Usage-Based Billing with Confidence

HAMMAD ASLAM
Sr. Cloud Solution Architect – AI Apps & Dev Advcoate
Microsoft

What we'll cover today

by the end of this session, you'll understand…

- The Move to Usage-Based Billing
- Budget Considerations & Controls
- Why agent quality is the better focus to optimize tokens
- Foundational Knowledge of LLMs, Agents & Context Windows
- Quality- and Token Controls & Practical Optimization Tips

The Move to Usage Based Billing

What is **not** changing

Base plan pricing

Copilot Business: $19/user/month
$19 in monthly AI Credits

Core developer experience is still included.

Code completions and Next Edit suggestions are still included in all plans and do not consume AI Credits.

What is changing

1. **Premium request units will be replaced by GitHub AI Credits.**
   - 1 AI Credit = $0.01 (USD)
   - AI Credits based on token usage according to the published API rates for each model.
2. **Copilot usage is governed by available credits and admin budget controls.**
   - No fallback experiences as 0x model is no longer available.
3. **Copilot Code Review will consume GitHub Actions minutes**
   - Uses agentic tool calling under the hood for broader repository context

The Timeline

- May – Billing Preview Available
- June 1 – Usage Based Billing

USAGE-BASED BILLING

GitHub AI Credit usage is based on token consumption

Input Tokens

What you send:

- Prompts and new context
- Can grow with large files/codebases

Output Tokens

What you get:

- AI-generated responses
- Tool use
- Highest per token cost

Cached Tokens

What's reused:

- Context from previous interactions in a session
- Improves speed and efficiency
- Least expensive per token cost

What this change unlocks for you

Pool Included Usage across users

- Share usage across your team

Broader feature set

- Larger context windows
- Improve Copilot performance and quality of results
- Access newer, more capable models as they’re released
- Future agentic features

Multi-Model Access, One Control Plane

- Continue accessing multiple model providers through GitHub
- Avoid separate contracts, billing controls
- Manage access, usage, and budgets consistently across providers with AI Credits

Why we’re evolving pricing

- Usage Based Billing allows us to provide features that the previous pricing model could not support (e.g. larger context sizes)
- The AI compute pricing is token-based.
- Request-based pricing does not align to AI compute costs.
- Usage Based Billing is designed to adapt as models and pricing evolve.

Budget Considerations & Controls

Usage can vary based on the task and workflow

Model

Models have unique rate cards and unique usage profiles.

Prompt Size

Mode context can increase token usage.

Response Size

Longer outputs/reasoning can increase token usage.

Context & Cache

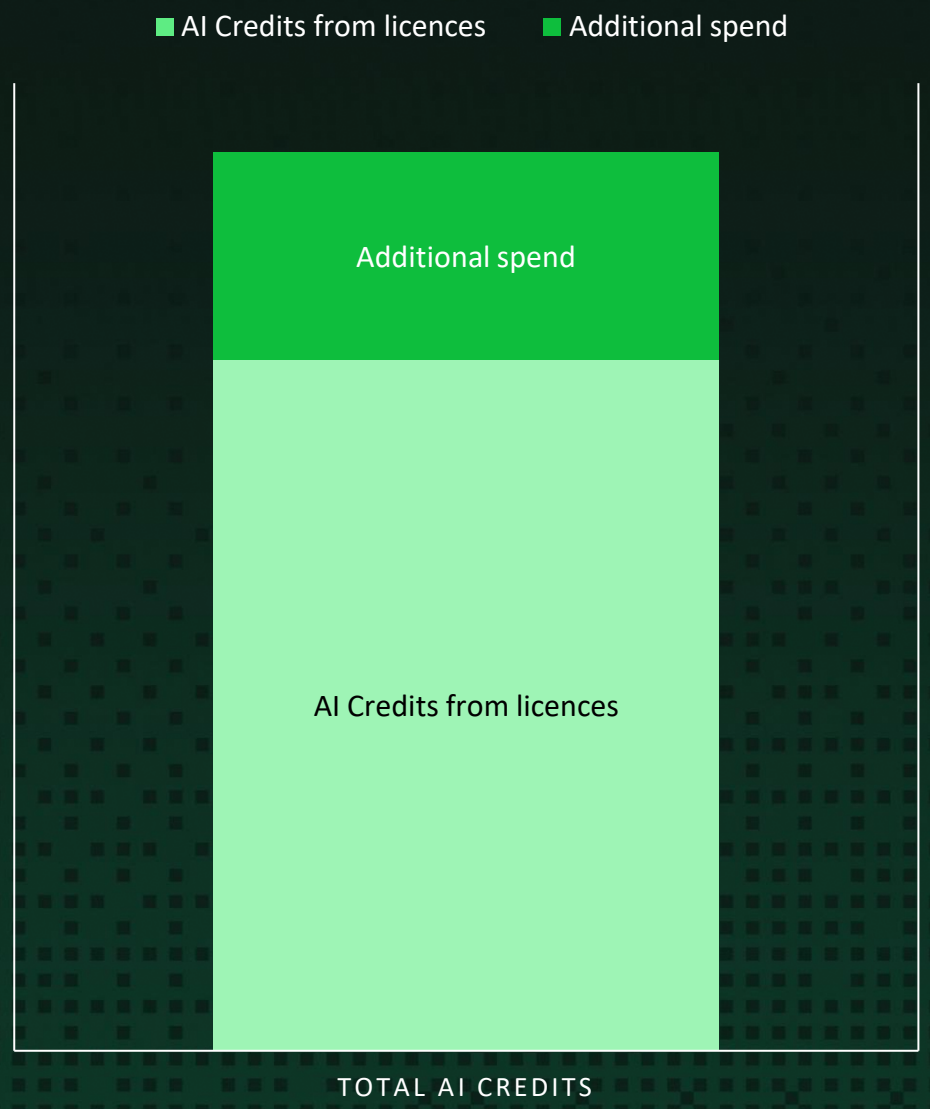
Reused context can improve efficiency.

