MICROSOFT

Agentic Engineering

Designed for Imperfection

Errors are the premise — quality nets are the answer.

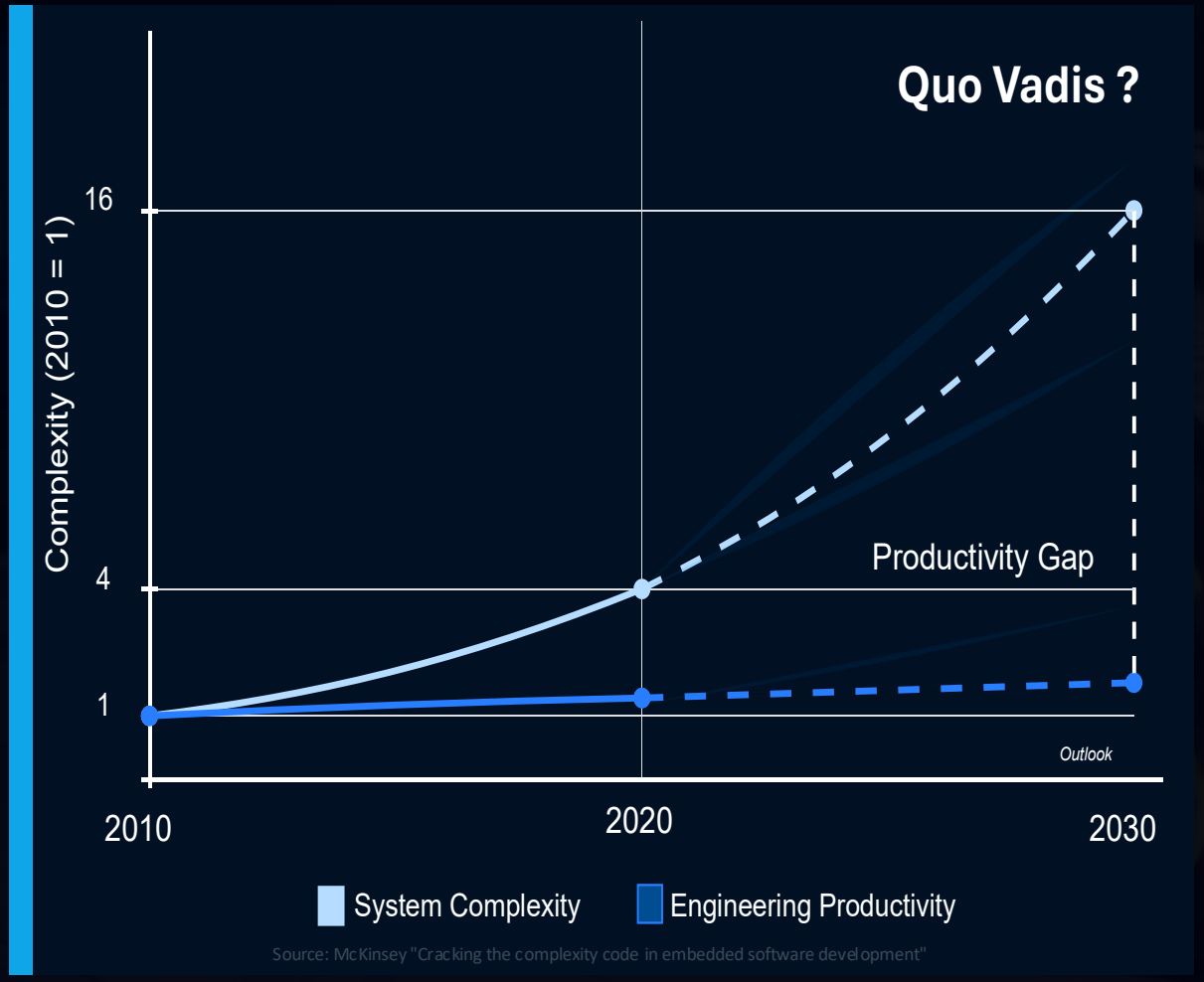
July 2026 · **Georg Doll**, CTO Automotive & Mobility

Agentic.

