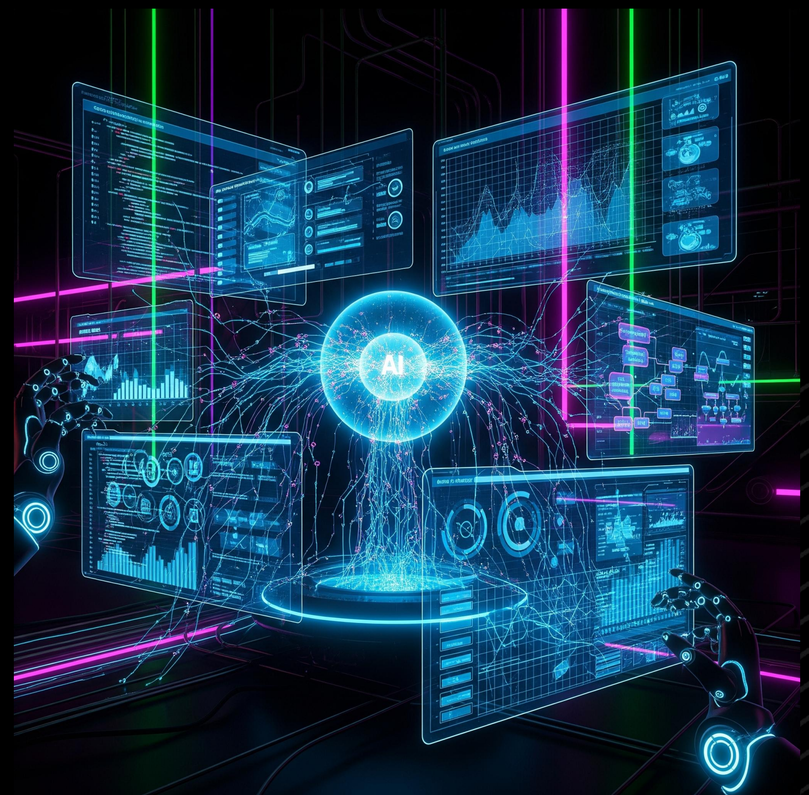
AI Agents Unbound: Scaling up Code Quality, Reliability, Product Iteration

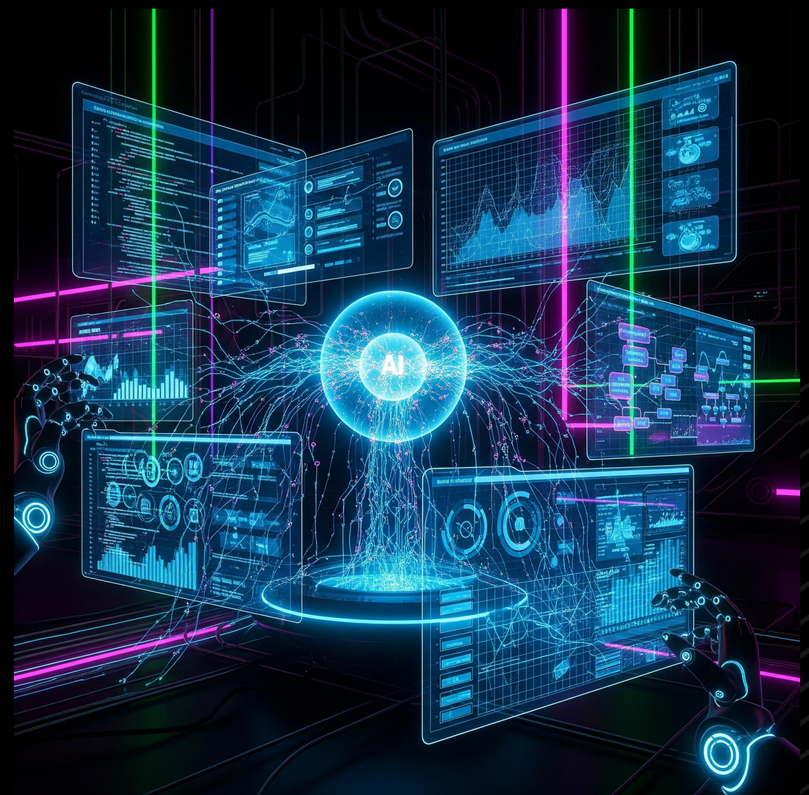


Jubin Chheda

Principal Engineer, Facebook

Title on the Advert

Refactoring at Scale: LLM-Powered Pipelines for Detecting Code Smells, Hardening Tests, and Modernizing Legacy C++



Jubin Chheda

Principal Engineer, Facebook

Titles that did not make the cut

Neon Agents: Scaling Code in the Machine City

Codepunk 2077: AI Agents and the Refactor Wars

Ghosts in the Code: AI Agents Unbound

Neural Overdrive: Scaling Code Quality in the Age of Agents

The Refactor Nexus: AI Agents at Meta Scale

Dark Code, Bright Agents

Megacorp Systems: AI Agents and the Reliability Paradox

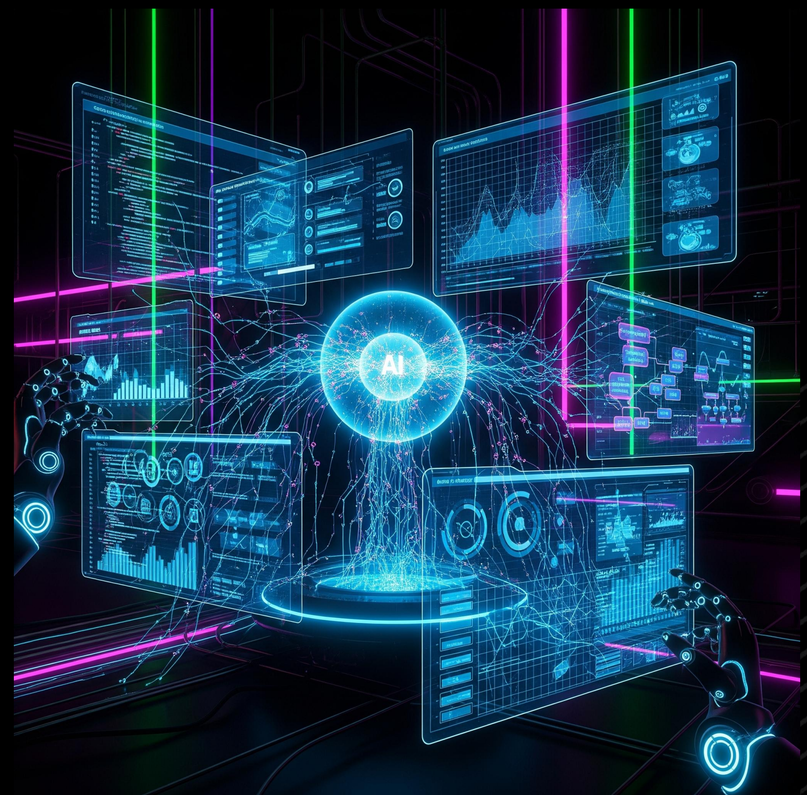
Synthetic Coders: Rise of the Agentic Flow

The Code Sprawl: AI Agents Hacking Scale

Blade of Code: Craft and Chaos in the Age of AI Agents

Jubin Chheda

Principal Engineer, Facebook



Speaker Profile: Jubin Chheda - Intro

Principal Engineer at Meta

Tech Lead for Meta AI on Facebook

[Products like Feed Deep Dive, Write with AI]

Lead, Facebook App’s AI-Native program

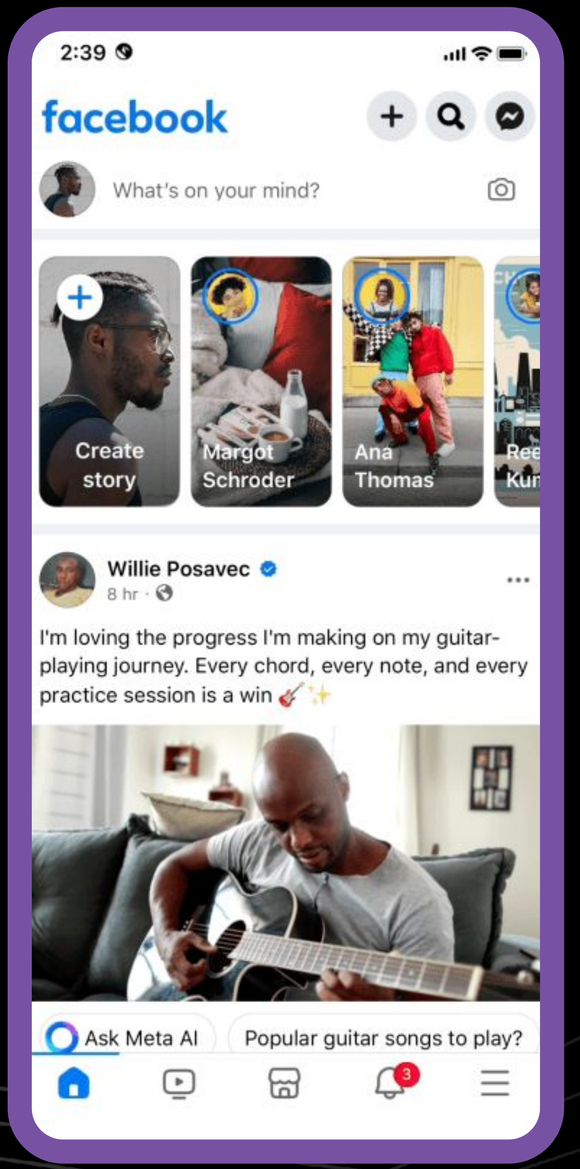
- 0>1 Prototyping
- Refactoring
- Experimentation
- Reliability
- Craft

Speaker Profile: Lead Engineer, Meta AI on Facebook

Meta AI @ 1 Billion+ Monthly Active Users
Meta AI on Facebook drives significant traffic

Key Technical features

- Scale
  - Every eligible post on Facebook
  - 1-1.2s latency
- Reasoning Models & Agentic flows
- Multimodal Understanding - Videos, images - memes, satire - sports, TV/movies, howtos
- Product Market Fit



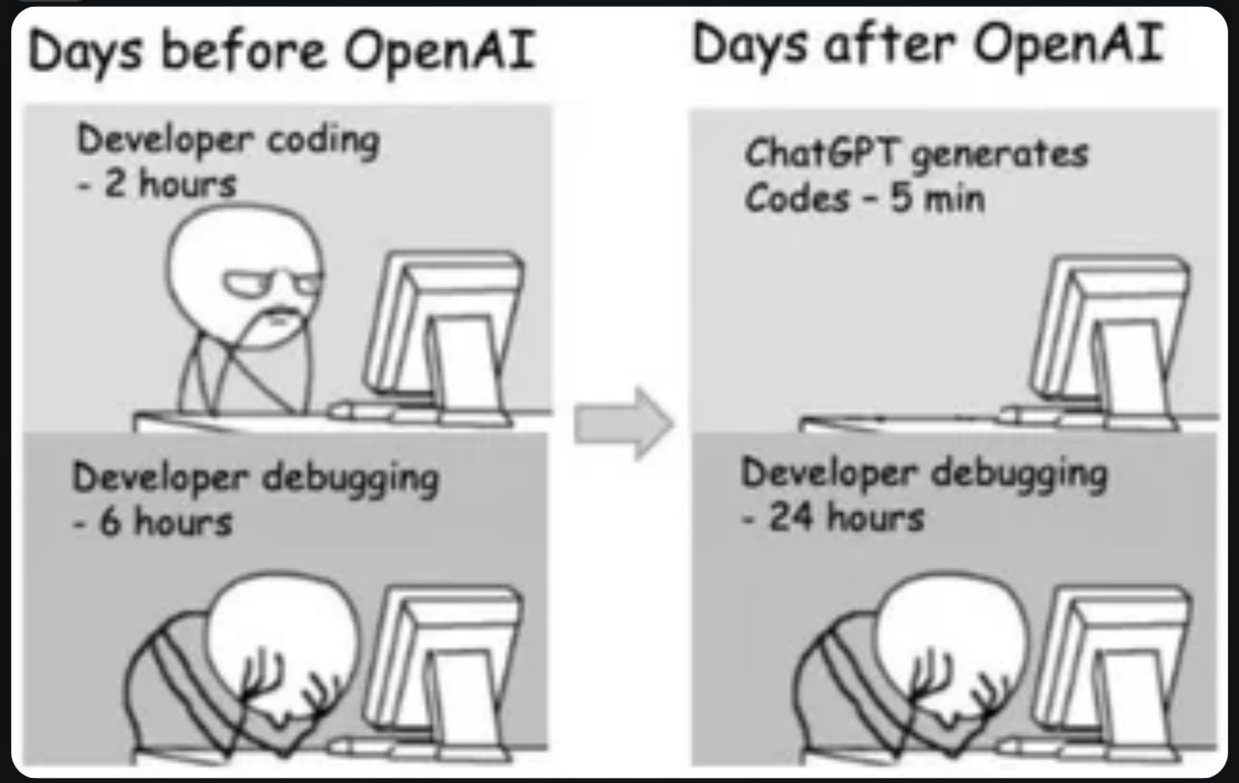
META Q1 2025 Earnings Call Transcript

What you can expect from this talk

r/ProgrammerHumor • 3 yr. ago
Dubabear

Ai wIlL rEpLaCe Us

Meme

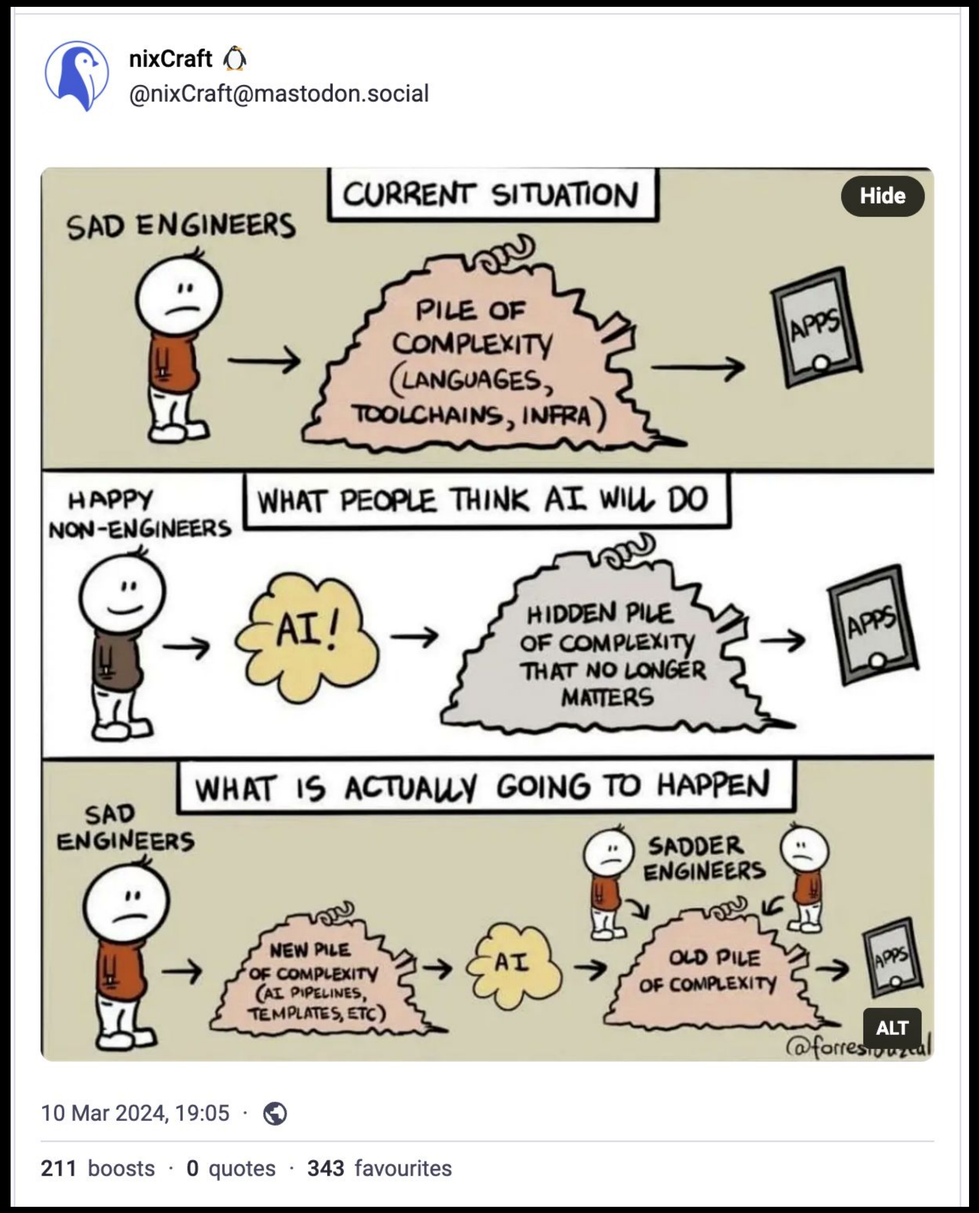


Archived post. New comments cannot be posted and votes cannot be cast.