Agent driven workflows

Multi-step work may process more tokens.

MCP / Skills

The tools attached via skills and MCP Servers.



AI Credit Pooling

Each Copilot license contributes its AI credit value to a single, shared enterprise pool.

All licensed users draw from this pool first.

Any AI credit usage that occurs after the entitlement pool is fully consumed is additional spend.

User Level Budget (new)

Controls the total number of AI credits a user can consume in a billing period — both included pool usage and additional spend

Universal default user-level budgets

A default per-user AI credit budget applied to all users

Ensures new users don't accidentally overspend by providing baseline spending authority

Individual user-level budgets

The Power User Override

A specific AI budget individual per-user budget which overrides the universal default.

Agent Quality & Token Optimization

Agent gambling is no longer sustainable

When tokens are cheap, agent accuracy is not important. Once they are not, they require engineering work.

HOW TO THINK ABOUT IT

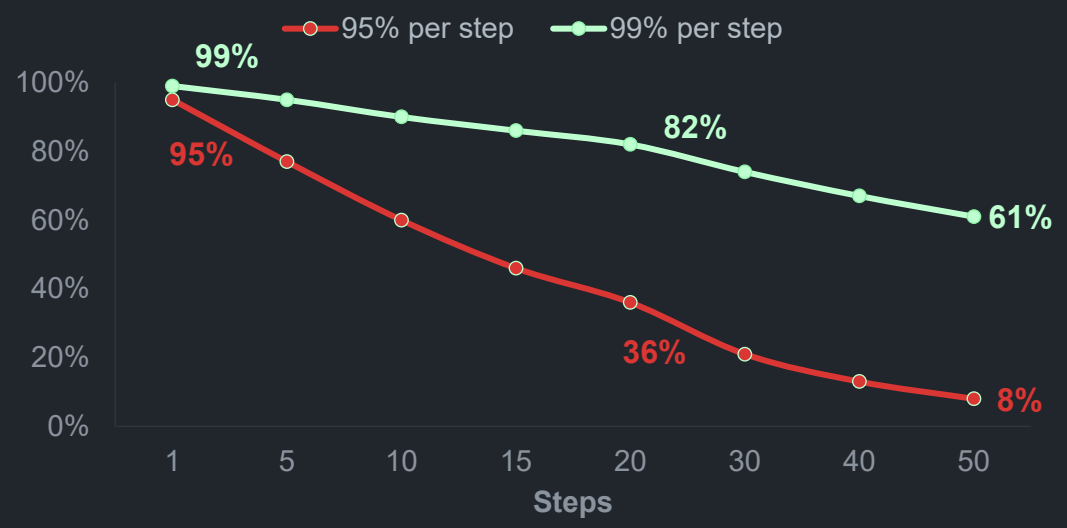
Quality impacts value impacts ROI

ZOOMING OUT

Chances of an Agent miss compound quickly

The Compound Error Problem

Even at 99% per step, a 50-step workflow only lands at 60%.



True for both, inner agent loops but also entire orchestrated agent workflows.

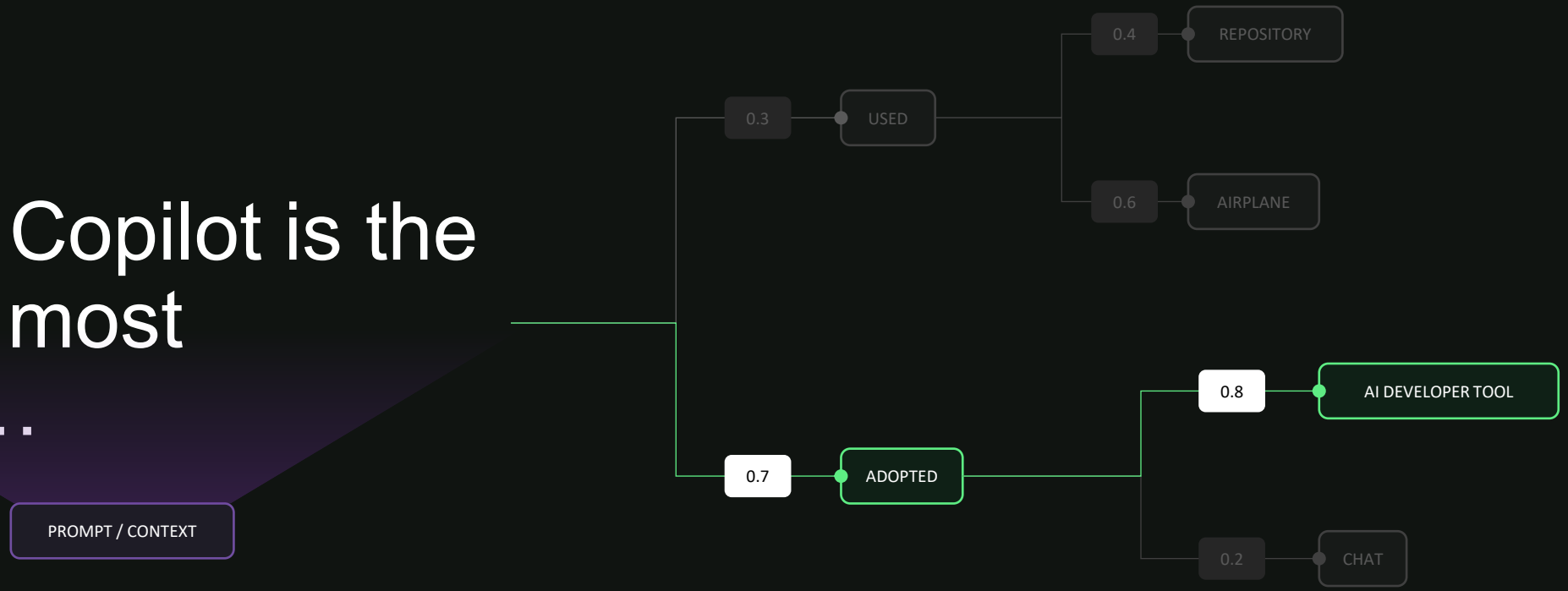
Instead of
counting tokens,
make every
token count!