Engineering becomes agentic

PRODUCTIVITY AND COMPLEXITY HAVE BEEN DIVERGING FOR DECADES

Feature and regulatory requirements are outpacing the pace of development.



16×

System complexity growth 2010 → 2030

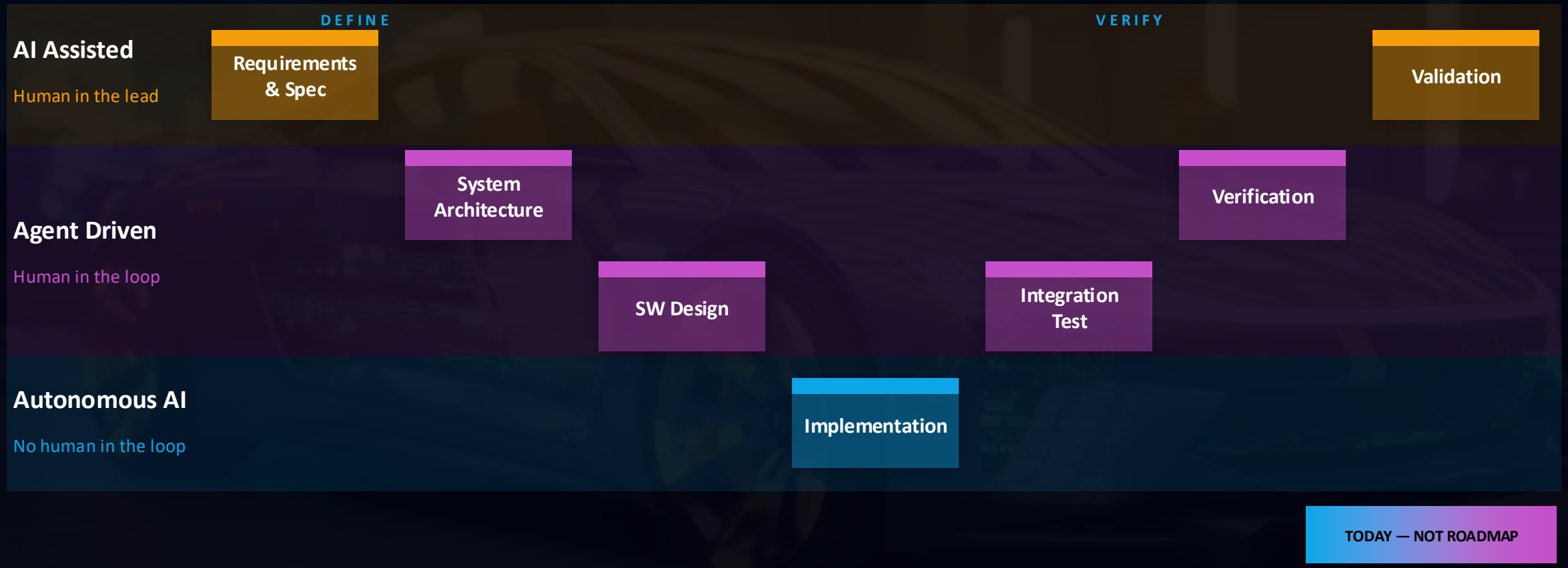
+40%

Engineering productivity growth (same period)

GenAI can help close this gap — but only with the right structure.

THE STARTING POINT — CAPABILITY TODAY

AI already covers more of the engineering workload than most realize.



Today, AI already spans the entire software-development process — every phase, at some level of autonomy.

Microsoft

More agents — higher speed AND higher quality.

dotnet/runtime — one of the most complex OSS repos. 10 months of Copilot Coding Agent. (Source: Microsoft)

878	↑68%	↑69%	↑77%	↓0.6%
PRs created by agent	Merge rate (from ~50%)	Success rate (from 38%)	Agent share of all PRs (from 19%)	Agent revert rate vs 0.8% human

The lever: a clear copilot-instructions.md — build conventions, test rules, architecture constraints. Better SPEC, not a better model. More agents = higher speed AND higher quality.