23K 393 Share

Ai wIlL rEpLaCe Us : r/ProgrammerHumor

What you can expect from this talk



nixCraft 🐧: "The cartoon suggests that AI w..." - Mastodon

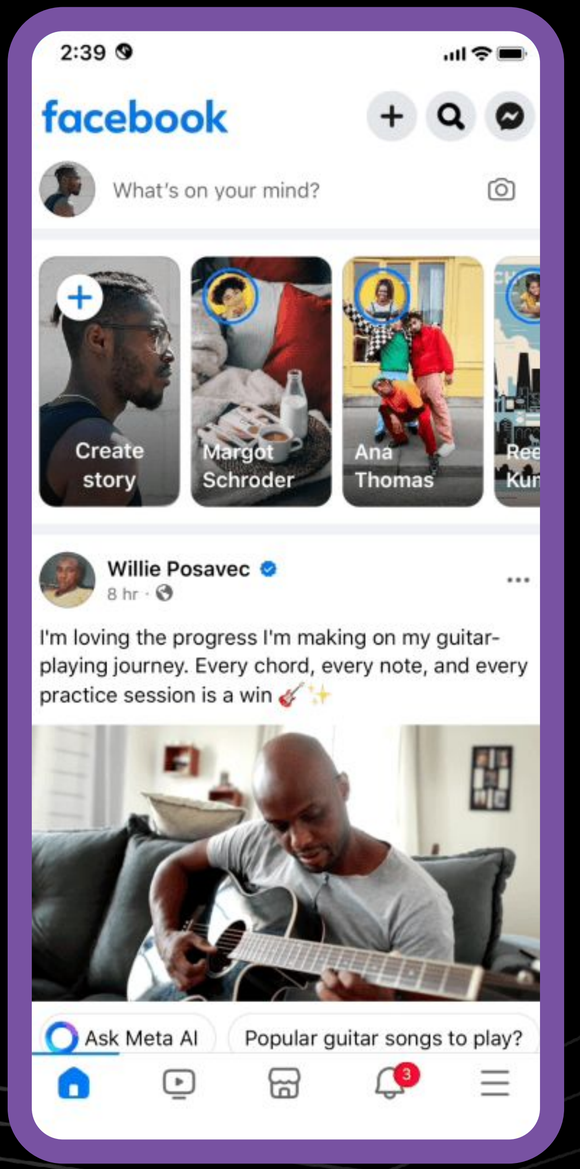
What you can expect from this talk

- Understand Internals of AI agents
- Actually Get more tasks done successfully with agents
- Improve Code Quality at scale

Speaker Profile: Lead, AI-Native Pods

- AI-forward: AI tools for data analysis, coding, experimentation
- Improve product throughput & velocity
- Speeding up 0>1
- Drive more value from already launched products
  - Model driven targeting, ranking, recs
  - Craft improvements

All @Meta Scale



Agenda

- Ch1: In a galaxy not far far away…
  Challenges of Scale & Need for focus on Code Quality
- Ch2: Karpathy Canon
  Rise of the Vibe Coder
- Ch3: Ex-Machina
  The Anatomy of an agent
- Ch4: Prometheus Unbound
  Harnessing AI Agents
- Ch5: The Agent Trilogy
  Cases from the Field

Agenda

- Ch1: In a galaxy not far far away…
  Challenges of Scale & Need for focus on Code Quality
- Ch2: Karpathy Canon
  Rise of the Vibe Coder
- Ch3: Ex-Machina
  The Anatomy of an agent
- Ch4: Prometheus Unbound
  Harnessing AI Agents
- Ch5: The Agent Trilogy
  Cases from the Field

Ch1: In a galaxy not far far away...
Challenge of Scale & Need for focus on Code Quality

The Challenge of Scale:

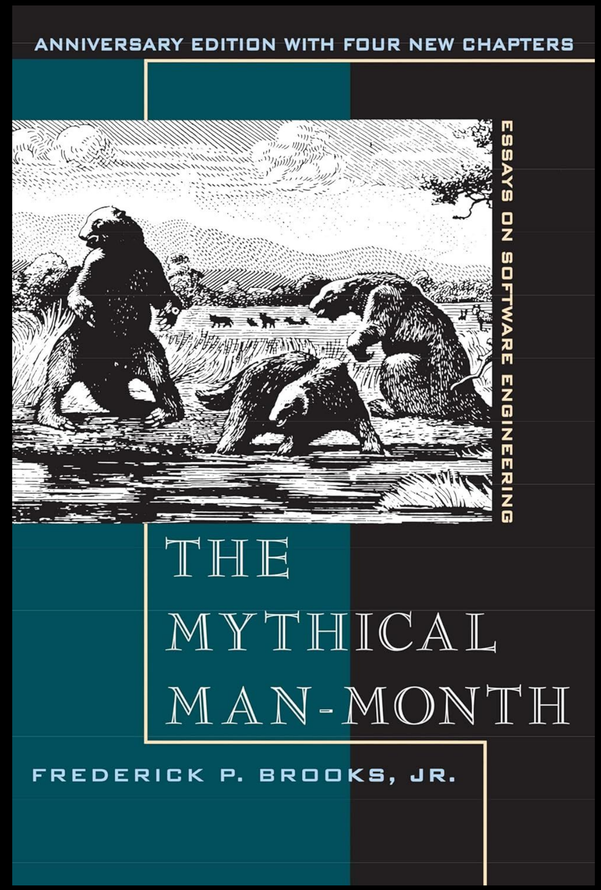
How do you maintain code quality at this scale?

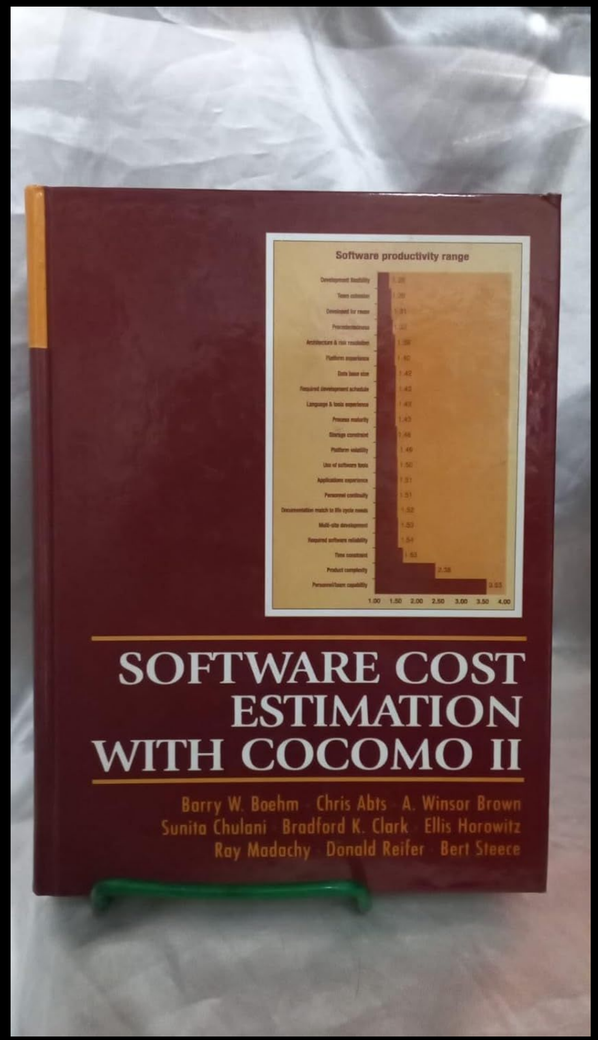
- **LOC: 100M+**
- Developers: 10K**+**
- Commits/day: 10K+
- Codebase age: 15+ years
- Domains:
  - Backend services
  - ML infrastructure
  - Mobile
  - AI
  - Product Infrastructure

Mockus 2025 Code Improvement Practices at Meta

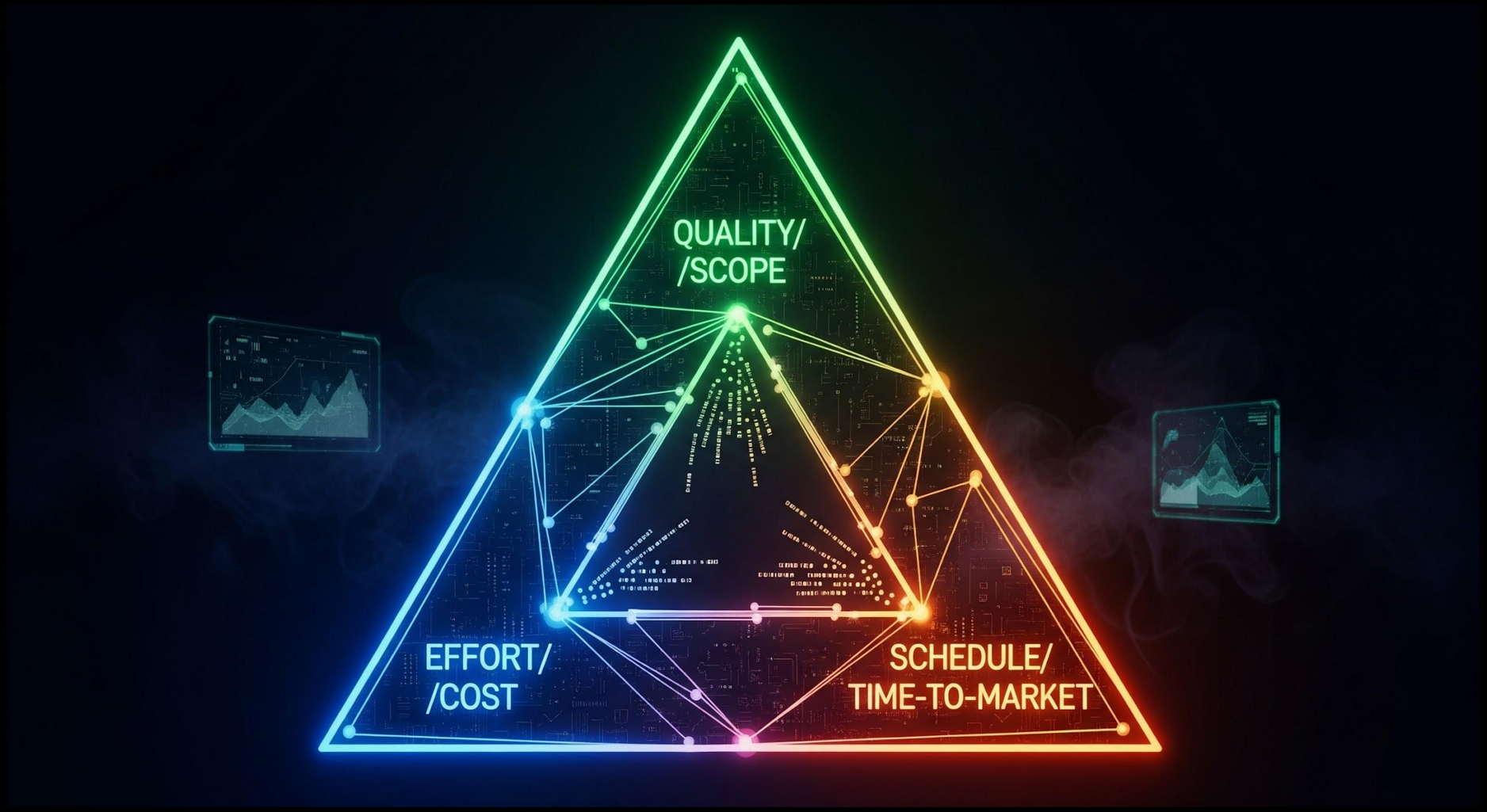
Why Code Quality Matters - The Classics

- Barry Boehm – Software Engineering Economics (1981) COCOMO
- Frederick P. Brooks – The Mythical Man-Month (1975)
- Watts Humphrey – Managing the Software Process (1989)
- Steve McConnell – Rapid Development (1996)





Why Code Quality Matters - Software Engineering Iron Triangle



Why Code Quality Matters - Outages & Eng Velocity

*“The fundamental challenge in software development quality improvement efforts is the **inverse relationship** between the delivery of software changes (referred to as diffs at Meta) that are necessary for enhancements or fixes to existing problems and outages (SEVs) these changes introduce. More numerous, especially rapidly made, changes lead to more problems on one hand, but fixes and enhancements are essential to keep users satisfied. Thus, the need for any software business is to deliver cutting-edge enhancements without significant deterioration in software reliability. The simplest (and obvious) solution to avoid outages is to keep the software codebase unchanged. Unfortunately, this approach conflicts with the need for rapid delivery of new functionality and new products.”*

- Leveraging Risk Models to Improve Productivity for Effective Code Un-Freeze at Scale Audris Mockus et al (Meta Platforms), 2025

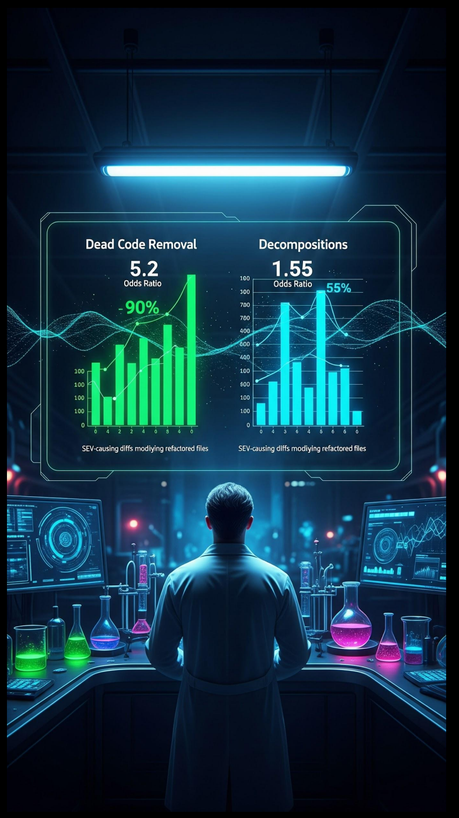
Why Code Quality Matters - Meta Study

“The odds ratio is 5.2 for dead code removal and 1.55 for decompositions showing 90% and 55% decrease in SEV-causing diffs modifying refactored files.”

TABLE VI: The odds ratio is 5.2 for dead code removal and 1.55 for decompositions showing 90% and 55% decrease in SEV-causing diffs modifying refactored files. The results are not statistically significant for platformization and large class decompositions. Instead of raw numbers, proportions of all (pre- and post-reengineering) diffs and triggers are shown

Type	Period	Proportion of Diffs (no Sev) during the period	Proportion of SEV-triggering Diffs
Dead code removal	**Before**	57%	76%
	After	43%	24%
CCN-driven decomp.	**Before**	72%	80%
	After	28%	20%

Substantial improvements in quality have been previously documented in other studies of reengineering, e.g., [6]. Often this is a result of a more transparent codebase where it is easier not to overlook some unanticipated effects of a code change.



Mockus 2025 Code Improvement Practices at Meta

Why Code Quality Matters - Data

“Code Quality Improvements decreased DAT - Diff Authoring Time ”

TABLE VII: The median DAT. The decrease is statistically significant (Mann-Whitney two sample test)

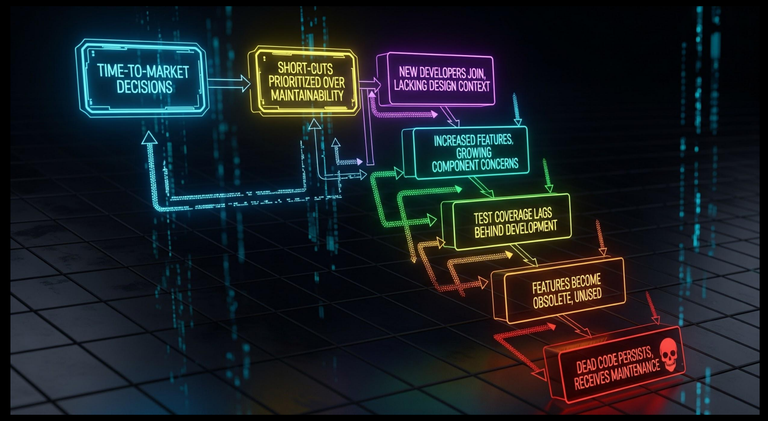
Type	Relative change
Dead code removal	0.59
CCN-driven decomp.	0.23
Large class decomposition	0.41
Platformization	0.52

To answer H2, a simpler codebase should also make it easier and more straightforward to write the code. Indeed, Table VII shows 41% to 77% reduction in DAT for all types of refactoring. We expect that this shorter authoring time will require fewer code editing sessions (and fewer time-consuming task context switches [37]).

Mockus 2025 Code Improvement Practices at Meta

Why Code Quality Matters - Slippery Slope to Code Complexity

- Time-to-market decisions lead to short-cuts for delivery over maintainability
- New developers joining lacking design decision context
- More features and more concerns on existing components
- Test coverage may not keep pace
- Features go out of use, yet dead code survives and even receives maintenance



Agenda

- Ch1: In a galaxy not far far away…
  Challenges of Scale & Need for focus on Code Quality
- Ch2: Karpathy Canon
  Rise of the Vibe Coder
- Ch3: Ex-Machina
  The Anatomy of an agent
- Ch4: Prometheus Unbound
  Harnessing AI Agents
- Ch5: The Agent Trilogy
  Cases from the Field


Ch2:
KARPATHY CANON
&
THE Rise OF THE Vibe Coder

Karpathy Canon - what is it?

Idealised philosophy of programming being seeded by an influential figure

- Literate Programming - Knuth 1992
- Egoless Programming - Weinberg 1971
- Karpathy Canon is only *185-word* tome

Andrej Karpathy
@karpathy

There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

6:17 PM · Feb 2, 2025 · **5.1M** Views

1.3K 5.2K 30K 15K

Andrej Karpathy on X: "There's a new kind of coding I call "vibe coding" / X

Andrej Karpathy
@karpathy

The hottest new programming language is English

3:14 PM · Jan 24, 2023 · **8M** Views

1.3K 7.6K 48K 4.6K

Andrej Karpathy on X: "The hottest new programming language is English" / X

Vibes - what could go wrong?

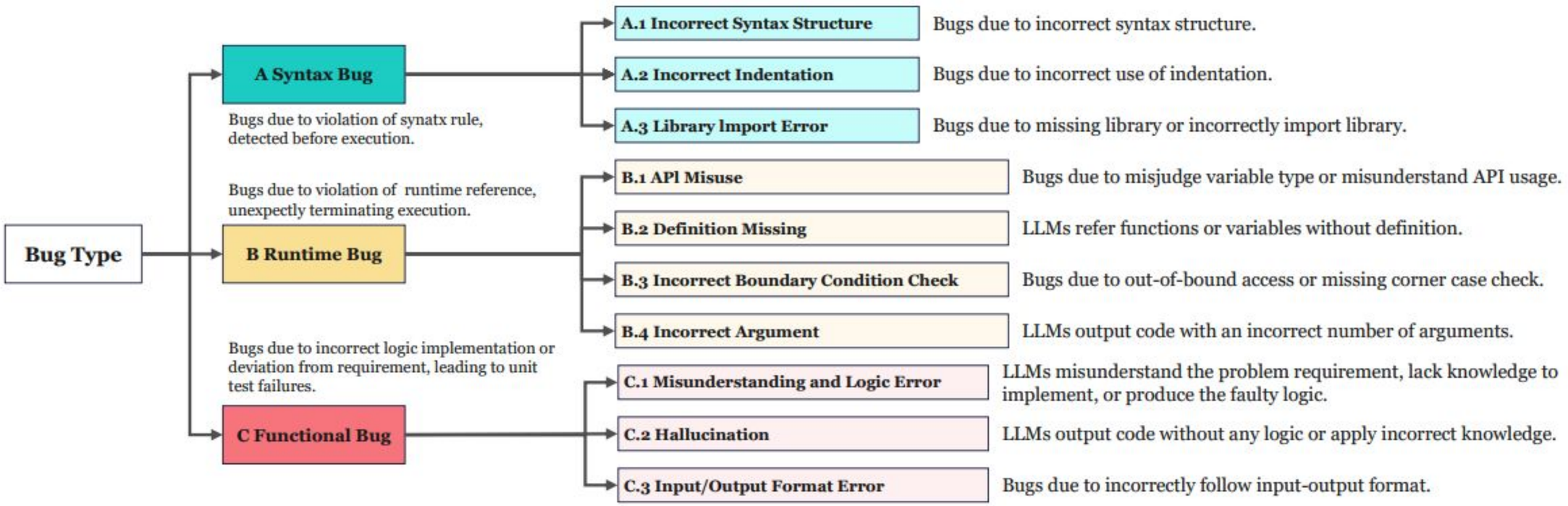


Figure 3: Taxonomy of bugs that occurred in code generated by LLMs.

Dou 2024 What's Wrong with Your Code Generated by Large Language Models? An Extensive Study

Vibes - what could go wrong?

Table 3: Types of bugs introduced during code translation by seven popular LLMs on three widely used benchmarks (*i.e.*, HumanEval+, MBPP+, and APPS+). SC2 denotes StarCoder-2, DC denotes DeepSeekCoder, LL3 denotes Llama-3, and CL3 denotes Claude-3. All values are in %.