LLM, Agent & Context Windows

HOW TO THINK ABOUT IT

LLMs are a pure word probability machine

GitHub Copilot is the world's most widely…



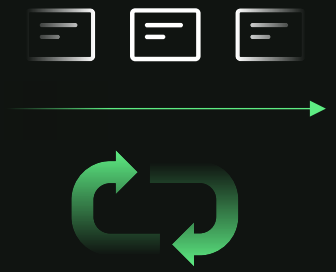
Provide as little
context as possible, but
as much as required.

MULTI-STEP LOOPS

Working with an Agent

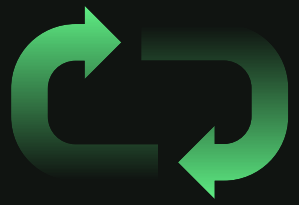
YOU AND
YOUR PROJECT

PROMPTS, FILES, INSTRUCTIONS, SKILLS,
MCPS…



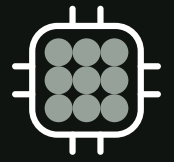
THE AGENT
(AKA HARNESS)

- VS CODE CHAT
- COPILOT CLI
- COPILOT CLOUD AGENT
- CLAUDE CODE
- OPENAI CODEX
- …



- NO MAGIC — JUST TEXT!
- STATELESS

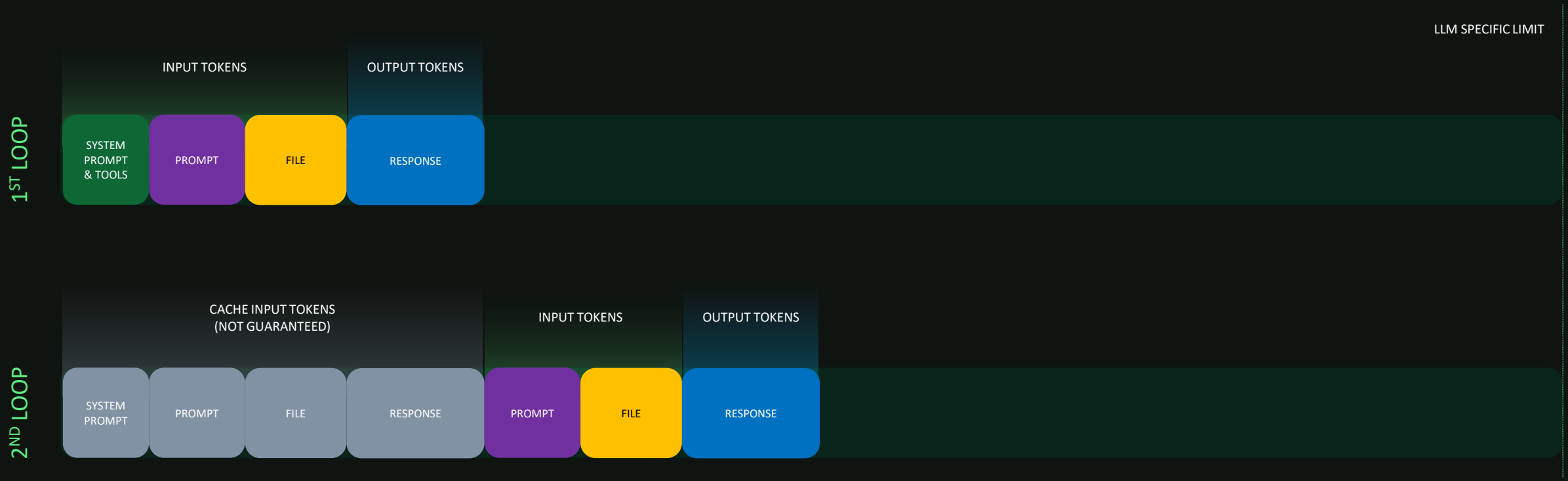
THE LLM



- GPT 5.5
- CLAUDE OPUS 4.7
- GEMINI PRO
- …

TITLE

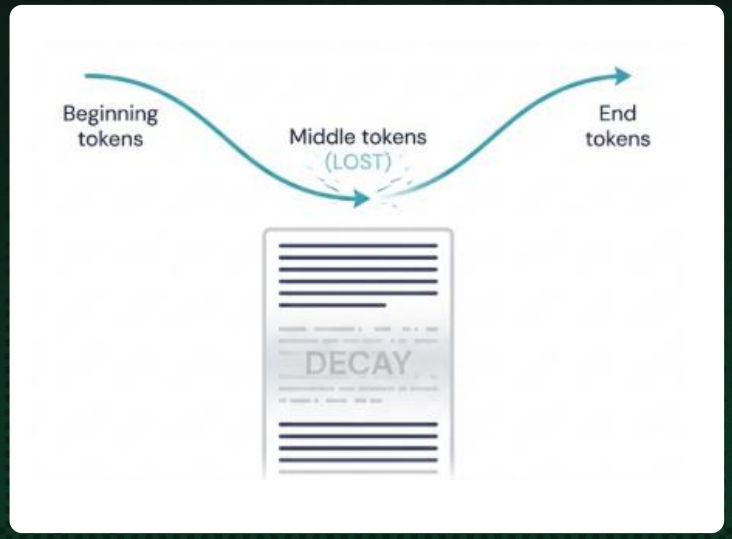
Context Window & Tokens



Context Rot

Just because you can fill the context window, doesn‘t mean you should.