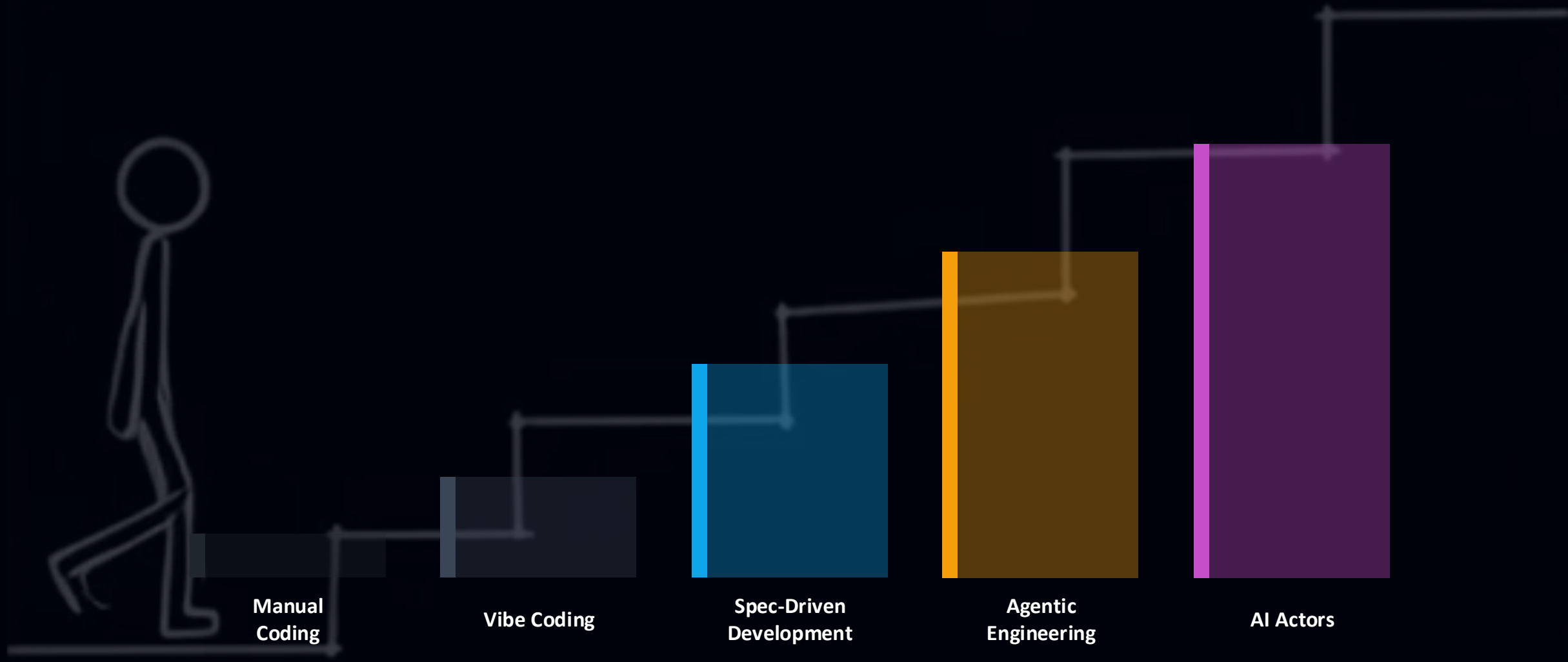
Actors.

The evolution from chat to actor

Carl Hewitt · Actor Model · 1973

FROM PROMPTING TO ACTING.

From writing code to managing the lifecycle.



WHERE WE ARE — HONESTLY

Great at tasks. Lost on the long run.

1

Today

Agents already handle bounded tasks well — a feature, a fix, a refactor. But on long tasks they drift, and chained agents multiply each other's errors.

