresolved
Only the latest check	Accumulated regression cases	A previous invariant regresses
Review until it feels right	Named exit criteria + review budget	Coverage is met or an owner is needed

Design guidance only • No claim about personal trials, dropped controls, or measured effectiveness.

Make the failure cases executable

Test the refusal paths and the preserved state—not the confidence of the reply.

Inject this	Expected decision / state
Missing or unresolved source	not-established; no recommendation
Wrong scope, partial read, or conflict	not-established; re-read or adjudicate
Read exceeds the age bound	stale; refresh the evidence
Superseded claim revision	Local revision rejection; rebuild dependents
Matched counts, unchanged rate	established-negative; reject improvement claim
Changed payload or reused approval	Reject authorization
Accepted action; no observation	Keep effect unconfirmed

Offline illustrative fixture checks run locally • Separate public callback regression was source-inspected, not executed.

What the small fixture leaves out

A runnable example is a starting point, not the complete trust boundary.

IMPLEMENTED IN THE OFFLINE FIXTURE

Immutable synthetic snapshots

Digest, scope, window, and freshness checks.

Recomputation + invalidation

Rate predicate; invalidated dependencies.

Payload match + observation ID

Single-use illustration; matching effect.

REQUIRED FOR A REAL ENFORCEMENT SYSTEM

Trusted readers and policy

Authenticate sources and approvers.

Durability and concurrency

Atomic state transitions; crash recovery.

Complete consumer coverage

No bypass path; observed effect at the right scope.

Design limits • No live actions, production benchmark, reliability gain, or end-to-end deployment claim.

Sources and references

Public source material and its limits—not adopted standards or a shipped release.

DRAFTS AND IMPLEMENTATION

Konda -01 evaluation states
Filed 10 Oct • individual draft, not adopted

A2A task / notification regression
PR #527 at 5d6ca99 • open, source-inspected

A2A delivery and storage limits
Same commit • no delivery/durability guarantee

Gilda -02 audit record
Section 5.4 • evaluation status and decision

Resuming an interrupted agent
B1-7 • merged working text, not a filed revision

PUBLIC DISCUSSIONS

Evidence age at the point of use
Public discussion • freshness at decision time

Snapshots or references
Public discussion • representation remains open

Record decision-time context
Public discussion • preserve what was known

Safe resumption after interruption
Public discussion • authority and failure checks

Distinguish missing from stale input
Public discussion • open state-model question

The examples are illustrative. These links identify the public material behind the technical claims.

Force the source, or drop the claim.

Evidence before recommendation.
Approval before action. Observation after it.



Girish Konda / Microsoft

The views expressed in this talk are my own and do not represent those of my employer, Microsoft Corporation.

Nothing here is an endorsement, a commitment, or a statement of product direction. Examples are generalized and contain no customer data.

Presentation template by Slidesgo.

Let's connect

go.girishkonda.com/conf42