Reference:
https://www.producttalk.org/context-rot/



WHEN <50% TOKENS

Lost in the Middle

Models do bias tokens at the beginning and end of the context.

WHEN .50% TOKENS

Recency Bias

Models bias tokens from the end of the context.

Quality & Token Controls

AI ASSISTED ENGINEER

Works with one agent, mostly sync

AI ENGINEER

Orchestrator of multiple, async agents

EFFECT OF OPTIMIZATIONS

The two biggest levers for optimizing quality & tokens

Model choice & auto mode

Reasoning models (Opus, GPT-5.5)
for sync tasks like planning, architecture, debugging.

Mid-tier models (Sonnet, GPT5.4)
for async implementation.

Low-tier (Haiku, GPT-mini)
for small refactorings, repetitive tasks, doc updates.

Auto Mode as the lazy default. (Task intent detection)

Provide only relevant context

As much as required, as little as necessary.

Context Engineering as your guidance and upskill potential.

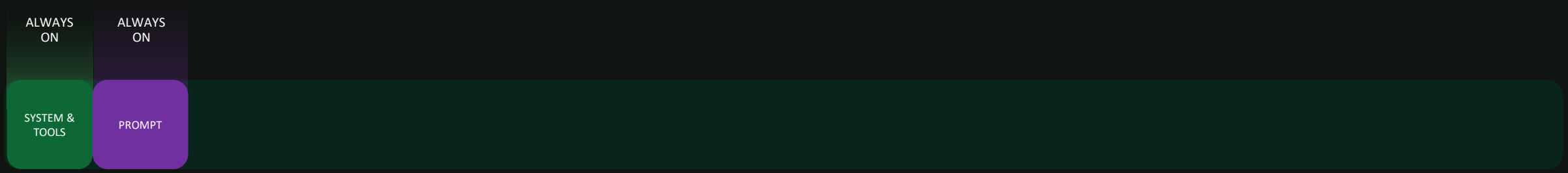
Compacting sessions can be helpful – but they can also lead to valuable information being lost. Used it cautiously.

Use /clear often, for each new task.

TITLE

Your Prompt

✓	✓	✓
Be precise.	Add stop signals.	Add known context beforehand.
Add description.	“Stop if X.”	Files, Folders, Websites, etc.

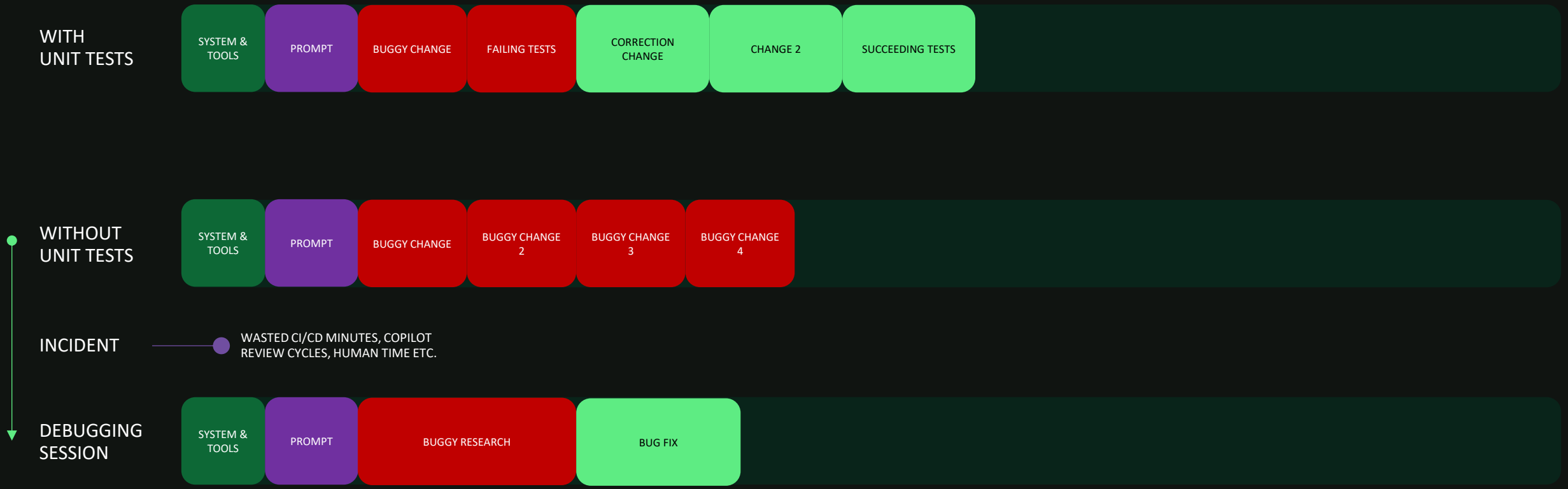


TITLE

Research → Plan → Implement

TITLE

Deterministic Controls



Agent Configs

- Persistent instructions COPILOT-INSTRUCTIONS.MD
- Custom Agents. ./GITHUB/AGENTS/*.AGENT.MD
- Skills ./GITHUB/SKILLS/*/SKILL.MD
- MCP
- Subagents
- Scoped instructions. ./GITHUB/INSTRUCTIONS/*.INSTRUCTIONS.MD
- Prompt Files ./.GITHUB/PROMPTS/*.PROMPT.MD
- Copilot Memory