2

What we need

A clear intent, a goal to reach, and guidelines to move along — so the work stays on course as it grows.

3

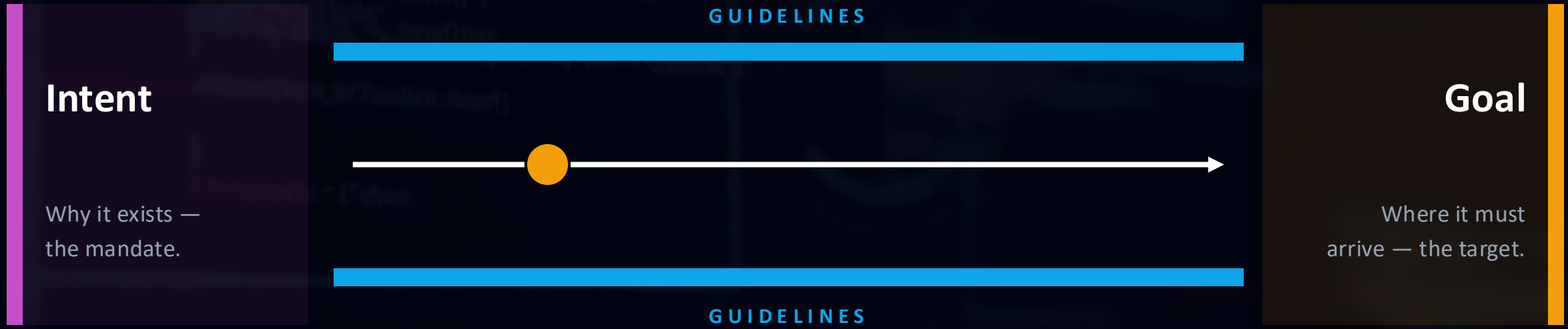
Way forward

Give them the freedom to learn, persist, and self-correct along the guardrails. They become Actors.

“Actor” — Carl Hewitt, 1973: the Actor Model. Concurrent, addressable, message-driven.

WHAT AN ACTOR NEEDS TO MOVE ON ITS OWN

A goal is not enough — you need a lane.



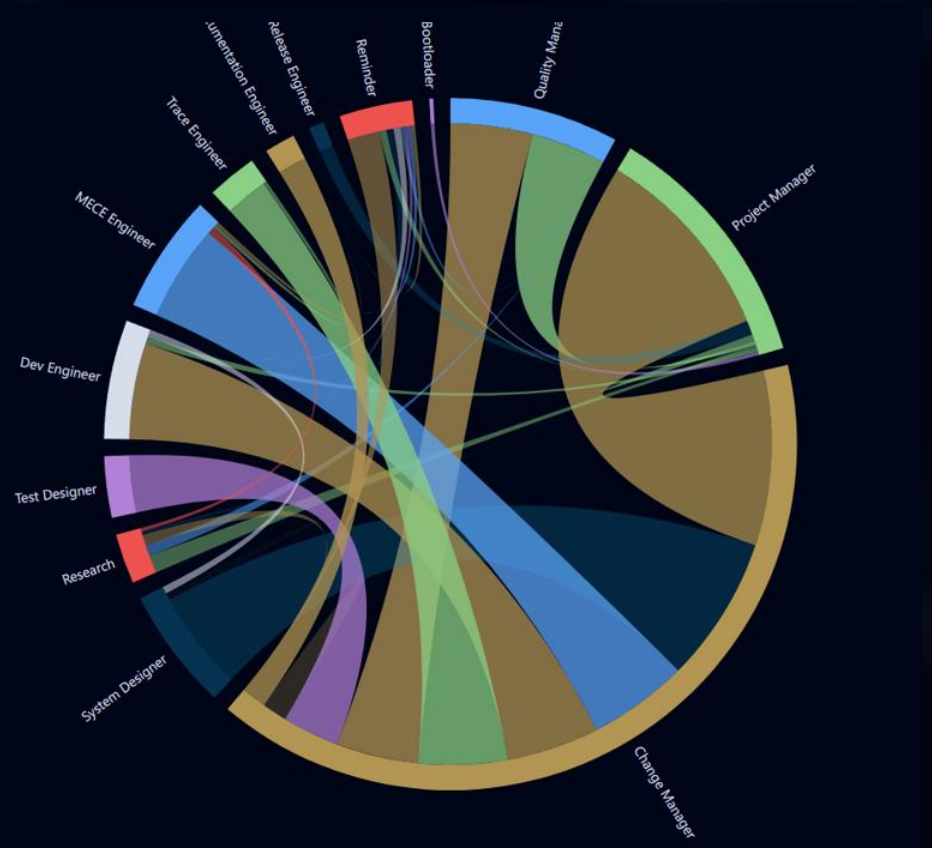
A goal alone is a wish. Guidelines are the road that carries an intent to its goal.