Bug Types	HumanEval Plus							MBPP Plus							APPS Plus						
	Open-Source				Closed-Source			Open-Source				Closed-Source			Open-Source				Closed-Source		
	SC2	DC	LL3	Phi3	GPT4	GPT3.5	CL3	SC2	DC	LL3	Phi3	GPT4	GPT3.5	CL3	SC2	DC	LL3	Phi3	GPT4	GPT3.5	CL3
A.1 Incomplete Syntax Structure	0.0	0.0	0.0	0.0	0.0	0.6	0.0	0.3	0.5	2.4	1.9	0.0	0.5	0.0	0.3	3.5	2.7	0.8	0.2	1.5	0.2
A.2 Incorrect Indentation	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	1.2	0.0
A.3 Library Import Error	3.7	2.4	10.4	0.0	0.6	2.4	0.0	0.3	0.0	0.8	0.0	0.3	0.5	0.0	0.0	0.0	0.5	0.2	1.0	2.0	2.5
A Syntax Bug	3.7	2.4	10.4	0.0	0.6	3.0	0.0	0.5	0.5	3.2	1.9	0.3	1.0	0.0	0.3	3.7	3.2	1.0	1.2	4.7	2.7
B.1 API Misuse	1.2	1.2	1.8	0.0	0.0	2.4	0.6	2.9	2.1	1.3	2.4	1.0	1.8	1.3	1.8	4.7	6.5	7.8	0.5	1.3	1.0
B.2 Definition Missing	0.0	2.4	0.0	0.6	0.0	5.5	0.0	0.8	0.3	0.0	0.0	0.0	0.0	0.0	1.0	1.2	2.3	3.3	0.5	2.2	1.8
B.3 Incorrect Boundary Condition Check	0.6	2.4	0.6	0.0	0.0	3.0	0.0	2.4	1.3	2.1	1.9	1.5	0.5	1.5	1.8	4.2	5.2	5.7	2.8	4.3	1.7
B.4 Incorrect Argument	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.8	0.0	0.5	1.0	0.0	0.3	37.7	0.7	0.2	2.8	1.5	0.3	1.0
B.5 Minors	3.7	1.8	1.2	1.2	0.6	0.6	0.6	1.1	0.8	0.5	0.3	0.5	0.0	2.3	1.0	1.8	2.7	1.3	1.8	1.7	2.2
B Runtime Bug	5.5	7.9	3.7	1.8	0.6	11.6	1.2	7.1	5.3	4.0	5.0	4.0	2.3	5.3	43.3	12.5	16.8	21.0	7.2	9.8	7.7
C.1 Misunderstanding and Logic Error	29.3	29.9	34.1	31.1	12.8	20.7	17.1	19.0	18.8	19.8	20.9	12.0	15.5	13.8	26.7	48.3	44.5	65.7	31.3	46.5	37.5
C.2 Hallucination	0.0	1.2	0.0	0.0	0.6	0.0	1.2	7.4	7.7	10.3	8.2	1.8	5.8	3.0	5.3	7.5	10.0	0.5	7.0	3.0	4.5
C.3 Input/Output Format Error	0.0	0.0	0.0	0.0	0.0	0.0	1.2	2.6	1.3	1.1	0.5	2.8	0.8	2.8	3.7	2.8	1.2	2.3	0.3	2.7	1.7
C.4 Minors	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.3	0.0	0.8	0.3	1.7	2.0	2.7	0.7	0.7	2.7	2.0
C Functional Bug	29.3	31.1	34.1	31.1	13.4	20.7	19.5	29.1	27.8	31.7	29.9	16.5	22.8	19.8	37.3	60.7	58.3	69.2	39.3	54.8	45.7
D Ambiguous Problem	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.8	0.8	0.8	0.8	0.8	0.8	0.8

Dou 2024 What's Wrong with Your Code Generated by Large Language Models? An Extensive Study

Vibe Coding: what are most people using it for?

Mostly for 0>1 prototypes

Lots of iterative development

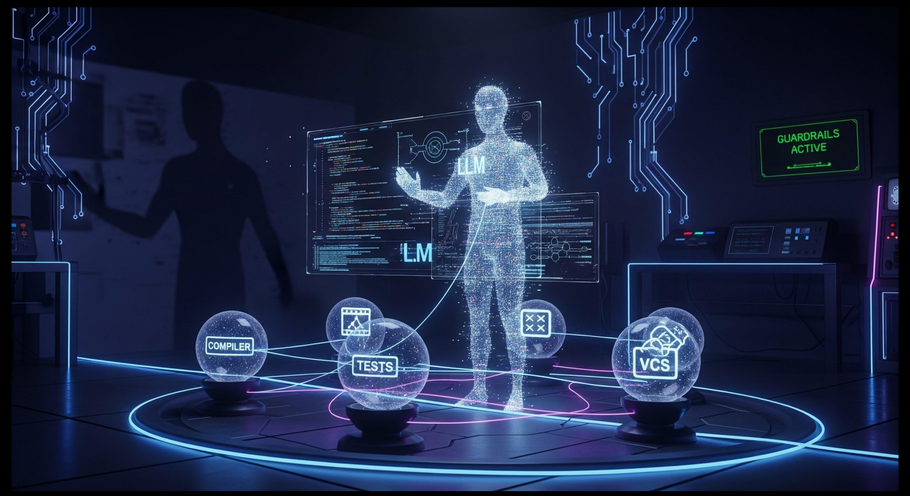
Getting a feel for the problem

Dou 2024 What's Wrong with Your Code Generated by Large Language Models? An Extensive Study

Another AI-driven SE paradigm: Agentic AI

Agentic AI–driven development: autonomous or semi-autonomous LLM agents that plan, decompose tasks, call tools (compiler, tests, file search, etc), and iteratively ship changes with monitoring/guardrails

Unlike conventional code generation tools, agentic systems are capable of decomposing high-level goals, coordinating multi-step processes, and adapting their behavior based on intermediate feedback.

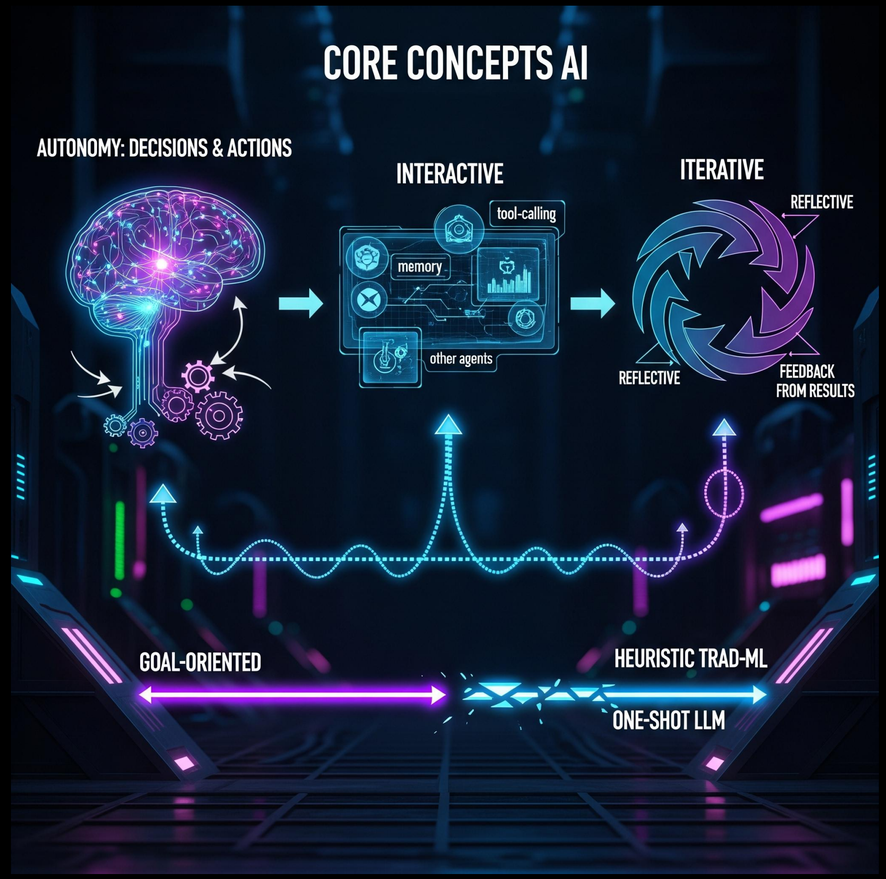


AI Agents & how they differ

- Autonomy: Decisions & Actions
- Interactive: tool-calling, memory, other agents
- Iterative: reflective, feedback from results
- Goal-oriented

Different than

- Heuristic
- Trad-ML
- One-shot LLM



e.g. “Implement a REST API endpoint that returns the top 10 most frequently accessed URLs from a web server log file. Include unit tests and documentation.”
-> This task requires integrating multiple software components, including file parsing, frequency analysis, web API implementation, testing, and documentation.

Dong 2025 A Survey on Code Generation with LLM-based Agents
Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

AI agents and code quality

- Amount of code written scales up dramatically

Google — Pichai said “more than 30%” of new code and ~10% productivity gain (Jun 2025 Business Insider).

Microsoft CEO says up to 30% of the company’s code was written by AI (Apr 2025 TechCrunch)

Image credits: Brian Smale, European Union

AI agents and code quality

- AI agents perform better if code is less complex and or fewer quality issues

Metric	Resolved Mean	Unresolved Mean	Mann-Whitney U Test p-value	Significance
1. Problem Statements				
Flesch_Reading_Ease	35.63	38.91	0.2425	✗
Flesch_Kincaid_Grade	11.40	10.81	0.2219	✗
Sentence_Count	27.46	37.22	0.0175	✓
Word_Count	178.72	209.86	0.0015	✓
Contains_Code_Snippets	0.45	0.48	0.5130	✗
Number_of_Code_Blocks	1.08	1.18	0.5295	✗
Lines_of_Code	14.44	18.56	0.47	✗
Code_to_Text_Ratio	0.24	0.24	0.7960	✗
2. Associated Source Code Files (Gold Patch Based)				
Code_Files_Count	1.09	1.53	3.53e-11	✓
Lines_of_Code	703.13	1087.38	0.0062	✓
Code__Cyclomatic_Complexity	192.25	303.12	0.0067	✓
3. Gold Patch Solutions				
Total_Lines_Change	9.29	24.12	1.67e-09	✓
Net_Code_Size_Change	3.22	10.08	9.18e-06	✓

Chen 2024 Evaluating Software Development Agents: Patch Patterns, Code Quality, and Issue Complexity in Real-World GitHub Scenarios

AI agents and code quality

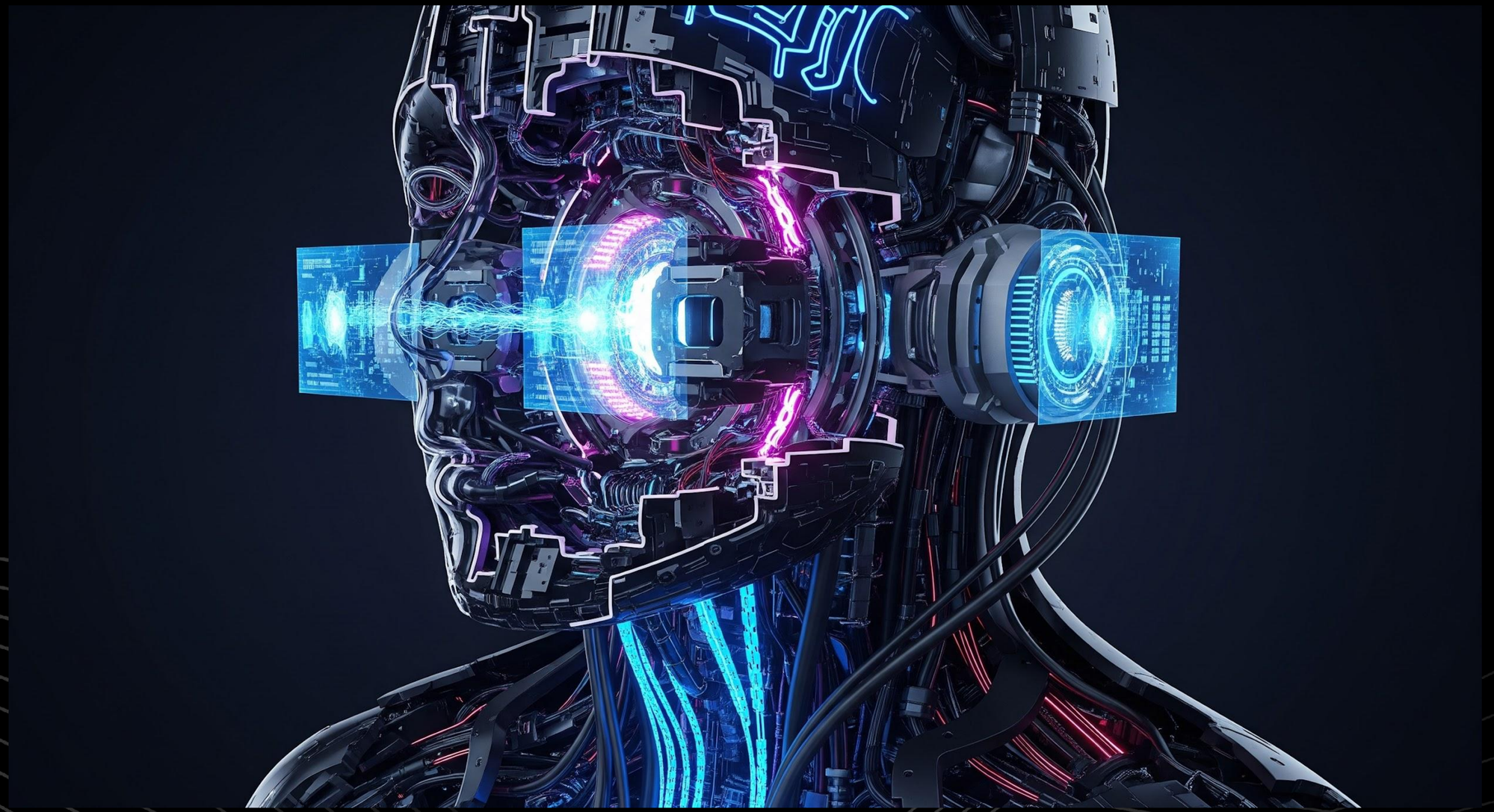
- AI agents can help bring code quality and complexity under control

We will focus on 2 main topics
. Leveraging AI agents for feature development
. AI agents for improving code quality, but first...

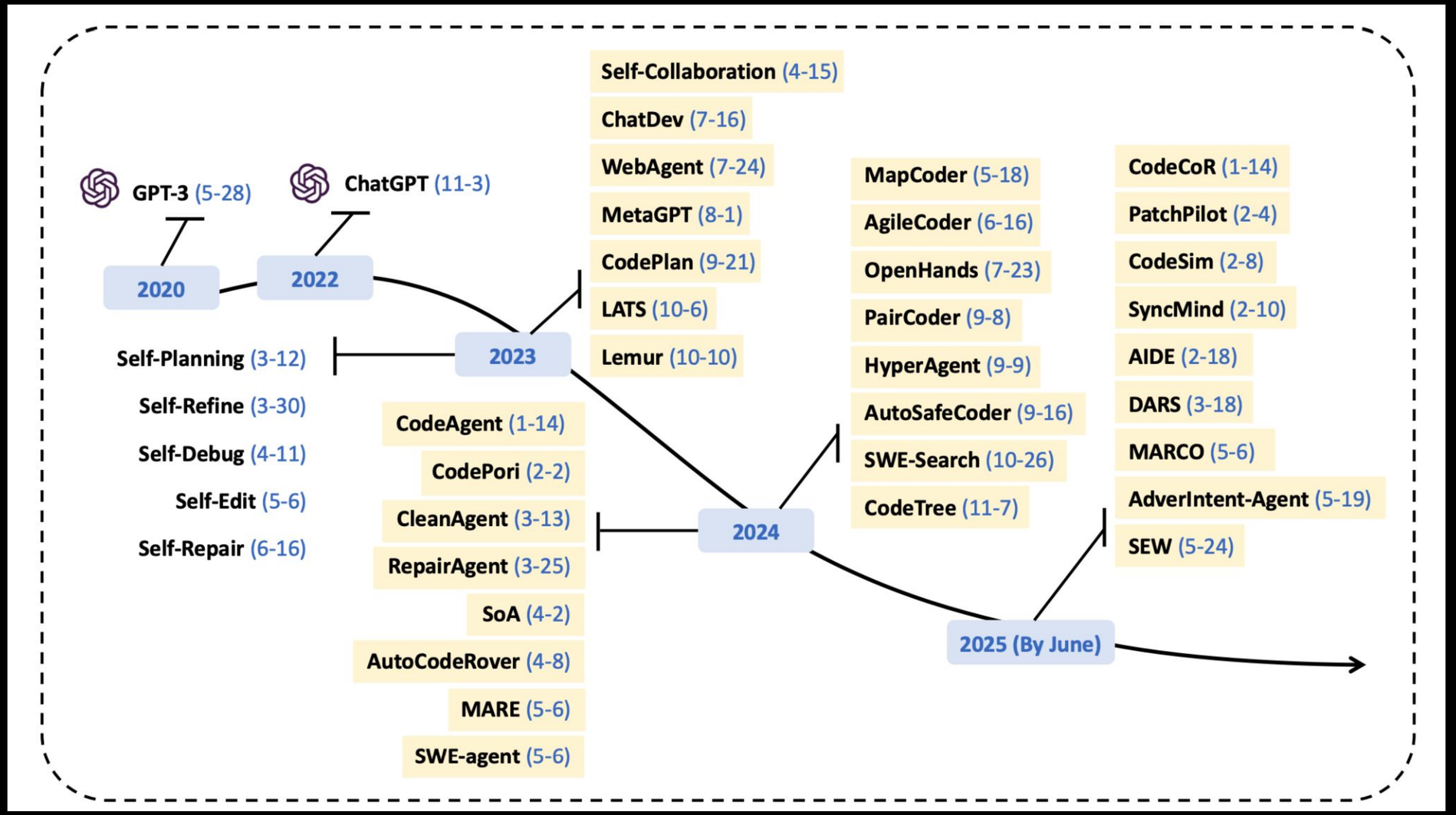
Agenda

- Ch1: In a galaxy not far far away…
  Challenges of Scale & Need for focus on Code Quality
- Ch2: Karpathy Canon
  Rise of the Vibe Coder
- Ch3: Ex-Machina
  The Anatomy of an agent
- Ch4: Prometheus Unbound
  Harnessing AI Agents
- Ch5: The Agent Trilogy
  Cases from the Field