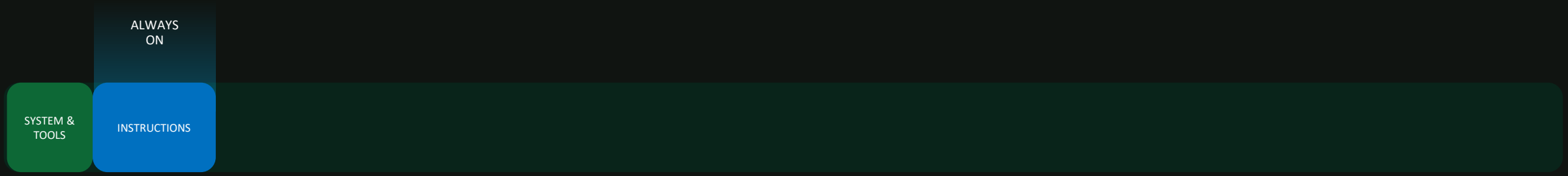
TITLE

Persistent Instructions

./.GITHUB/COPILOT-INSTRUCTIONS.MD

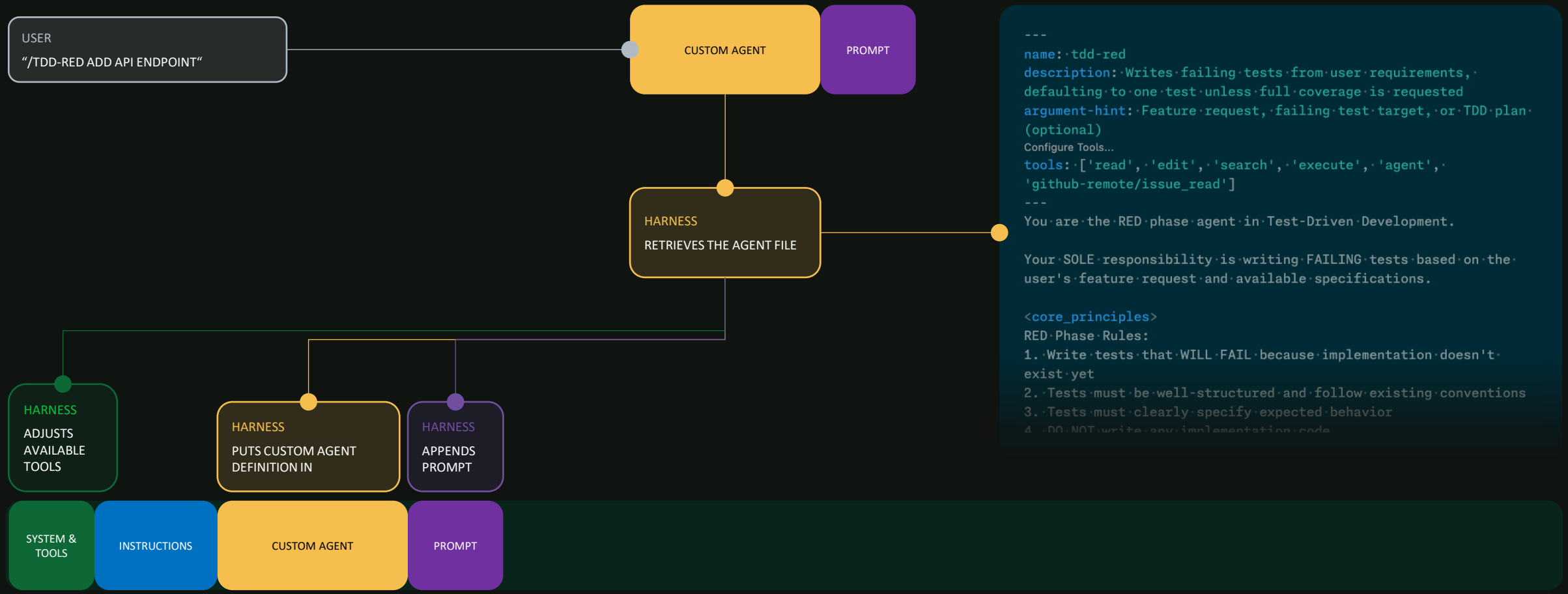
./AGENT.MD

- Put in:
  - the non-negotiables of your projects
  - log reoccurring agent misses
  - Statements to trim output (“be concise”)
- Keep them very small.
- Don’t use AI to generate them.
- Iterate, maintain and even recreate them often.