A single actor barely learns. An organization does.

One actor learns only from its own guardrails — narrow.

Connect actors — roles talking to roles — and judgment accumulates across the organization.

Learning is the mesh getting tighter over time — not the model getting bigger.

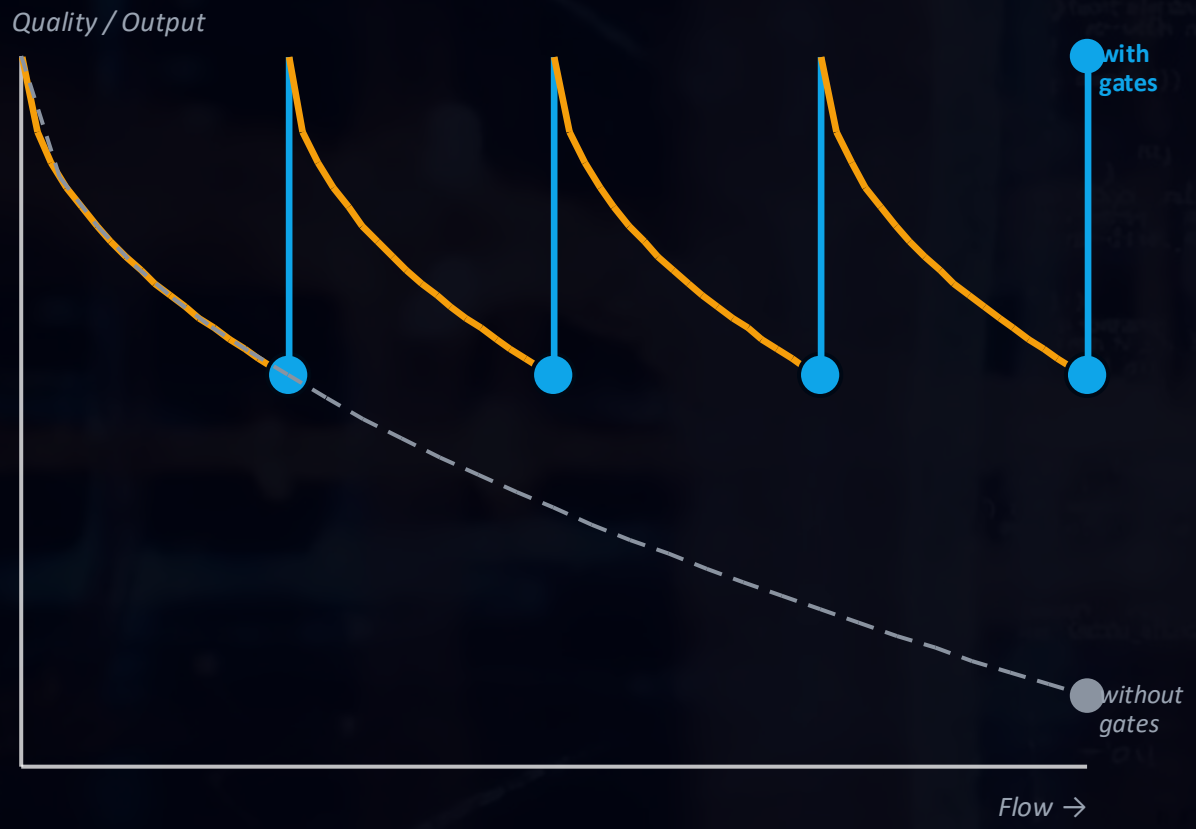


Real inter-actor communication in a running agentic organization · syspilot · github.com/enthali/syspilot