Ch3: Ex-machina:
The anatomy of an AI Agent

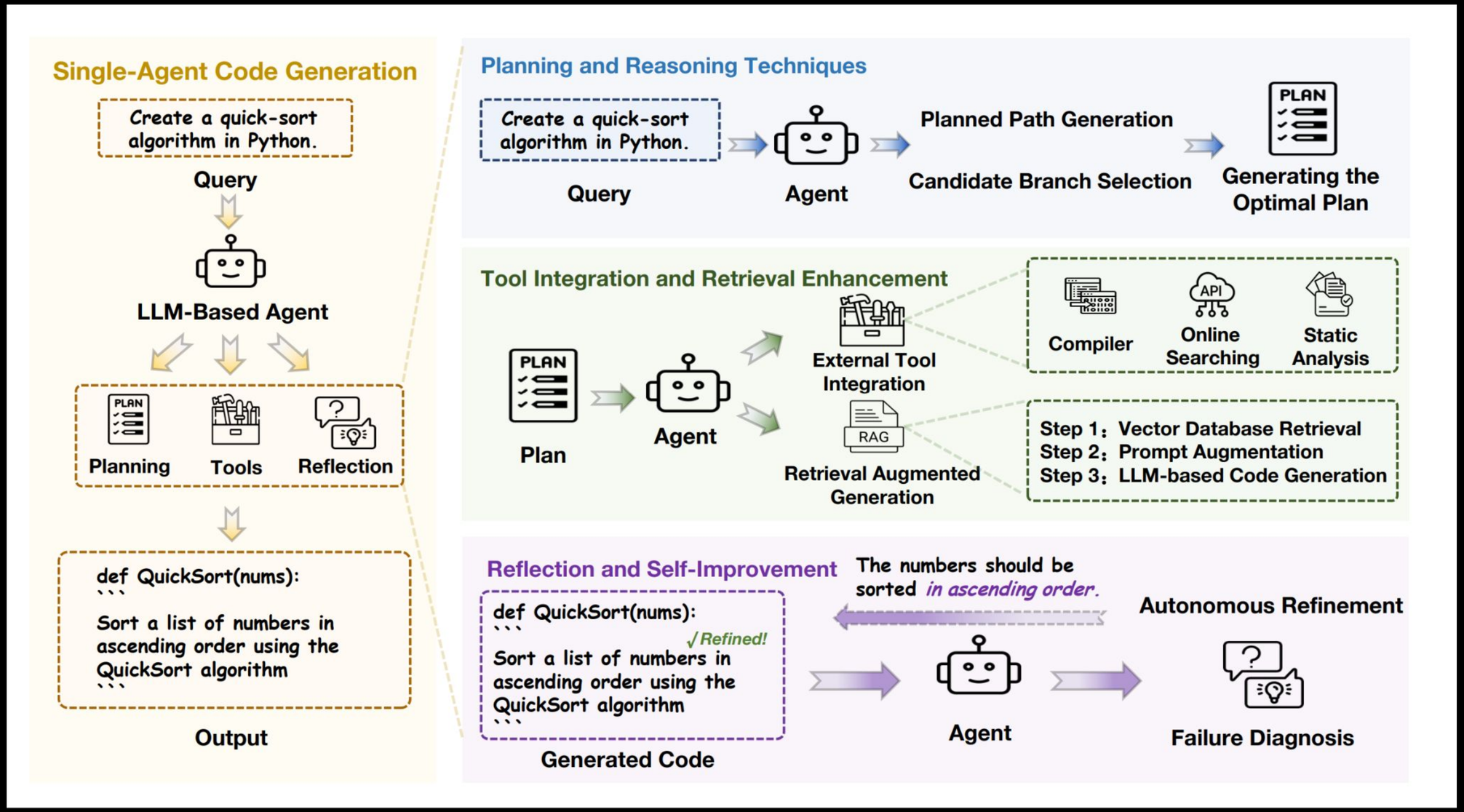


Evolution of work that powers coding AI agents



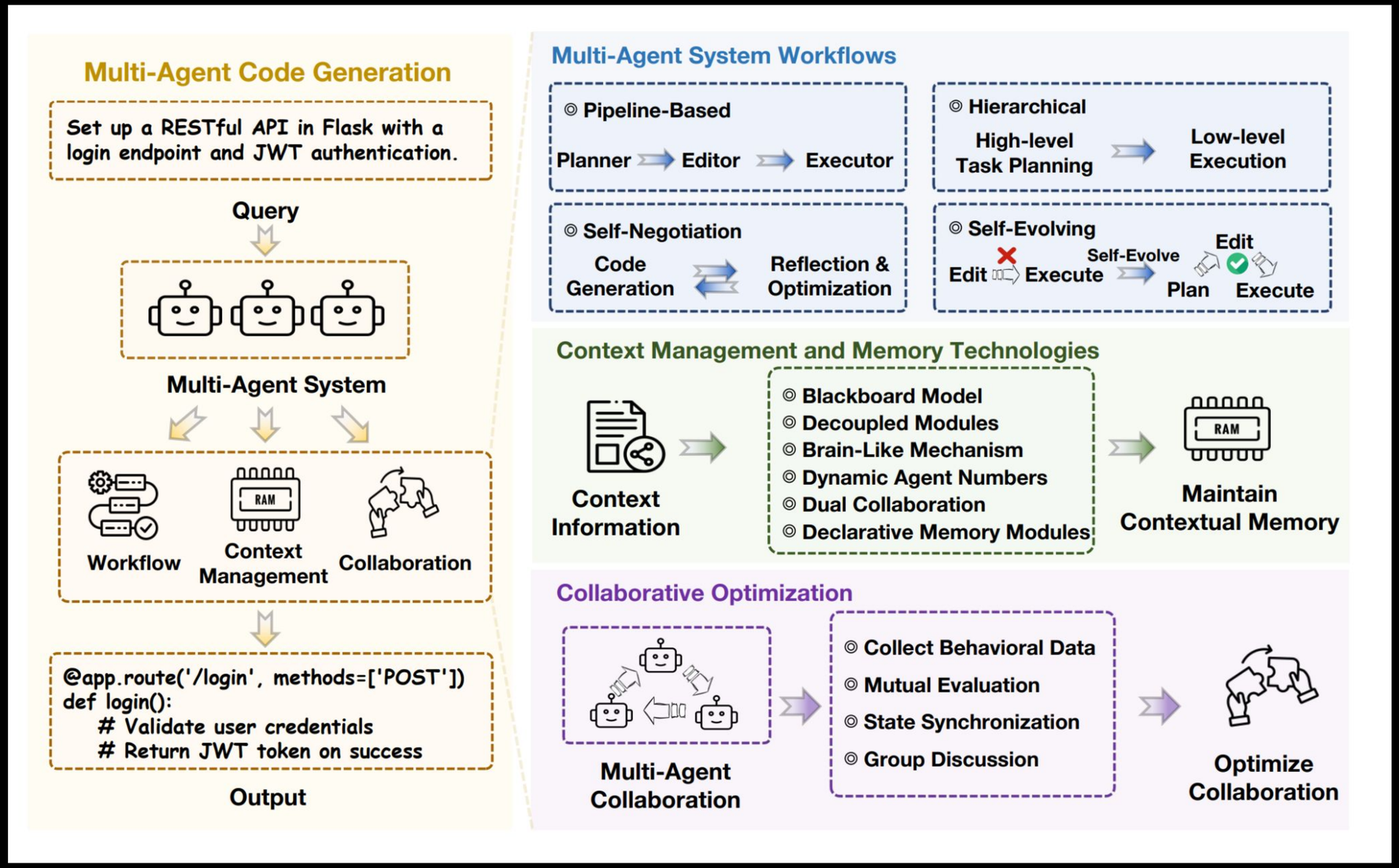
Dong 2025 A Survey on Code Generation with LLM-based Agents
Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

Single Agent Code Generation



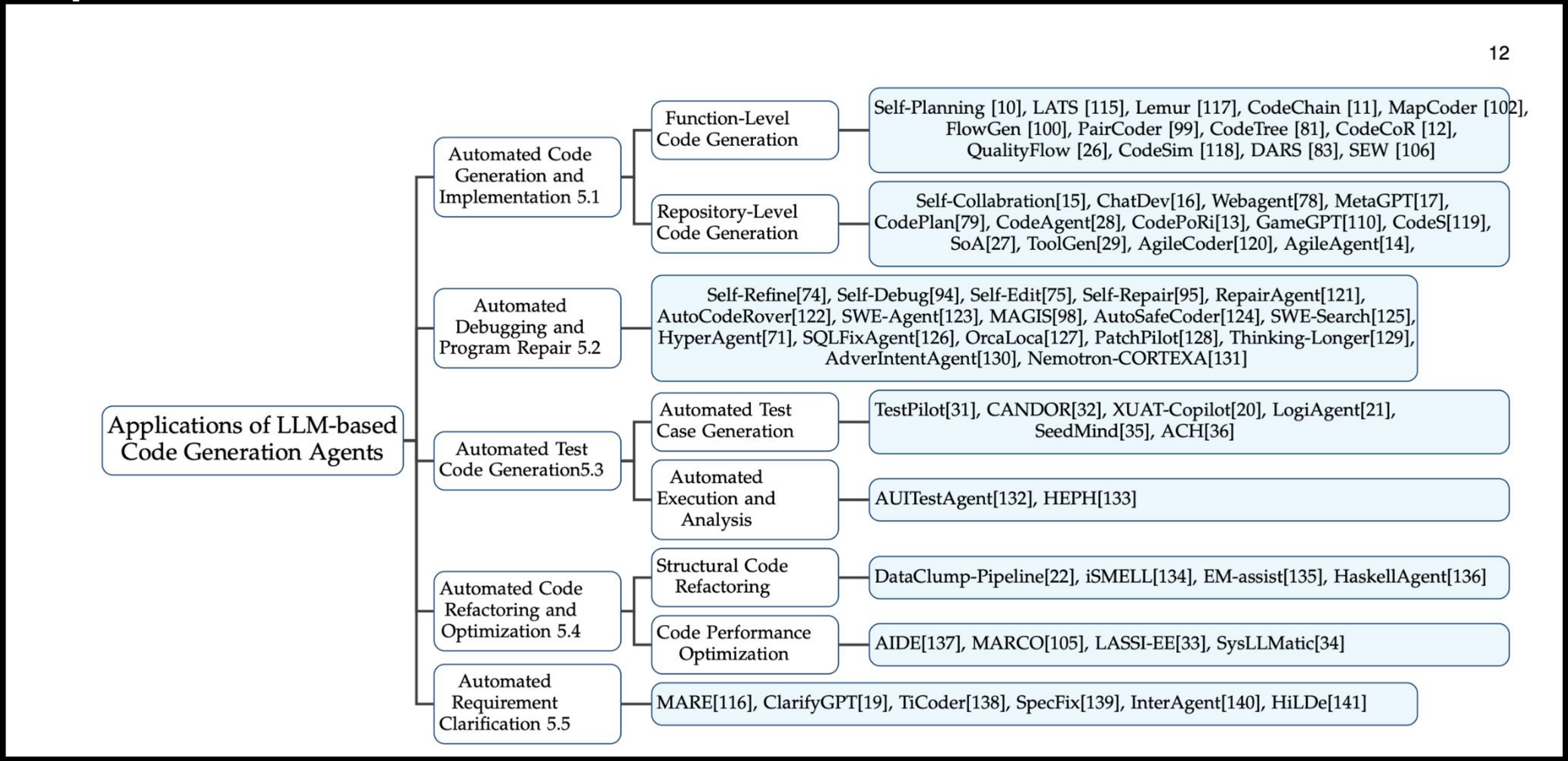
Dong 2025 A Survey on Code Generation with LLM-based Agents
Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

Multi-Agent Code Generation



Dong 2025 A Survey on Code Generation with LLM-based Agents
Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

LLM-based code generation agents in software development tasks

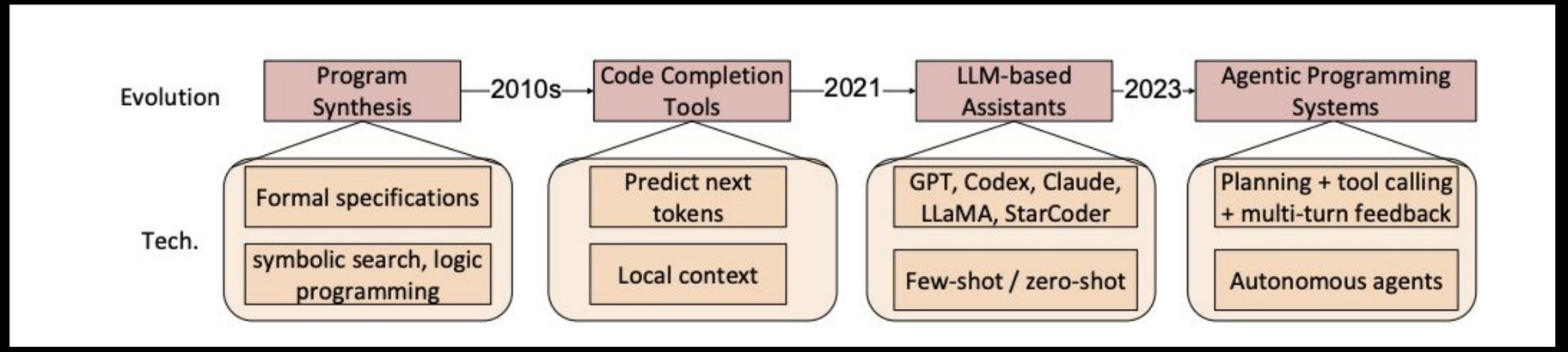


https://arxiv.org/abs/2508.00083v1

Coding LLMs

Typical code generation LLMs include Codex, CodeLlama, DeepSeek-Coder, and Qwen2.5-Coder

Training data contains a large amount of high-quality open-source code libraries and programming documentation, LLMs can learn and master syntactic rules of multiple programming languages as well as common programming paradigms, while understanding the mapping relationship between natural language descriptions and code logic



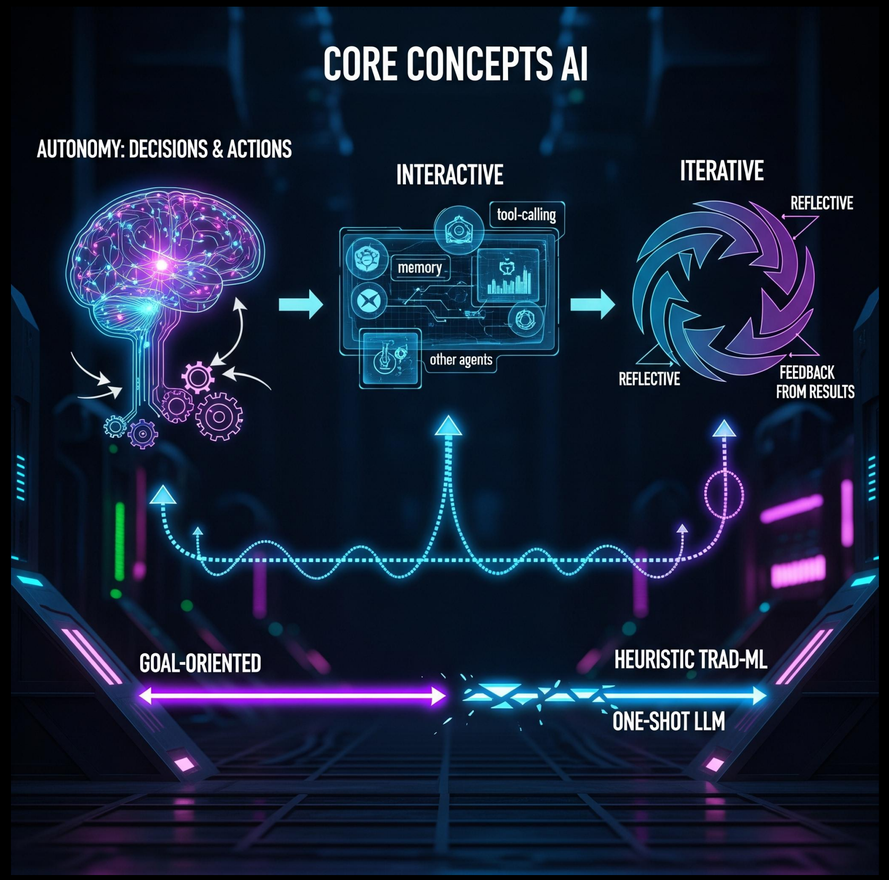
Dong 2025 A Survey on Code Generation with LLM-based Agents
Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

AI Agents & how they differ

- Autonomy: Decisions & Actions
- Interactive: tool-calling, memory, other agents
- Iterative: reflective, feedback from results
- Goal-oriented

Different than

- Heuristic
- Trad-ML
- One-shot LLM



e.g. “Implement a REST API endpoint that returns the top 10 most frequently accessed URLs from a web server log file. Include unit tests and documentation.”
-> This task requires integrating multiple software components, including file parsing, frequency analysis, web API implementation, testing, and documentation.