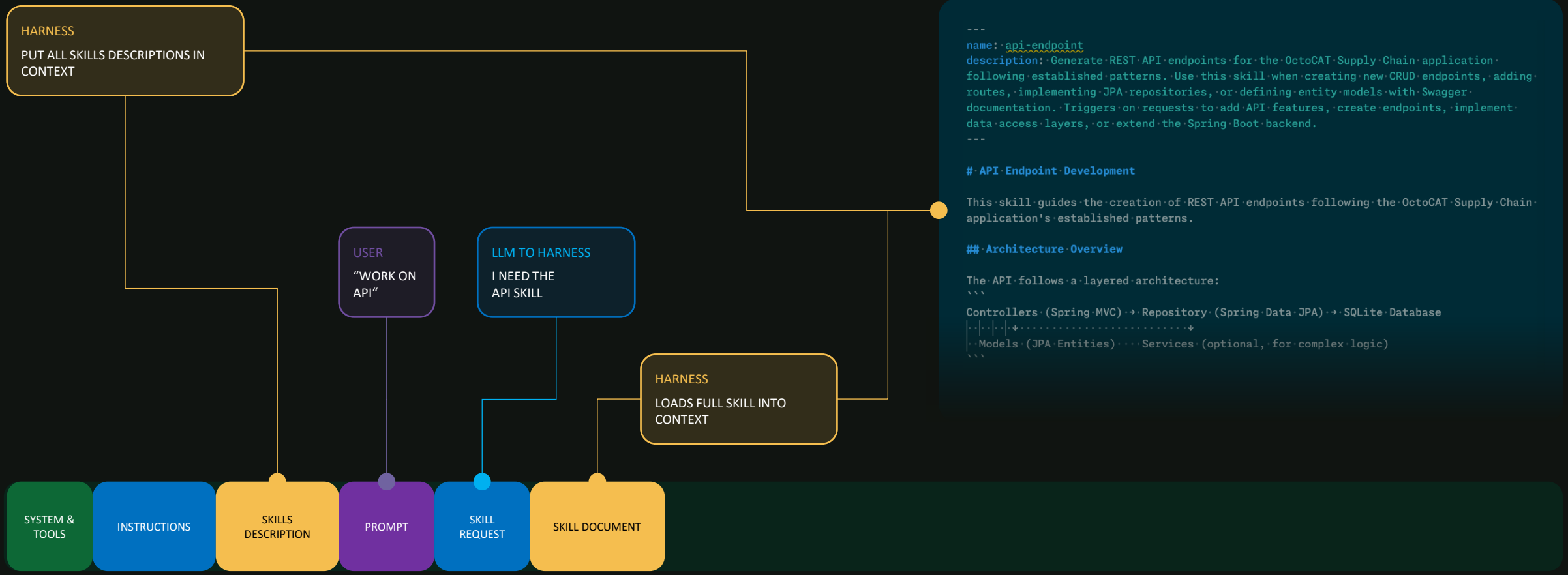
TITLE

Custom Agents



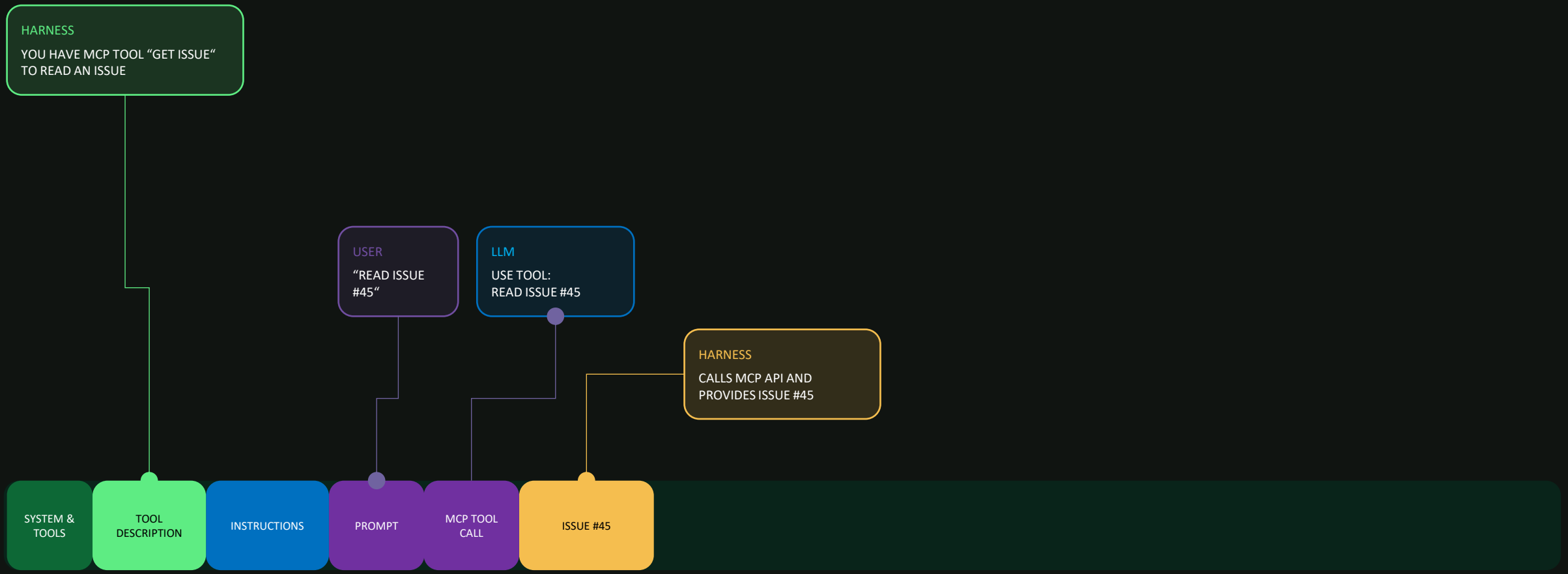
TITLE

Skills



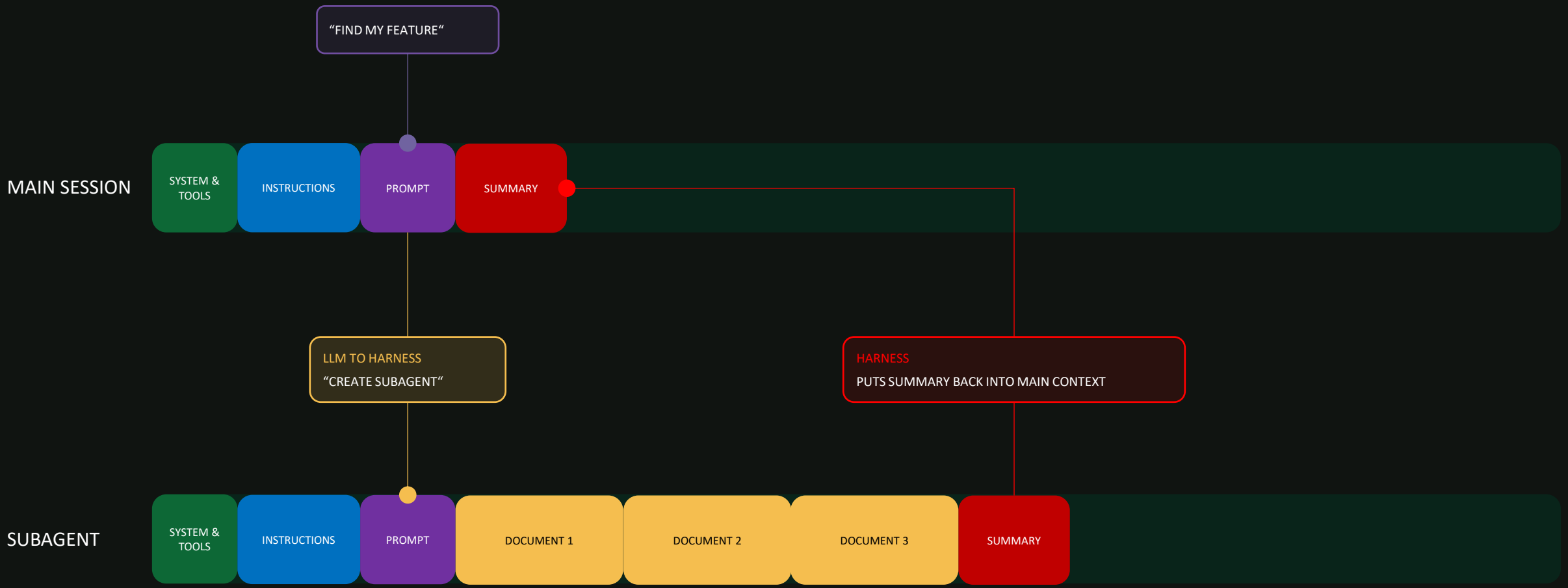
TITLE

MCPs


HARNESS
YOU HAVE MCP TOOL “GET ISSUE“ TO READ AN ISSUE
USER
“READ ISSUE #45“
LLM
USE TOOL: READ ISSUE #45
HARNESS
CALLS MCP API AND PROVIDES ISSUE #45
SYSTEM & TOOLS
TOOL DESCRIPTION
INSTRUCTIONS
PROMPT
MCP TOOL CALL
ISSUE #45

TITLE

Subagents



WHAT ELSE

Other Agent Configs

Scoped Instructions

Conditional Instructions based on file path patterns.

None-deterministic – offered to the Agent same as Skills.

Prompt Files

Manually invoked prompts.

Can be used to trim Toolset and Invoke Custom Agents.

Use them as “manual starting point” to kick of Custom agents and / or Skills with a common prompt.

Copilot Memory

Automated, small “always-on” instructions.

Used across Copilot Surfaces uniformly.

Makes sense to regularly check them.

Power User Guidance

These Tips require good knowledge and time-invest, as they are conditional and / or come with trade-offs.

Think in Code

Prefer creating scripts to analyze files over feeding it to AI.

Consider CLIs vs. MCPs

CLI Tools can be more optimal in certain scenarios due to less static tokens.

Improve Shell Outputs

Shell outputs can be extremely long – use something like https://github.com/rtk-ai/rtk to reduce their outputs.

Run “/chronicle tip” regularly

Analyze your usage in Copilot CLI to find improvement areas.

Collapse Tool Calls

Using something like https://github.com/jsturtevant/copilot-codeact-plugin can help collapse multiple tool calls into one.

Model Specific Context Optimization

Models behave different and can be tweaked.

WHAT ELSE

Other Agent Configs

Build your Analytical Skills

Coding was never the true value of developers – it was their analytical skills and the ability to become proficient in any domain quickly.

Being able to tell an Agent precisely what to do, in the speak of the domain, will become the most valuable.

Apply Good Architecture

Domain Driven Design, Hexagonal Architecture, CQRS, Event Driven Design…

Allows easier agent discovery.

Gives stronger guardrails, avoids excess agent session (fixing, debugging, errors etc.).

Iterate on Prompts & Agent Configs

Approach AI Agents with an engineering mind.

Keep configs fresh, treat agent misses like incidents.