Imperfection.

Designed for imperfection

LLMs don't decide — they estimate.



errors accumulate — quality decays; each Q-gate resets us to 100%

Neural nets output probabilities — not reproducible 0/1 decisions.

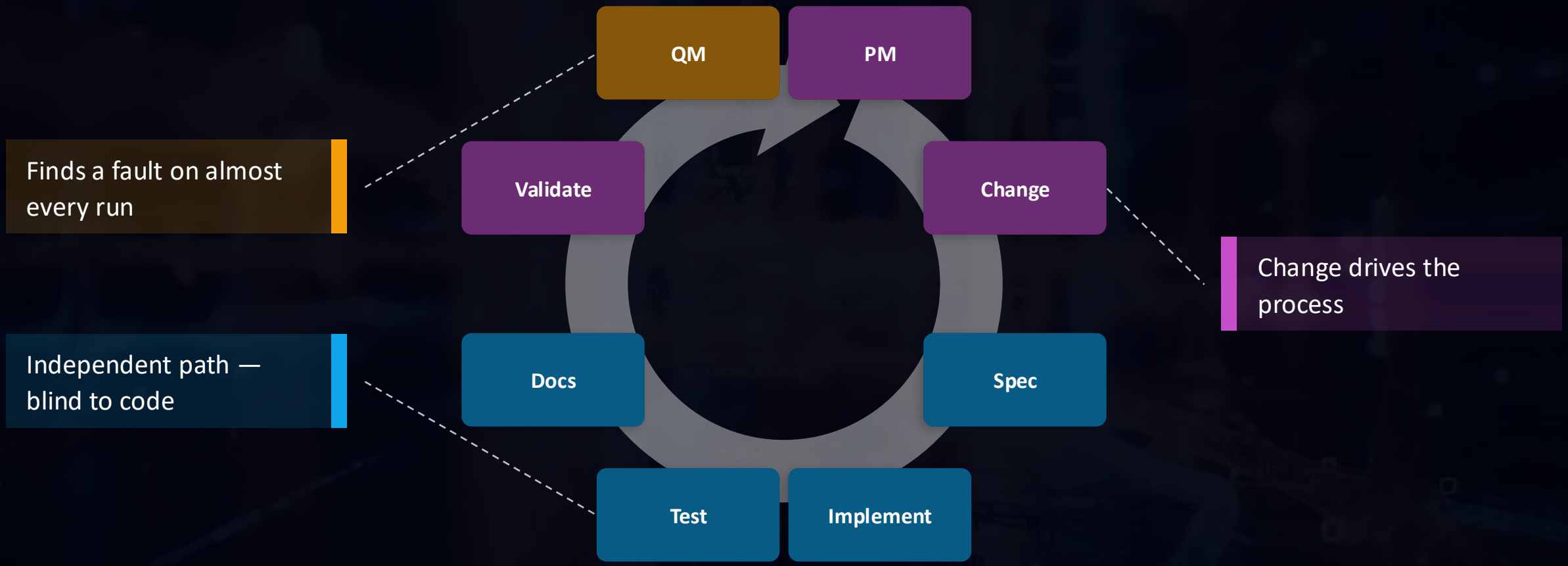
Every step introduces error. Many steps in sequence → compound error rate.

Errors are not a bug to engineer away — they are the design premise.