Dong 2025 A Survey on Code Generation with LLM-based Agents
Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

Top 10 URL Example: Steps

1. Understand prompt
2. Planner
3. Write code for
   a. Parse Log File
   b. Frequency Dictionary & Analysis
4. Write tests
5. Build and execute
6. Document

Many of these steps require tool calls

Many can fail and need iteration

Unlike one-shot LLM calls

Dong 2025 A Survey on Code Generation with LLM-based Agents
Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

Reasoning Model

Generic Reasoning LLM - code generation, task planning, debugging, documentation, and natural language interaction

e.g. GPT-5, Claude, DeepSeek, Gemini

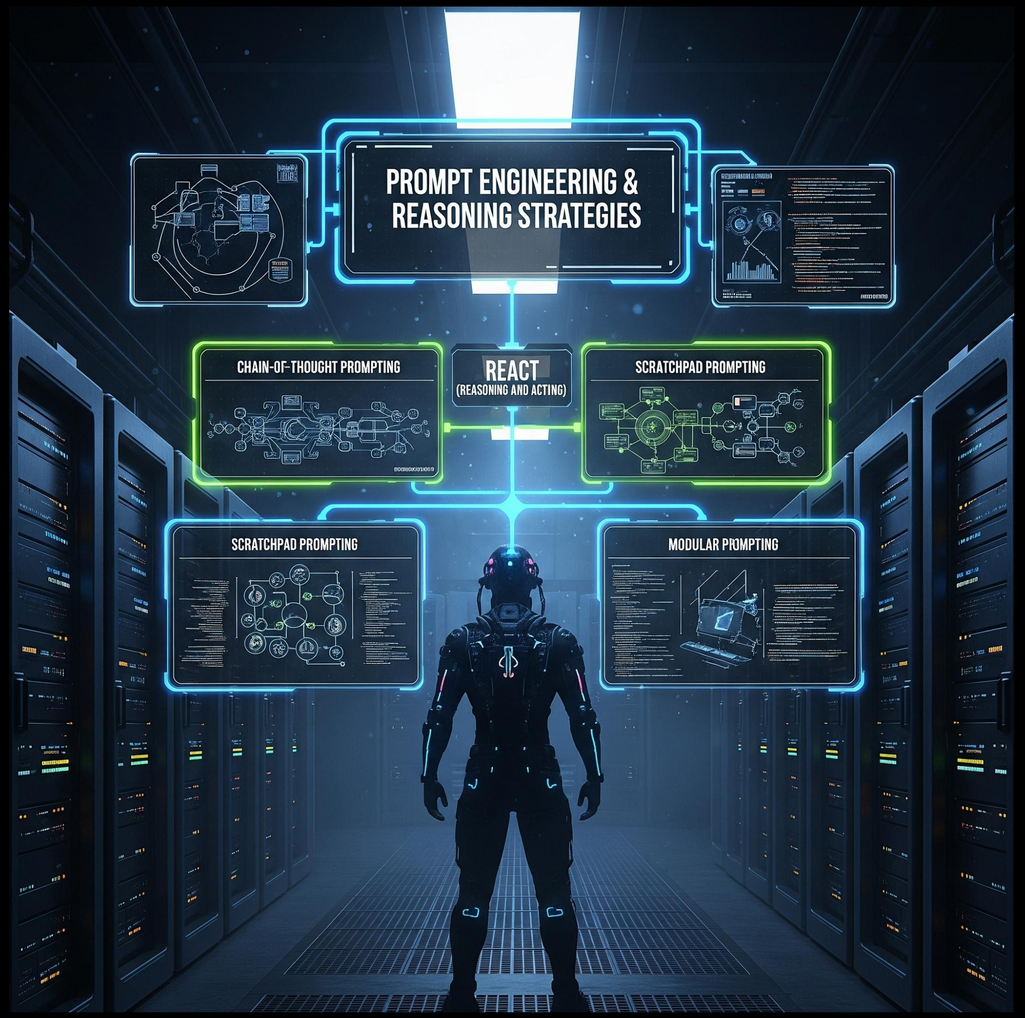
Generate syntactically correct and semantically meaningful code, answer development-related queries, and engage in multi-turn conversations with minimal task-specific fine-tuning

Some LLMs like Grok and Claude Opus increasingly optimized for coding tasks through specialized instruction tuning, extended context length, tool use capabilities, and integration with retrieval-based systems

Dong 2025 A Survey on Code Generation with LLM-based Agents
Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

Prompt Engineering & Reasoning Strategies

Chain-of-thought prompting
ReAct (reasoning and acting)
Scratchpad prompting
Modular prompting



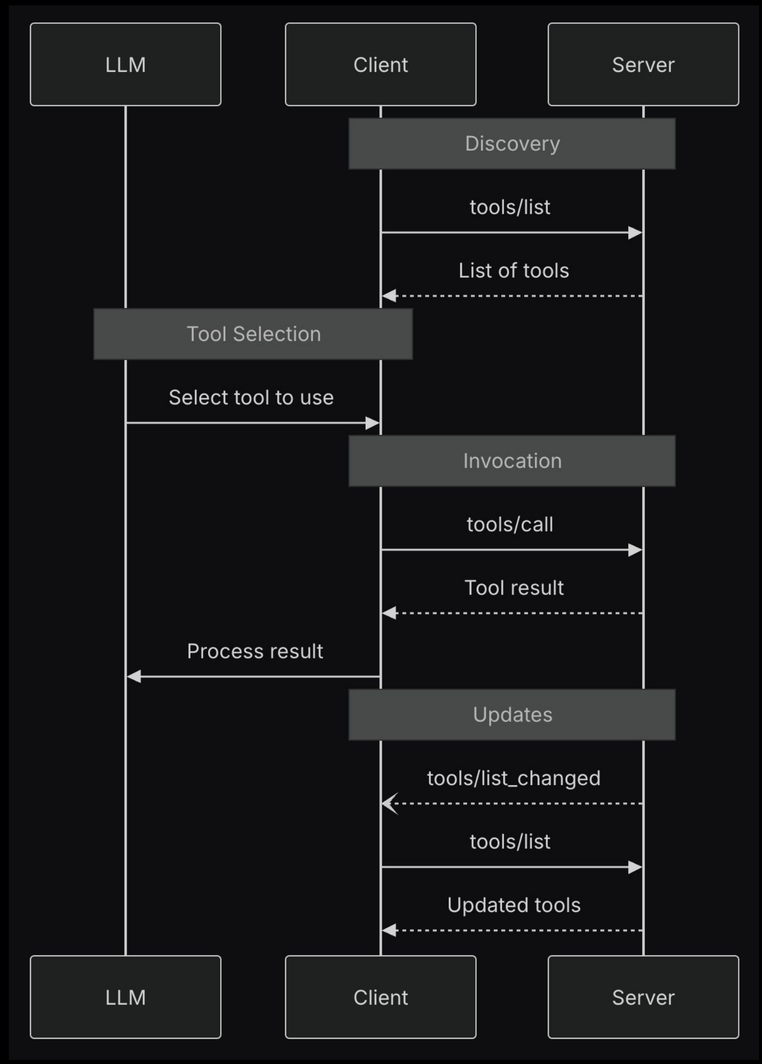
Dong 2025 A Survey on Code Generation with LLM-based Agents
Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

Tool use, MCP

The Model Context Protocol (MCP) allows servers to expose tools that can be invoked by language models

Tools enable models to interact with external systems, such as querying databases, calling APIs, or performing computations. Also for file discovery, compilers, executing, version control, etc.

Each tool is uniquely identified by a name and includes metadata describing its schema.



Tools - Model Context Protocol

Copilot Core Tools

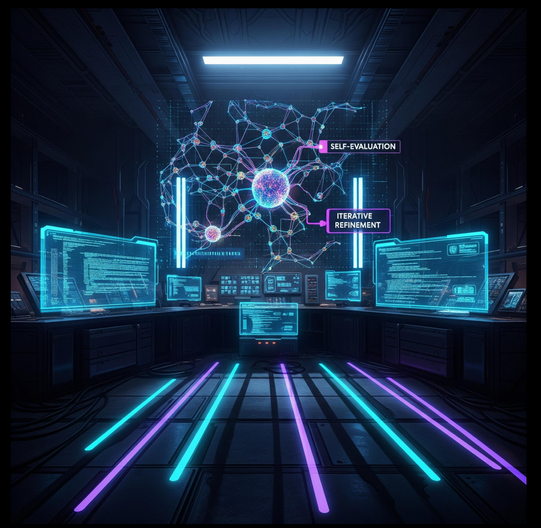
Tool Type	Examples
Compiler	gcc [1], clang [2], javac [3], tsc [4]
Debugger	gdb [5], lldb [7], pdb [8]
Test Framework	pytest [9], unittest [10], Jest [11], Mocha [12]
Linter	eslint [13], flake8 [14], black [15], prettier [16]
Version Control	git [17]
Build System	make [18], cmake [19], npm [20], maven [21]
Package Manager	pip [24] yarn [25], cargo [26]
Language Server	pyright [22], tsserver [23]

Dong 2025 A Survey on Code Generation with LLM-based Agents
Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

Reflection and Self-Improvement

Unlike one-shot generation methods, these approaches enable the model to review its intermediate outputs, provide internal feedback, and iteratively refine the code during the generation process. By mimicking the human process of generating, evaluating, and revising code, they help improve the overall correctness and quality of the final output.

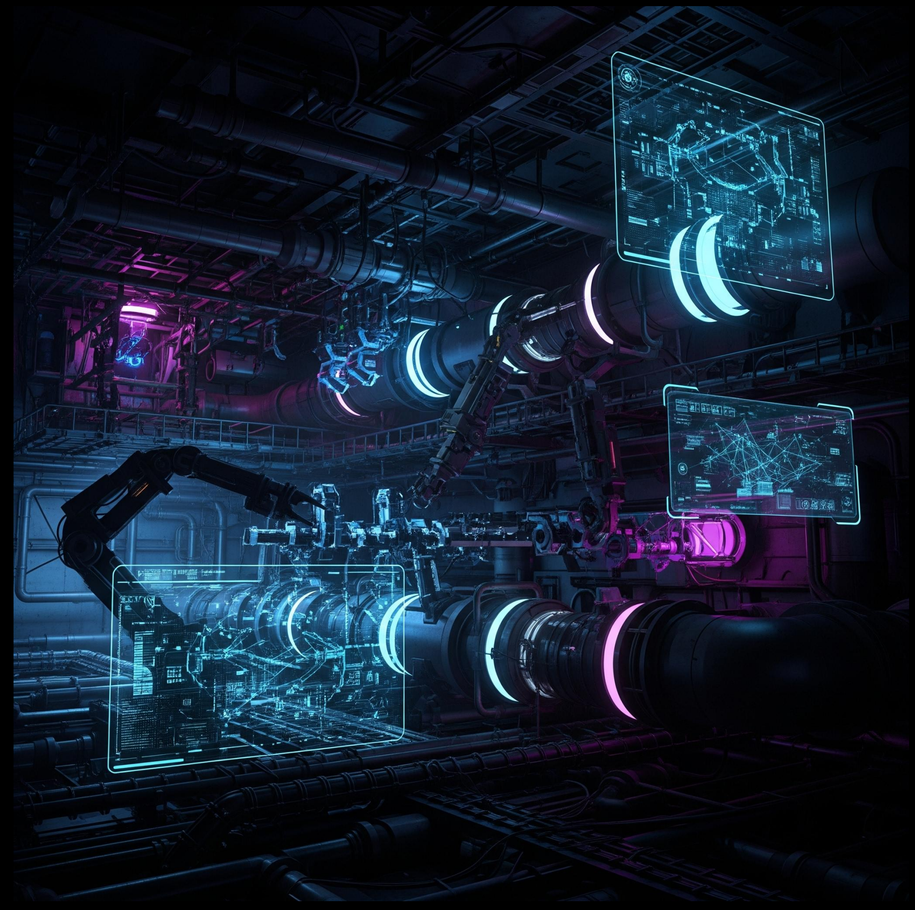
After generating an initial output, the model performs natural language self-evaluation to identify potential issues.



Dong 2025 A Survey on Code Generation with LLM-based Agents
Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

Multi-agent Architectures

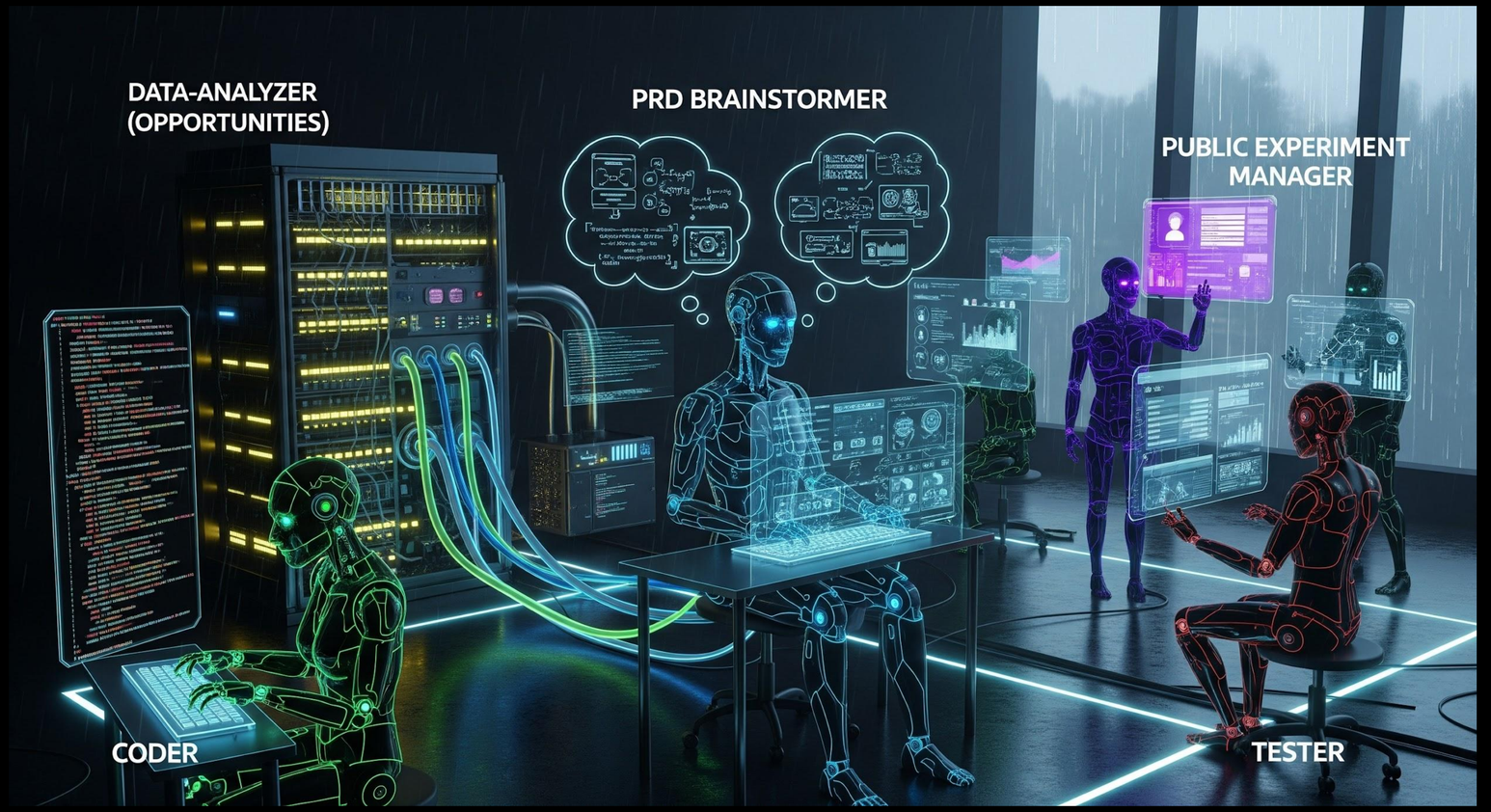
Pipeline-based labor division
Hierarchical Planning x Execution
Self-negotiation
Self-evolving structural updates



Dong 2025 A Survey on Code Generation with LLM-based Agents
Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

Multi-agent Architectures: Pipeline-based

Data-Analyzer (opportunities) -> PRD Brainstormer -> Coder -> Tester -> Public Experiment Manager -> Data Analyzer



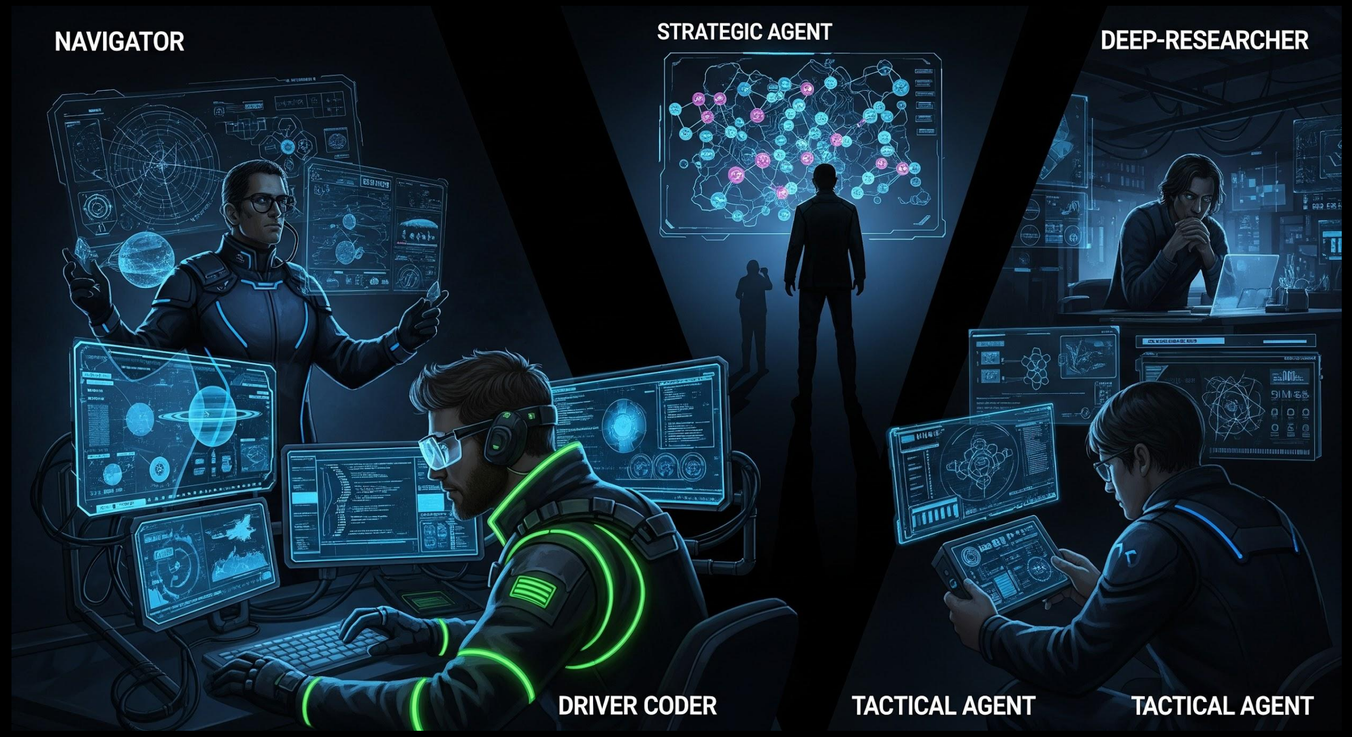
Dong 2025 A Survey on Code Generation with LLM-based Agents
Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