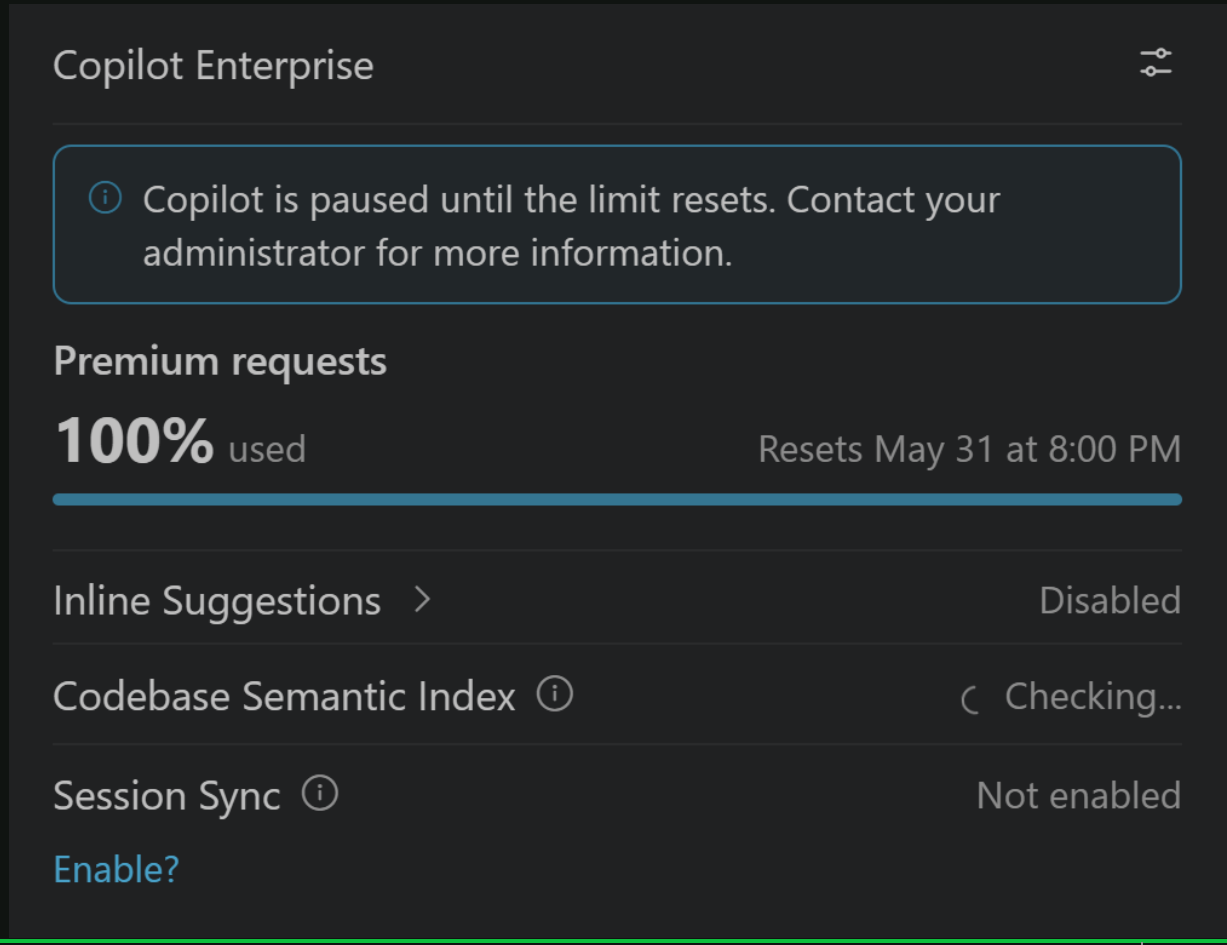
Use `/chronicle` in the CLI regularly to understand improvement areas.

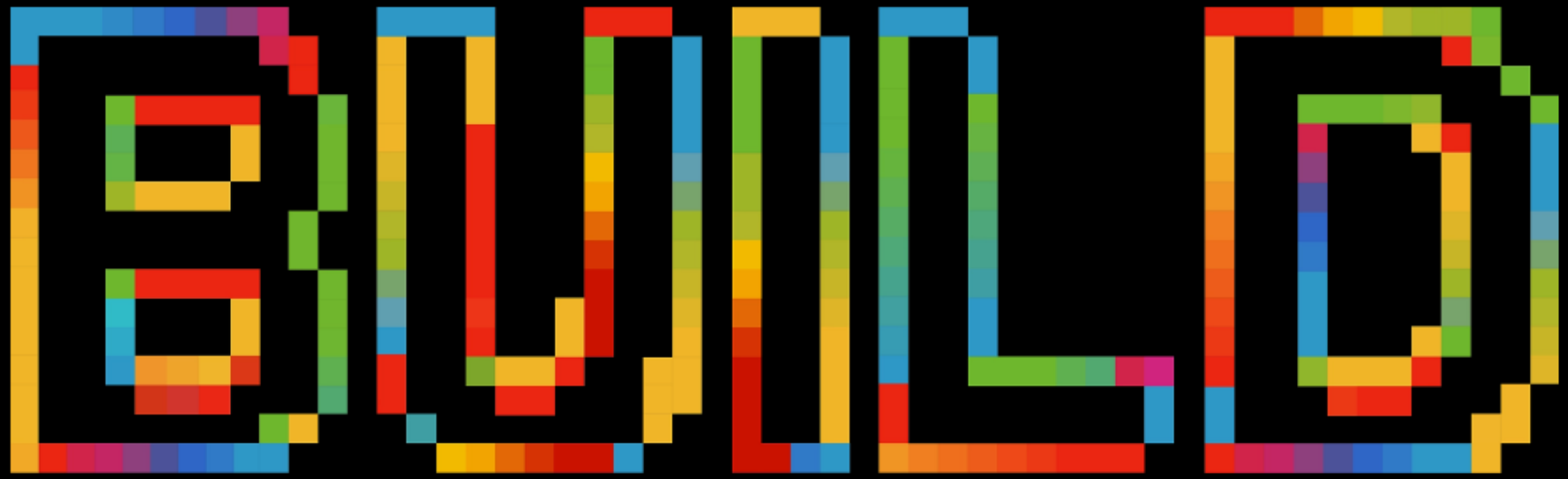
SUMMARY

5 things you can start doing today

- Chose the right model for the right task. Lean on Auto-model.
- Provide clear guidance in your prompts.
- Research – Plan – Implement.
- Provide deterministic guardrails (tests, linters, security scans etc.).
- Maintain a concise, human-written copilot-instructions.md. Use it as agent-miss log & to trim outputs.

Quick UI Review





Now shipping: Microsoft Build 2026

June 2–3, 2026 / San Francisco and online

At Microsoft Build, you'll go deep on real code, real systems, and real workflows with the teams building and scaling AI. Two days. Hands-on sessions. No fluff.

Get started

Thank you

haslam93/GitHub-Copilot---Usage-Based-Billing-Info-Tips