So we place a quality gate after every step — **back onto the path of virtue.**

ONE CHANGE, THROUGH THE ORGANIZATION

Continuous improvement — not a waterfall.



One change flows through the whole organization — and quality is never reached by LLMs alone.

Microsoft

Two layers — because agent-tests-agent is non-determinism squared.

Layer 1 · Deterministic algorithm

- Correctness. No model involved.
- If the rule says it fails — it fails, every time.
- Traceability · schema · completeness checks

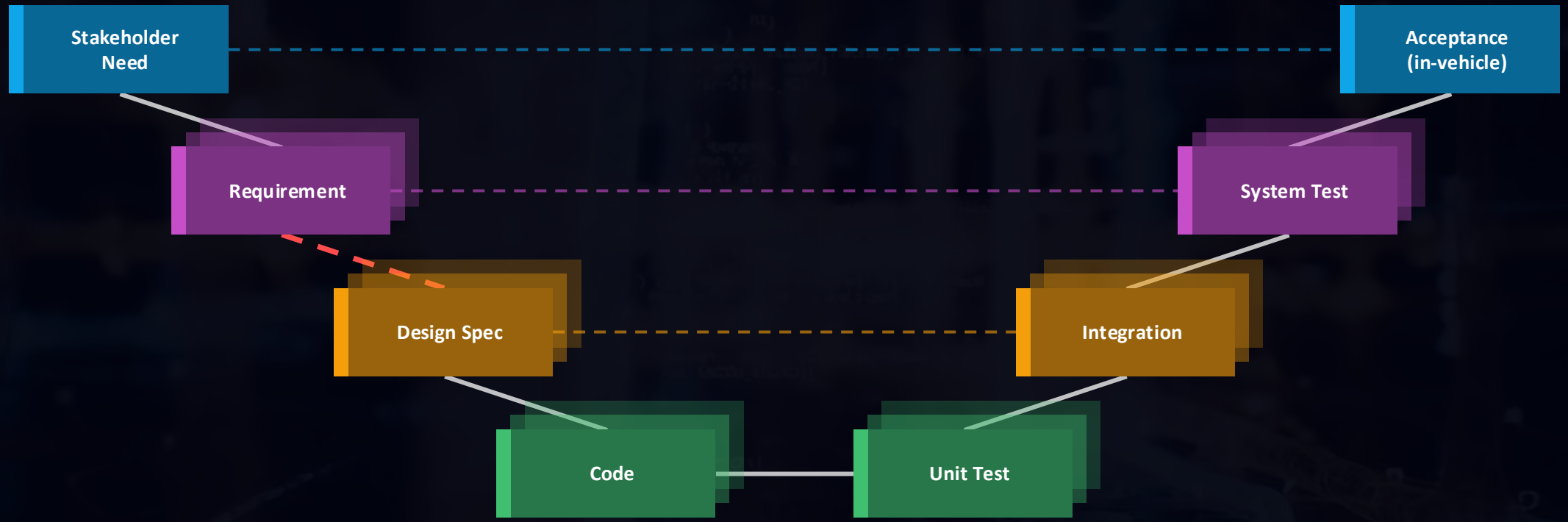
Layer 2 · Semantic agent

- **Quality & completeness — interpretive**
- **Catches what rules can't: intent, ambiguity, gaps**
- **Spec coherence · requirement clarity · edge cases**

Agents where agents belong — **algorithms where correctness is required.**

Microsoft

Traceability is a tree — maintained and consistent with the code.



Left: what to build, top-down. Right: how it's proven, bottom-up — each level with its own test level. Agents build and maintain the whole graph; a missing link is findable, not just felt.