Multi-agent Architectures: Hierarchical

Navigator<>Driver

Coder<>Deep-researcher

Task breakdown - strategic vs tactical agents



Dong 2025 A Survey on Code Generation with LLM-based Agents

Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

Multi-agent Architectures: Self-negotiation

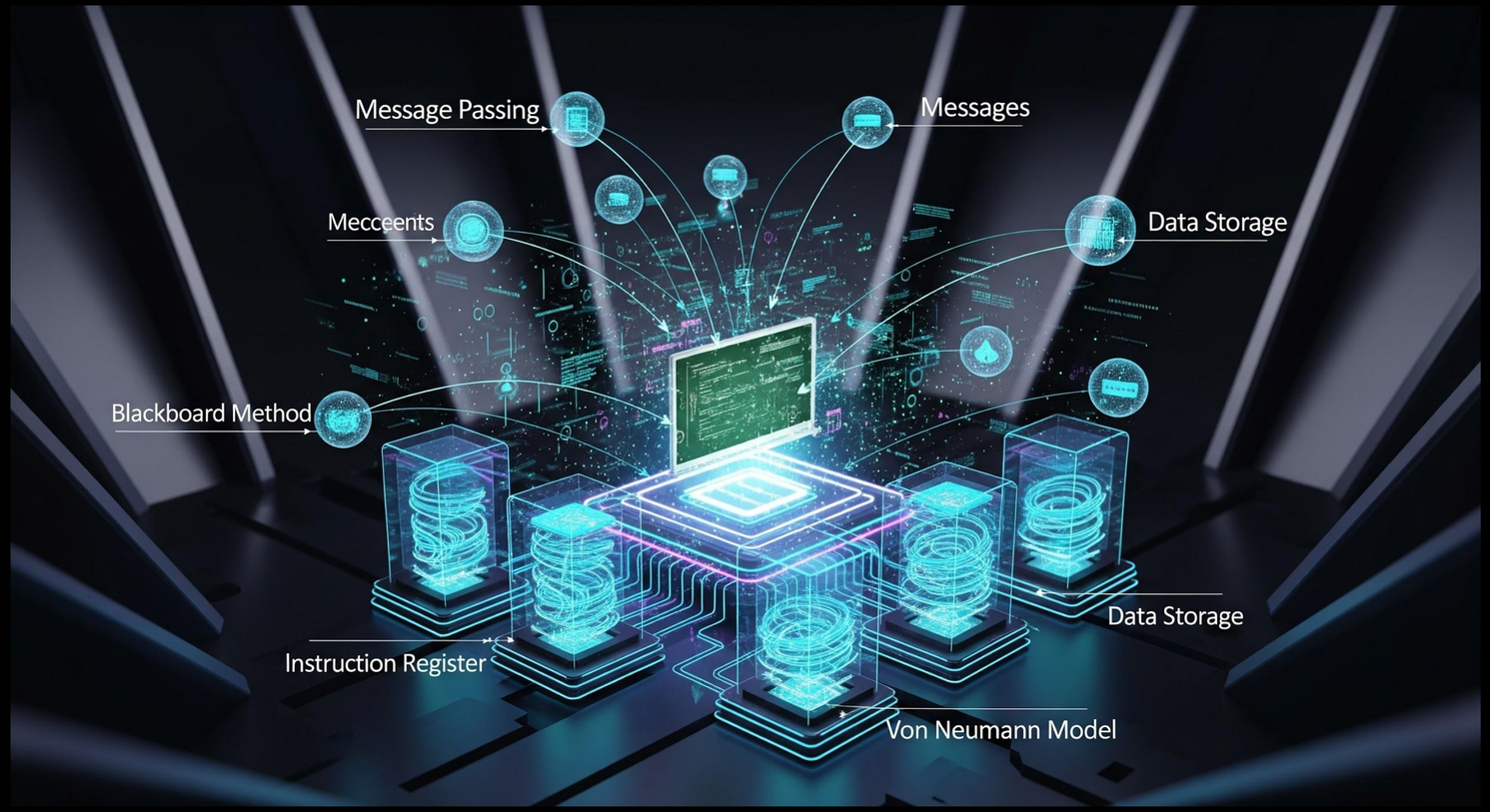
Reflection agents grade/score work of each stage of multi-agents providing feedback to iterate on the work

Dong 2025 A Survey on Code Generation with LLM-based Agents
Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

Context Management & Memory Technologies

Message Passing
Blackboard Method
vonNeuman Model

- instruction register
- data storage



Dong 2025 A Survey on Code Generation with LLM-based Agents
Wang 2025 AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities

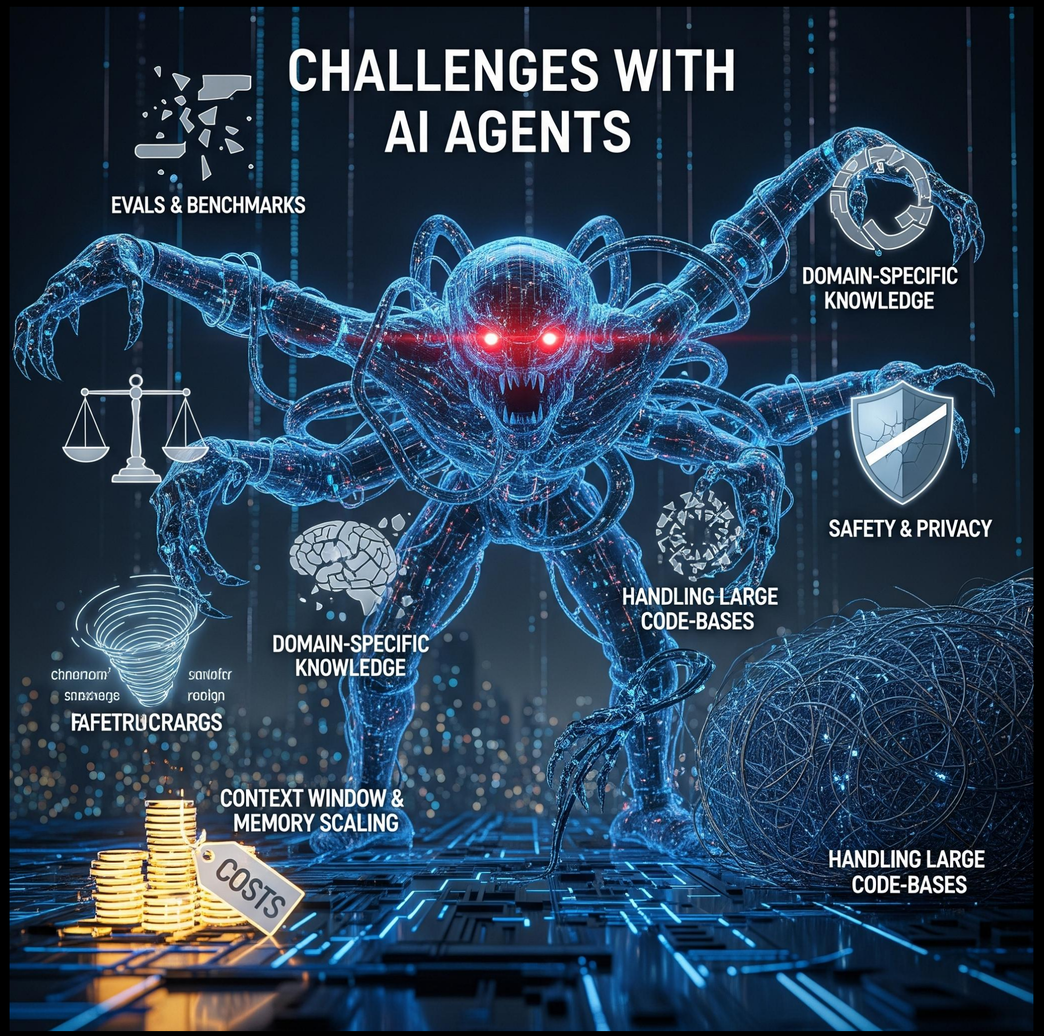
Agenda

- Ch1: In a galaxy not far far away…
  Challenges of Scale & Need for focus on Code Quality
- Ch2: Karpathy Canon
  Rise of the Vibe Coder
- Ch3: Ex-Machina
  The Anatomy of an agent
- Ch4: Prometheus Unbound
  Harnessing AI Agents
- Ch5: The Agent Trilogy
  Cases from the Field

Ch4:

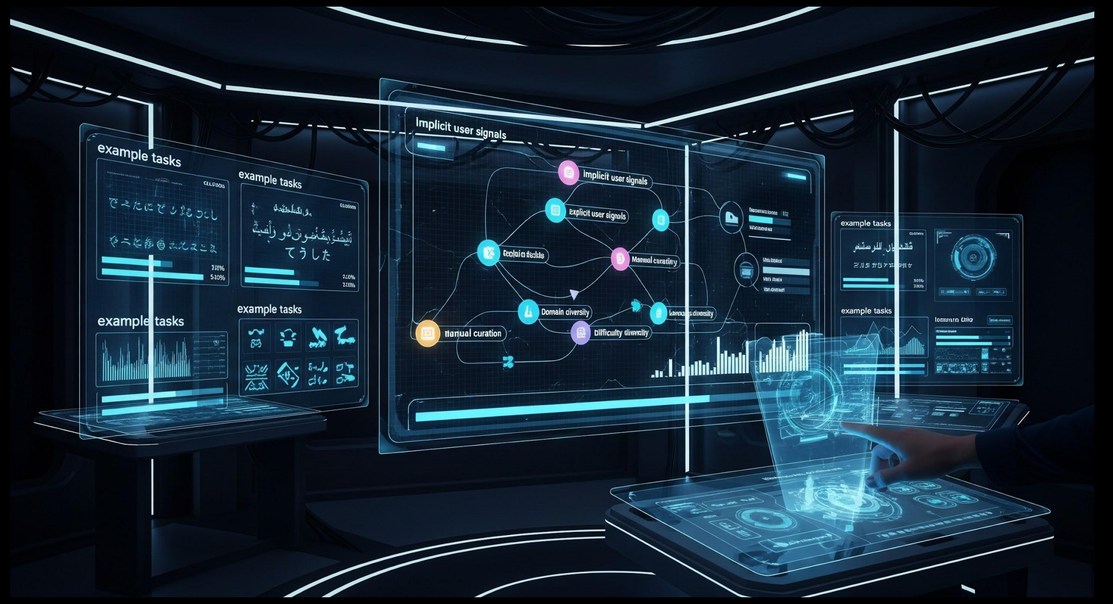
Challenges with AI Agents

- Evals & Benchmarks for your tasks
- Domain-specific knowledge
- Context Window & Memory Scaling
- Costs
- Handling Large Code-bases



Evals & Benchmarks

- Setting example tasks from various sets of your org with manual curation
- Gathering from implicit and explicit user signal
- Having enough diversity in domain, language, difficulty, etc for the tasks



Evals & Benchmarks: Manual & Automated

- Manual Curation
  - Google’s benchmark for Code Reviews - 100 manually curated <review comment, patch> pairs
- Automated annotation
  - Meta’s mining of 5000 <review comment, patch> pairs
  - “OSS benchmarks do not necessarily capture the patterns, depth and complexity involved in generating patches and addressing code review comments at a large enterprise” - can likely extend this to any sufficiently nuanced code-base

Froemmgen 2024 Resolving Code Review Comments with Machine Learning

Maddila 2025 AI-Assisted Fixes to Code Review Comments at Scale

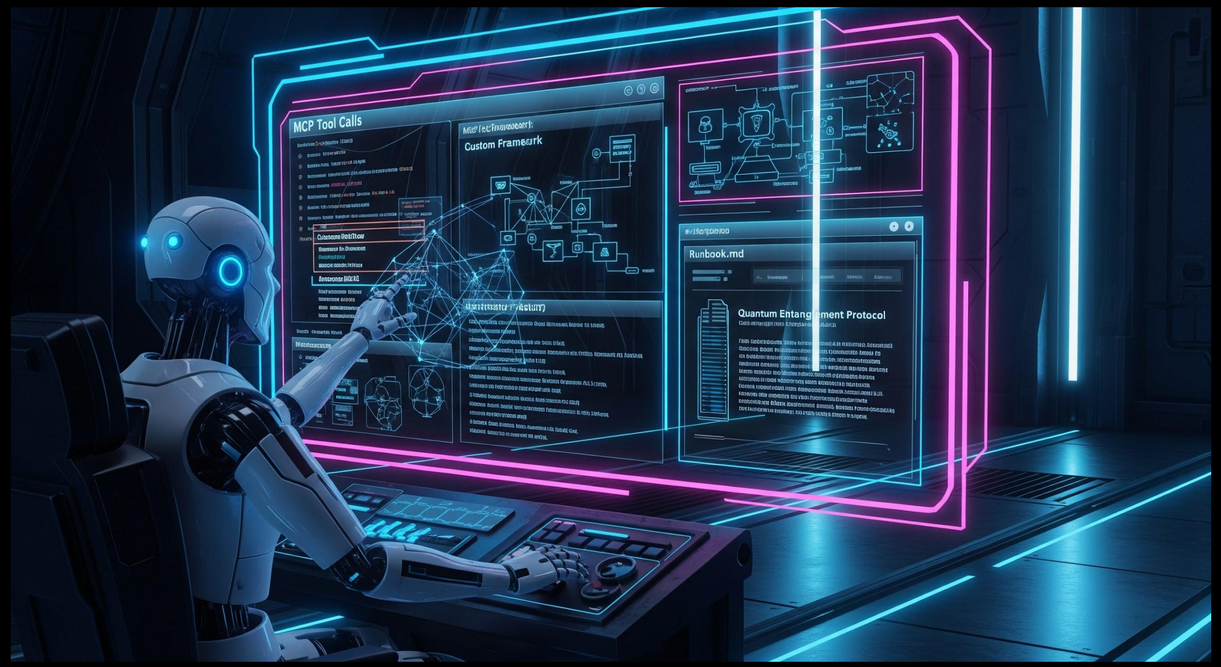
LargeLSFT: Scaling SFT with Human + AI Labeling (for code-review patching)

- Human SFT build-out: 14 weeks, ~7 annotators on average ⇒ 18k ⟨review_comment, patch⟩ pairs.
- Classifier to expand data: Llama 8B trained on 7.5k human-labeled points to predict “good vs bad.”
- Classifier quality (on 700 held-out): Precision 0.88, Recall 0.75, Accuracy 79%.
- Rapid scale-up: Generated +46k additional data points; <1 day to scale to 48k ⟨review_comment, patch⟩ pairs vs 14 weeks for human 2.5k→18k.
- Total SFT corpus: 64k examples (18k human + 46k model-classified).
- Model & context: Fine-tuned Llama-70B → Llama-LargeLSFT with 128k context (vs 16k in SmallLSFT).
- Rationale: Leverages tolerance that large SFT size can offset slight quality drop to boost performance at scale.

Froemmgen 2024 Resolving Code Review Comments with Machine Learning
Maddila 2025 AI-Assisted Fixes to Code Review Comments at Scale

Domain specific knowledge

- MCP Tool calls - especially useful for custom frameworks and DSLs
- Runbook .md files - great for recipes and cookbooks - so the agent “doesn’t do its own thing”

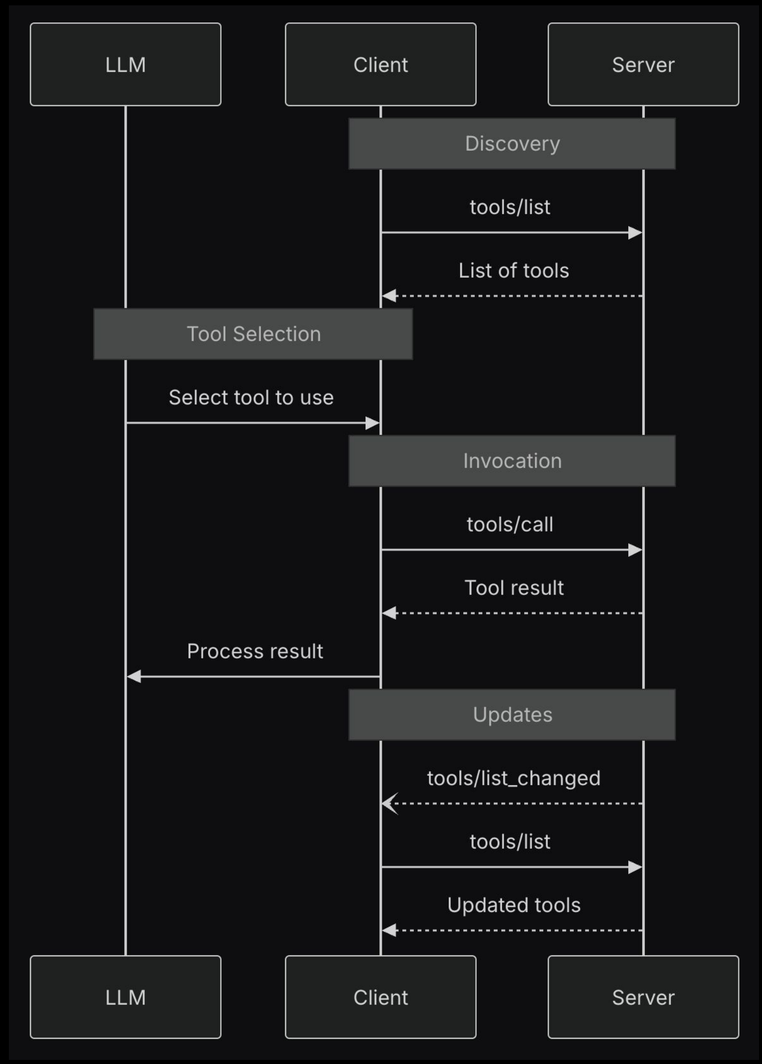


Tool use, MCP

The Model Context Protocol (MCP) allows servers to expose tools that can be invoked by language models

Tools enable models to interact with external systems, such as querying databases, calling APIs, or performing computations. Also for file discovery, compilers, executing, version control, etc.

Each tool is uniquely identified by a name and includes metadata describing its schema.



Tools - Model Context Protocol

Domain specific knowledge: MCP Tool examples

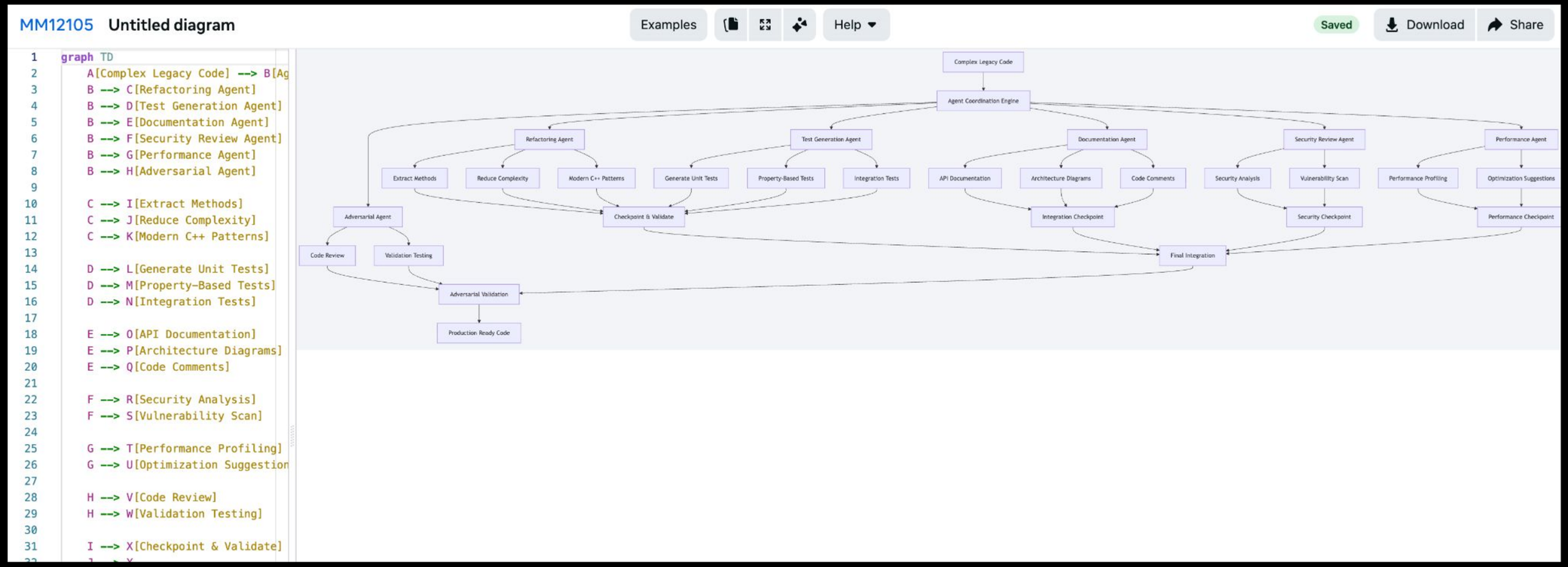
- Example: Annotated code with __DEADCODE_SCAVENGER__

```
"type": "object",
        "properties": {
            "root": {"type": "string", "description": "Repo root to scan", "default": "."},
            "action": {"type": "string", "enum": ["list","stats","patch"], "default": "stats"},
            "include_ext": {
                "type": "array",
                "items": {"type": "string"},
                "description": "File extensions to include (no dot). Empty means all.",
                "default": DEFAULT_INCLUDE_EXT
            },
        },
```