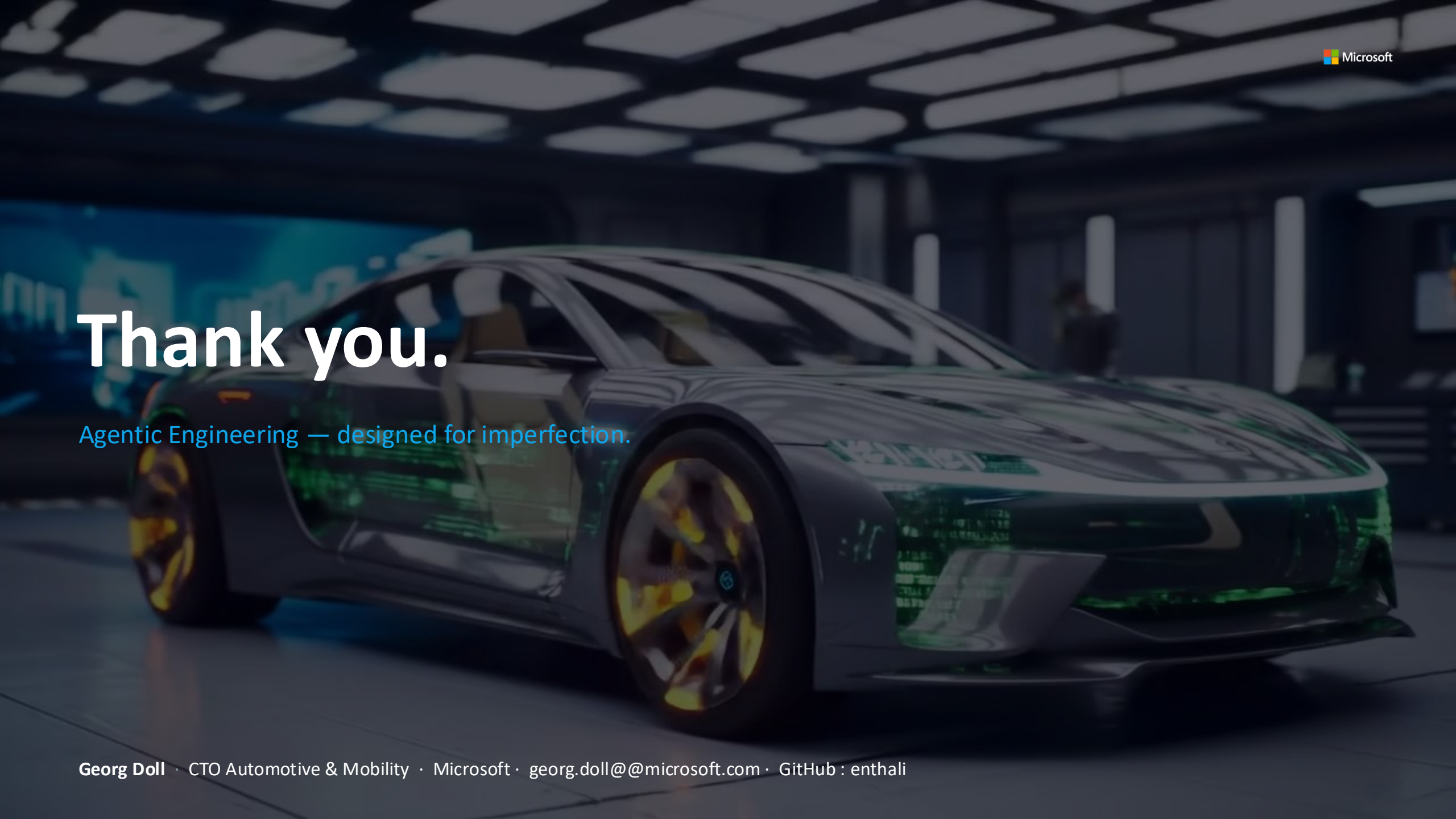
Microsoft

The circle that learns itself.

The validation run is where virtual testing lives.

The loop is designed to become self-learning — the organization improves its own nets over time.

But self-learning only works once boundaries, gates, intent and goal are sharp.



Thank you.

Agentic Engineering — designed for imperfection.

Georg Doll · CTO Automotive & Mobility · Microsoft · georg.doll@@microsoft.com · GitHub : enthali