P1949686102

Context Window & Memory Issues

- Summarize the code-base as .md files with mermaid diagrams for class diagrams, interaction diagrams, etc etc
- Session Trace & Checkpointing
- llms.txt file (https://llmstxt.org/)



Context Window & Memory Issues

- Summarize the code-base as .md files with mermaid diagrams for class diagrams, interaction diagrams, etc etc
- Session Trace & Checkpointing
- llms.txt file (https://llmstxt.org/)

```
/checkpoint [optional-name] - Save current session state as a checkpoint
 /list-checkpoints - Show available checkpoints with their session IDs
```

Context Window & Memory Issues

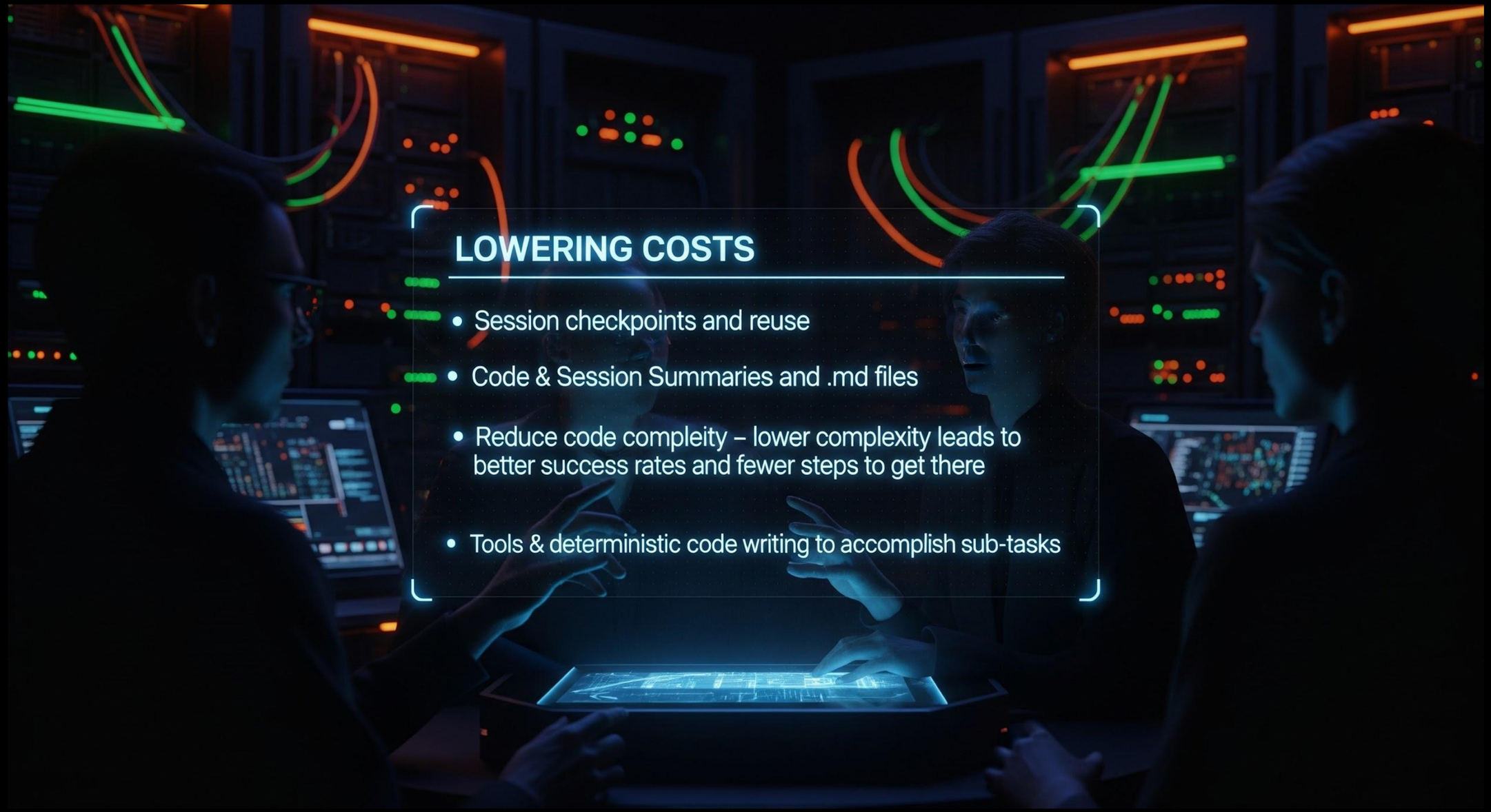
- Summarize the code-base as .md files with mermaid diagrams for class diagrams, interaction diagrams, etc etc
- Session Trace & Checkpointing
- llms.txt file (https://llmstxt.org/)

```
# FastHTML

> FastHTML is a python library which brings together Starlette, Uvicorn, HTMX, and fastcore's `FT` "FastTags" into a library for creating server-rendered hypermedia applications.

Important notes:

- Although parts of its API are inspired by FastAPI, it is *not* compatible with FastAPI syntax and is not targeted at creating API services
- FastHTML is compatible with JS-native web components and any vanilla JS library, but not with React, Vue, or Svelte.

## Docs

- [FastHTML quick start](https://fastht.ml/docs/tutorials/quickstart_for_web_devs.html.md): A brief overview of many FastHTML features
- [HTMX reference](https://github.com/bigskysoftware/htmx/blob/master/www/content/reference.md): Brief description of all HTMX attributes, CSS classes, headers, events, extensions, js lib methods, and config options

## Examples

- [Todo list application](https://github.com/AnswerDotAI/fasthtml/blob/main/examples/adv_app.py): Detailed walk-thru of a complete CRUD app in FastHTML showing idiomatic use of FastHTML and HTMX patterns.
```

Context Window & Memory Issues

- Memory Compression
- Caching
- Offloading

Reducing Costs

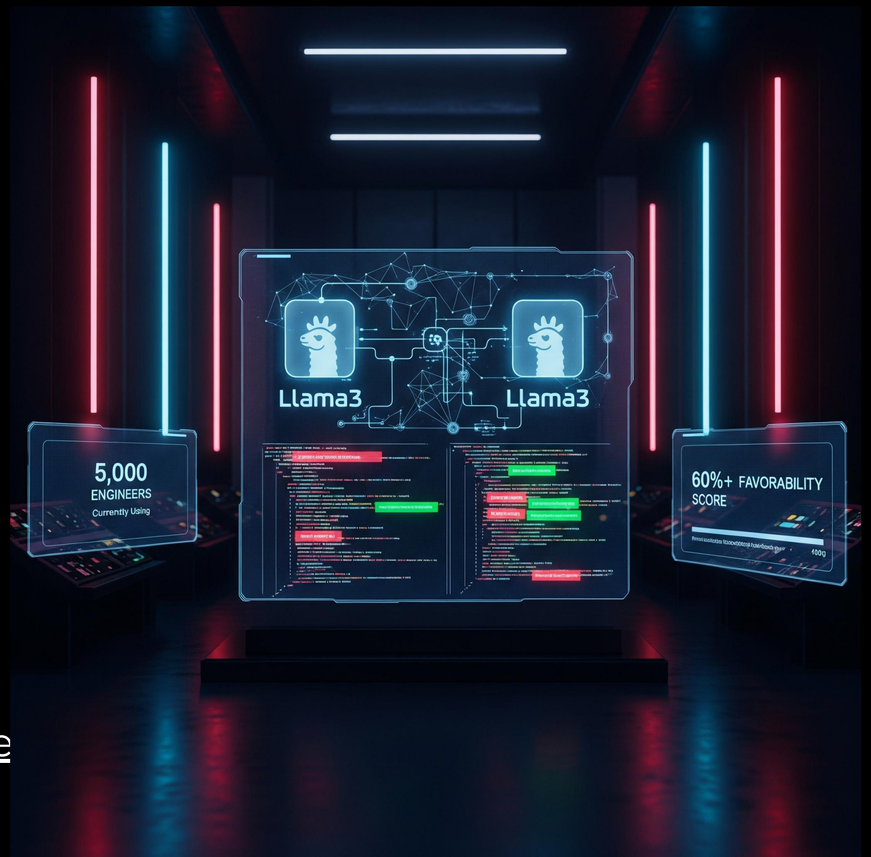


CH5:
THE AGENT TRILOGY

THREE CASE STUDIES

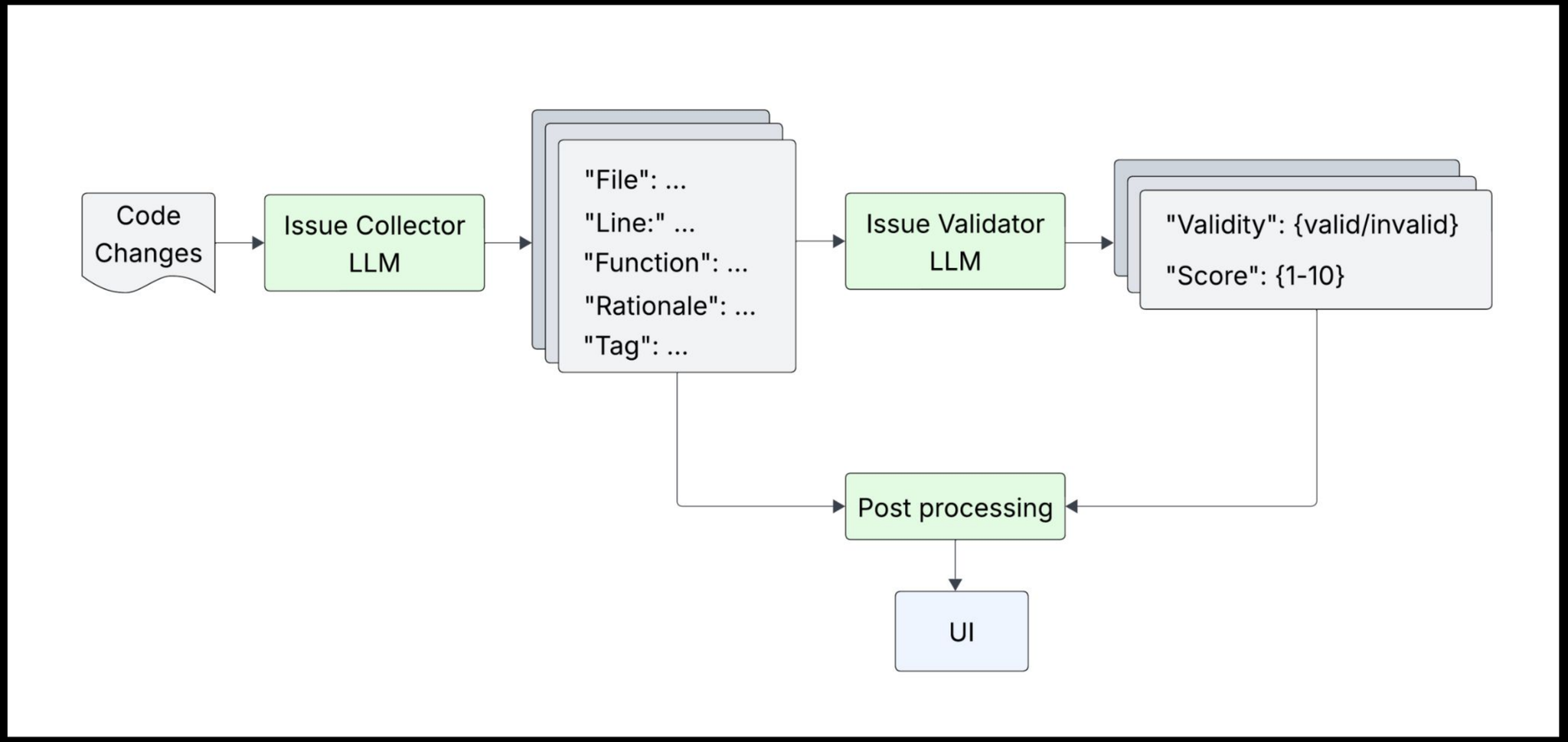
Code Quality Score

- Powered by two Llama3 models
  - detect common code quality issues related to coding best practices
  - provide good “critiques” for LLM-generated code review
- Used by 5K engineers, with 60%+ favorability score



Wong 2025 A Note on Code Quality Score: LLMs for Maintainable Large Code

Code Quality Score



Wong 2025 A Note on Code Quality Score: LLMs for Maintainable Large Codebases

Code Quality Score

src/handler/Handler.cpp

```
...  @@ -412,7 +413,9 @@
413  413    initCache();
414  414
415  415    const auto& rankCfg =
415  -      *cfg->get_rankSettings();
416  +      FLAG_use_old_rank_settings
417  +      ? *cfg->get_rankSettings()
418  +      : *idx::RankSettingsProvider::getSnapshot();
419  419    const auto& structCfg =
420  420        *cfg->get_structConfig();
421  421
```

src/retriever/RetrieverHandler.cpp

```
...  @@ -262,8 +263,10 @@
263  263
264  264    const auto structCfg =
265  265        *rCfg->get_structConfig();
265  -  const auto rankCfg =
266  -      *rCfg->get_rankSettings();
266  +  const auto& rankCfg =
267  +      FLAG_use_old_rank_settings
```

File: src/retriever/RetrieverHandler.cpp
Line: 267
Function: handleRetrieval
Rationale: The new code has duplicated logic in both Handler.cpp and RetrieverHandler.cpp for getting the ranking settings.
Tag: DedupeLogic

```
268  +      ? *rCfg->get_rankSettings()
269  +      : *idx::RankSettingsProvider::getSnapshot();
270  270    const auto dataLoaders = std::make_shared(
271  271        *rCfg->get_structConfig()->whitelistModels());
272  272
```

Wong 2025 A Note on Code Quality Score: LLMs for Maintainable Large Codebases

Code Quality Score: Issue Collector Agent

- Llama 3.1-70B
- Scans diffs, outputs scores per issue
  - readability
  - complexity
  - modularity

SFT followed by DPO

Wong 2025 A Note on Code Quality Score: LLMs for Maintainable Large Codebases

Code Quality Score: Issue Collector Agent

A.1 SYSTEM INSTRUCTION FOR THE ISSUE COLLECTOR LLM

```
=== Raw Source Control Code Changes BEGIN ===
code changes go here
=== Raw Source Control Code Changes END ===

=== Context for the Code Change BEGIN ===
The following additional information from the author may
provide context about the code changes and their purpose:
Title:  title
Summary:  summary of code changes
=== Context for the Code Change END ===

Your task is to provide constructive and concise feedback for
the code changes based on the following criteria (each goes
with a tag name in the beginning):
```

- DedupeLogic: Deduplicate logic into shared functions except for logging.
- DictionaryKeyExistenceCheck: Check for dictionary key existence before accessing it.
- UseConstant: Use constants instead of literals, unless it's a constant definition.
- RenamingVariable: Variable names should be pronounceable, easily readable, and reveal intent.
- DomainSpecificName: Use solution domain names, computer science (CS) terms, algorithm names, pattern names, math terms. When there is no name from the solution domain then prefer problem domain names.
- BreakdownLongFunction: Break down long functions into smaller, more focused functions. Only include this issue if the length of the current function is longer than 50 lines.
- ExtractMethod: When we have a block of code that can be extracted into a separate method, we should do so.

Wong 2025 A Note on Code Quality Score: LLMs for Maintainable Large Codebases

Code Quality Score: Issue Collector Agent

```
Example output:

{
  "function": "[exact function name from code]",
  "rationale": "[clear explanation of the issue and suggested
  improvement]",
  "file": "[file path where the function is located]",
  "line": [numeric line number],
  "tag": "[one of the tags as described above]"
}

Note you should not be overly critic and generate many bullet
points.  Please pin-point the 'problematic_function' with
issues, and don't forget to include line numbers.  The
output field 'problematic_function' is mandatory in the
response if the problem is inside a function.  The output
field 'relevant_file' should include full path of the file
(including directory and filename).  In your response, do
not describe what the code changes is about because the code
author already knows about it.  Pay attention to the code
change.
```

Wong 2025 A Note on Code Quality Score: LLMs for Maintainable Large Codebases

Code Quality Score: Issue Validator Agent

- Llama 3.1-70B

Validates each score of the previous agent

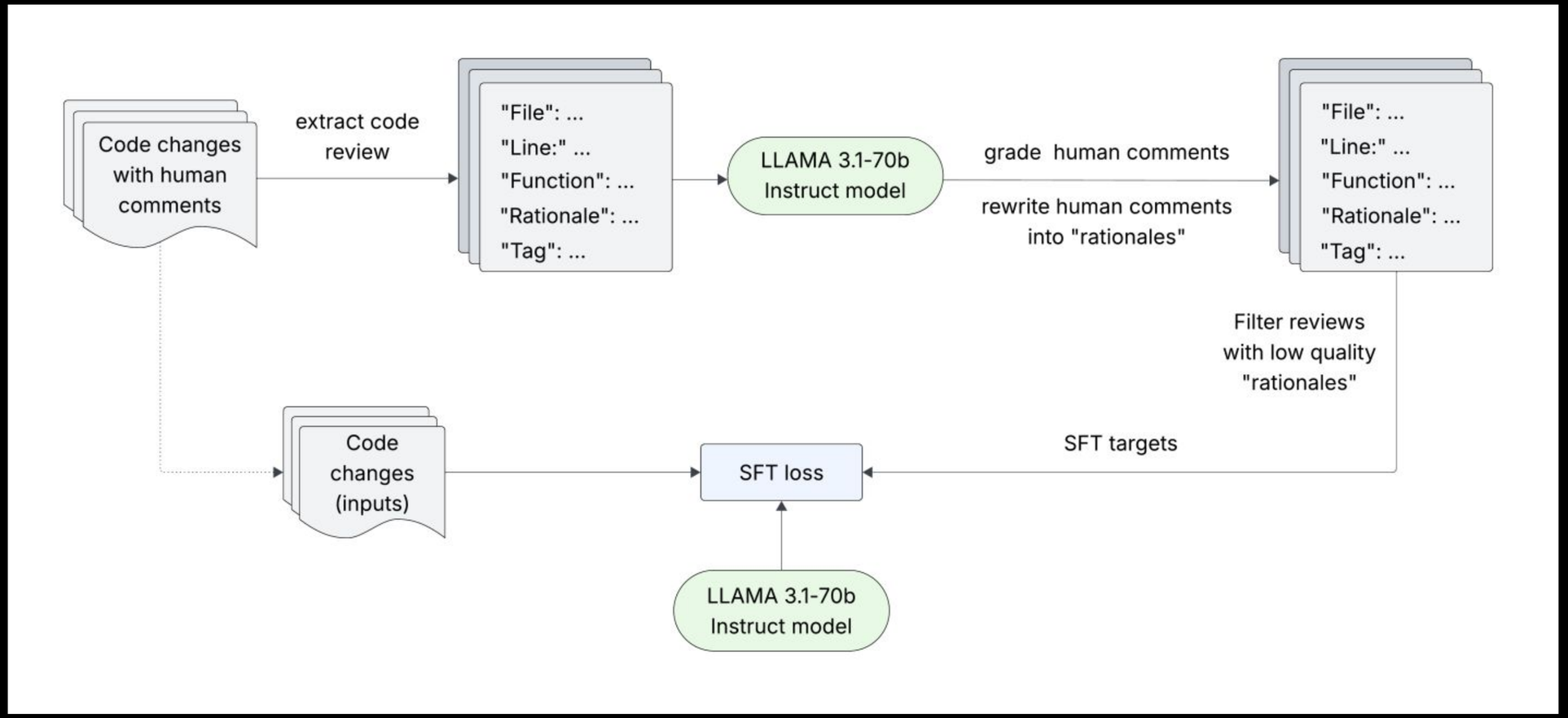
Wong 2025 A Note on Code Quality Score: LLMs for Maintainable Large Codebases

Code Quality Score: Issue Validator Agent

```
Your input is a difference view of code changes, and a list
of code issues for it.
Your goal is to inspect, review the issues and score each of
them.
Be aware - the issues may not always be correct or accurate,
and you should evaluate them in relation to the actual code
changes presented.


Review issues.  Carefully review both the issue content, and
the related code changes.  Mistakes in the issue can occur.
Make sure the issues are correct and properly derived from
the code changes.  Mark each issue as valid or invalid.

Score issues.  High scores (8 to 10) should be given to
correct issues that address major bugs, or security concerns.
Lower scores (3 to 7) should be for issues that address
minor concerns for example code style, code readability,
maintainability, etc.  Don't give high scores to suggestions
that are not crucial, and bring only small improvement or
optimization to the code.  Incorrect issues should be scored
as 0.  Order the feedback the same way the issues are ordered
in the input.
```

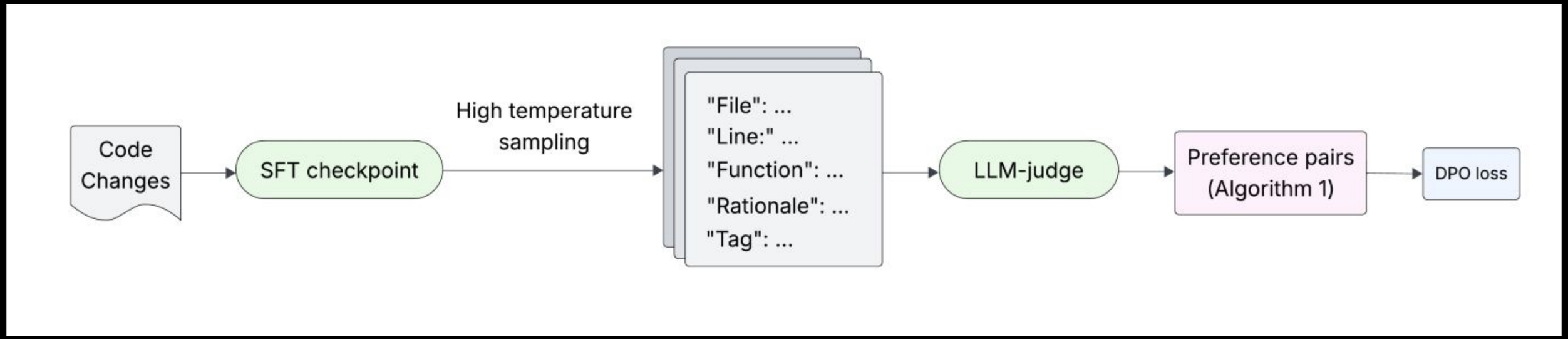
Wong 2025 A Note on Code Quality Score: LLMs for Maintainable Large Codebases

Issue Collector Agent Training S1



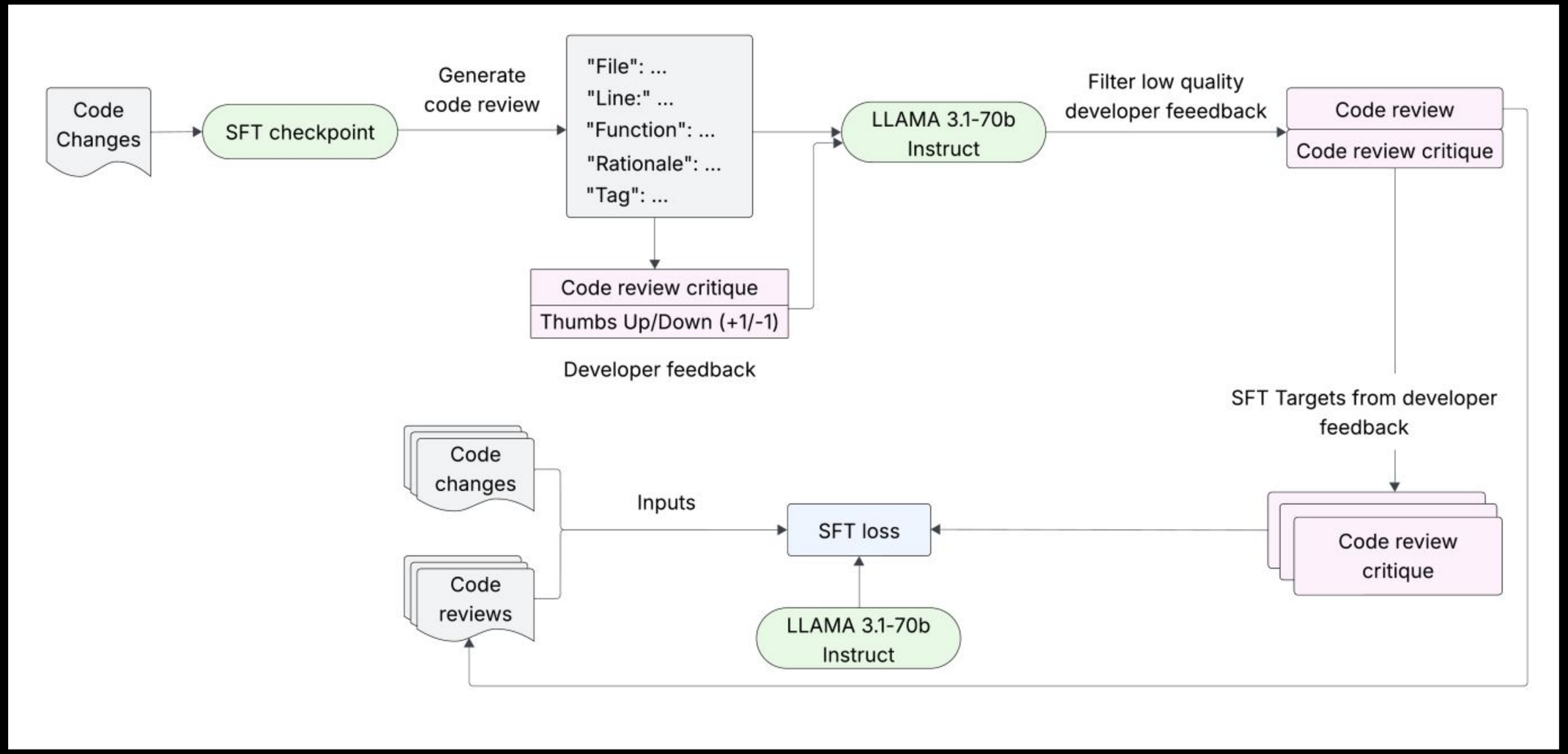
Wong 2025 A Note on Code Quality Score: LLMs for Maintainable Large Codebases

Issue Collector Agent Training S2



Wong 2025 A Note on Code Quality Score: LLMs for Maintainable Large Codebases

Issue Validator Agent Training



Wong 2025 A Note on Code Quality Score: LLMs for Maintainable Large Codebases

Diff Risk Score

How likely is a diff to cause a SEV (a severe fault that impacts end-users)

TABLE I: The features used in the logistic regression that is in production

Feature type	Feature used in Logistic Regression
Diff	*log* of the added and deleted SLOC relative to size of file (ratio)
	New files created by the diff (boolean)
	Diff only creates new files (boolean)
Diffusion	log of the number of files in this diff
	log of the number of authors that modified changed files
Criticality	Previous SEV in the file (boolean)
	Previous SEV in the folder (boolean)
	Is file involved in high-criticality service (boolean)
File	Total logical complexity of files touched in this diff
	Programming language (seven boolean indicators if at least one file in that language is modified)
Expertise	If the author is the original creator of the file
	Number of diffs previously landed by the author

TABLE II: The features fed to the LLMs during both training and inference for DRS

Feature type	Feature fed to the LLM
Diff Title	Title of the diff, typically a concise description of the code change in a few words
Test Plan	Commands (build, lint, tests) executed by the diff author to validate the code changes
Code changes	Filenames and the corresponding code changes in the standard unified diff (“unidiff”) format

Abreu 2024 Moving Faster and Reducing Risk: Using LLMs in Release Deployment

Diff Risk Score

How likely is a diff to cause a SEV (a severe fault that impacts end-users)

Model	Weekend (g = 5%)	
	SEVs Captured	vs Regression
Logistic Regression	18.7 %	– ×
StarBERT	11.5 %	0.61 ×
iCodeLlama-34B	10.8 %	0.58 ×
iCodeLlama-34B risk aligned	23.6 %	1.26 ×
iDiffLlama-13B	12.1 %	0.65 ×
iDiffLlama-13B risk aligned	**26.2 %**	**1.40 ×**

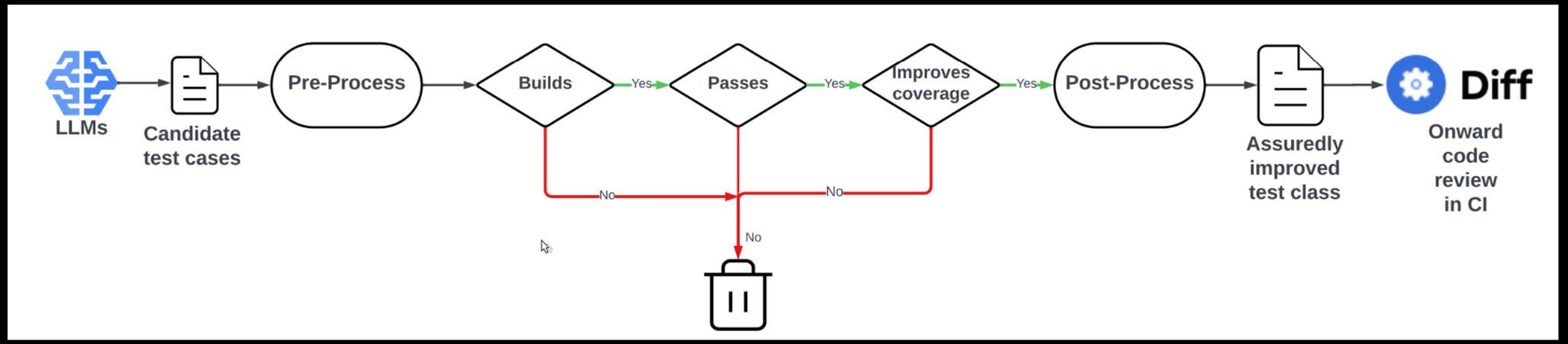
Abreu 2024 Moving Faster and Reducing Risk: Using LLMs in Release Deployment

TestGen-LLM

uses LLMs to automatically improve existing human-written tests

improved 11.5% of all classes to which it was applied

73% of its recommendations being accepted by engineers



Alshahwan 2024 Automated Unit Test Improvement using Large Language Models at Meta

TestGen-LLM

rank	test author	No, of tests	lines covered	diffs
1.	Threads Engineer	40	1,047	8
2.	Home Engineer	34	650	6
3.	Business Engineer	34	443	3
4.	Sharing Engineer	33	816	8
5.	Messaging Engineer	18	157	2
6.	**TestGen-LLM**	17	1,460	17
7.	Friends Engineer	12	143	2
8.	Home Engineer	10	273	2
9.	Creators Engineer	10	198	3
10.	Friends Engineer	10	196	5

Table 1: Results from the First Instagram Test-a-thon, conducted in November 2023. Human test authors are named by the product component on which they worked. The TestGen-LLM tool landed in sixth place overall, demonstrating its human-competitive added value.

Alshahwan 2024 Automated Unit Test Improvement using Large Language Models at Meta

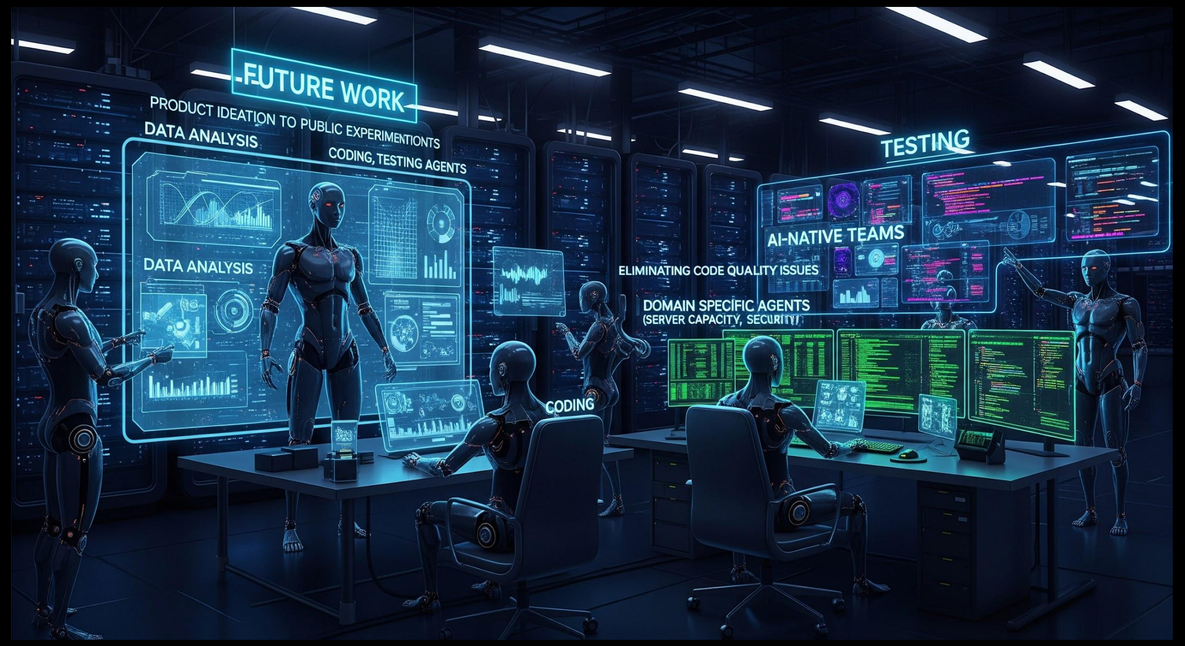
TestGen-LLM: Prompt

Prompt name	Prompt Template
extend_test	Here is a Kotlin unit test class: {existing_test_class}. Write an extended version of the test class that includes additional tests to cover some extra corner cases.
extend_coverage	Here is a Kotlin unit test class and the class that it tests: {existing_test_class} {class_under_test}. Write an extended version of the test class that includes additional unit tests that will increase the test coverage of the class under test.
corner_cases	Here is a Kotlin unit test class and the class that it tests: {existing_test_class} {class_under_test}. Write an extended version of the test class that includes additional unit tests that will cover corner cases missed by the original and will increase the test coverage of the class under test.
statement_to_complete	Here is a Kotlin class under test {class_under_test} This class under test can be tested with this Kotlin unit test class {existing_test_class}. Here is an extended version of the unit test class that includes additional unit test cases that will cover methods, edge cases, corner cases, and other features of the class under test that were missed by the original unit test class:

Alshahwan 2024 Automated Unit Test Improvement using Large Language Models at Meta

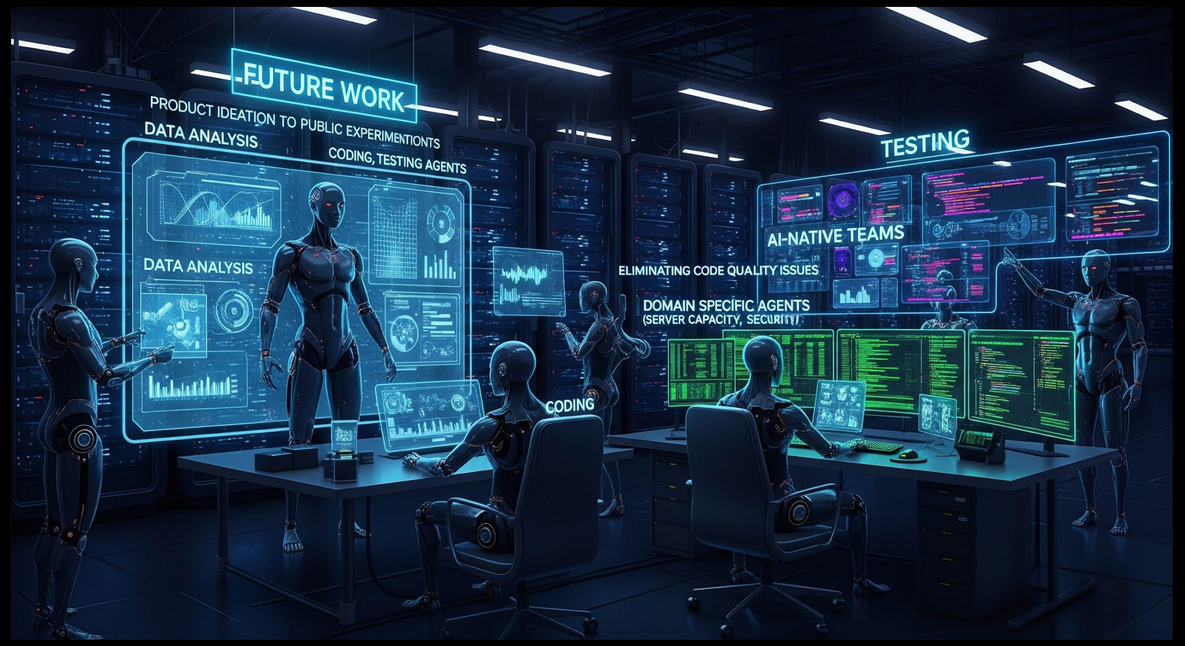
Key takeaways

- These are times of unprecedented challenge and unprecedented opportunities
- Vibe coding is here and AI agents are also here and here to stay
- Success of AI agents for developing new features relies heavily on the state of existing code-base in terms of success rates and costs and time to deliver
- Leveraging context engineering, custom tools, .md files, memory, reflection etc is crucial to make AI agents successful
- AI agents can help scale up defence of code quality, complexity and reliability



Future Work

- Early results in creating patterns for multi-agents to go from product ideation to public experimentation and launch of features
- AI-native teams focussed on leveraging AI for all their work including all non-coding work
- Self-healing systems
- Investing in domain specific agents like server capacity management, security vulnerability detection



Thank you.