DevSecOps at Scale:

Multi-Agent System for Automating Build Failures in Large Enterprise IT Ecosystems

Chidambaram GS, Senior Technologist | Rajeshwari Ganesan, Distinguished Technologist

Infosys

A large enterprise with 315k employees and 100 applications

Distribution Characteristics of Applications

Factor	Details
Deployment Model	Cloud-native, Hybrid, On-Premise, Multi-cloud
Programming Languages	Java, Python, JavaScript, C# etc.
Frameworks/Libraries	Spring-boot, Angular, Golang-Mux, Django, React etc.
Databases	MySQL, PostgreSQL, MongoDB, Oracle, SQL Server, Cassandra, etc.
Cloud Providers	AWS, Azure, Google Cloud, IBM Cloud, Oracle Cloud
Security Types	Vulnerability, Penetration, Static, Dynamic, Compliance
Domain	Finance, Life Sciences, Manufacturing, Healthcare, Retail, etc.
Project Maturity Levels	High, Medium, Low
Compliance Requirements	21 CFR Part 11, HIPAA, GDPR, PCI-DSS, etc.
Containerization	Docker, Kubernetes, OpenShift, ECS, AKS

Technology	Build Failure Rate
Java	27%
Angular	24%
Node.js	19%
Golang	17%
.Net	12%
Python	8%
JavaScript	6%

Distribution Characteristics of Applications

Metric	Impact on Build Failures
Commit Size	Larger commits (> 50 LOC) increase failure rates by 40%
Temporal Patterns	End-of-day builds have a 35% failure rate
Code Review Coverage	Higher coverage reduces failures by 20%
Automated Test Coverage	Higher coverage reduces failures by 30%
Developer Experience	More experienced developers reduce failures by 15%
Toolchain Stability	Stable toolchains reduce failures by 25%
CI/CD Pipeline Maturity	Mature pipelines reduce failures by 20%
Dependency Management	Effective management reduces failures by 15%
Build Frequency	Frequent builds (multiple per day) reduce failure rates by 10%
Environment Consistency	Consistent environments reduce failures by 18%

Internal Platform	Build Failure Rate
Finance	18%
Healthcare	15%
Developer Productivity	25%
Employee Engagement	21%
Human Resource	22%

How do we remove costly delays and resource drain?

Our DevSecOPS journey

Historic Logs

- 500,000 build records over 5 years
- 100 enterprise apps, running in 56 countries
- Over 315,000 users

Process Failure logs

- Filter Build Failures (10k)
- Extract error patterns and resolutions (25+)
- Establish MTTR, Cost, etc.
- Evaluation chassis construction.

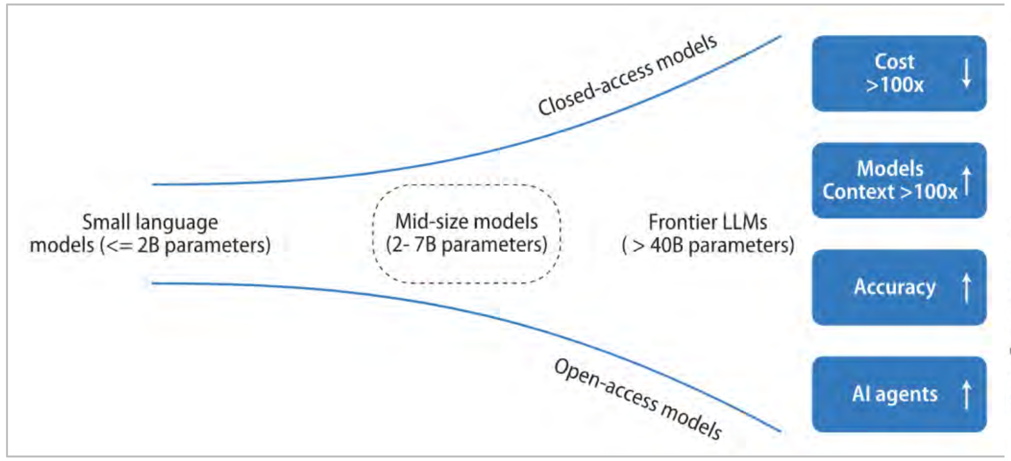
Multi-Agent System

- Prompt engineering with Frontier LLMs for RCA.
- RAG to capture domain specific.
- Support model swappability and prompt management.
- Multi-Agent System for RCA and issue resolution.

Why Agentic AI?

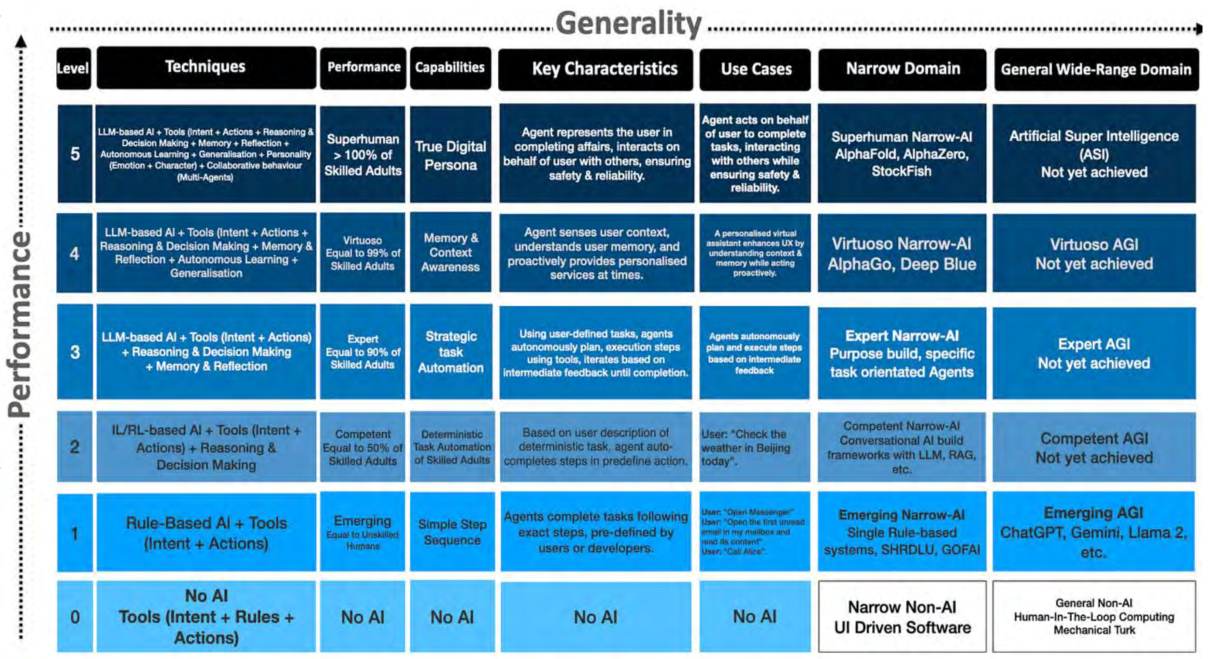
DevSecOps is a partially observable domain, where incomplete data and evolving threats demand advanced reasoning capabilities.

Viability


Closed-access models
Small language models (<= 2B parameters)
Mid-size models (2- 7B parameters)
Frontier LLMs (> 40B parameters)
Open-access models
Cost >100x ↓
Models Context >100x ↑
Accuracy ↑
AI agents ↑

Source: https://www.infosys.com/iki/research/top10-ai-imperatives-2025.html

Feasibility

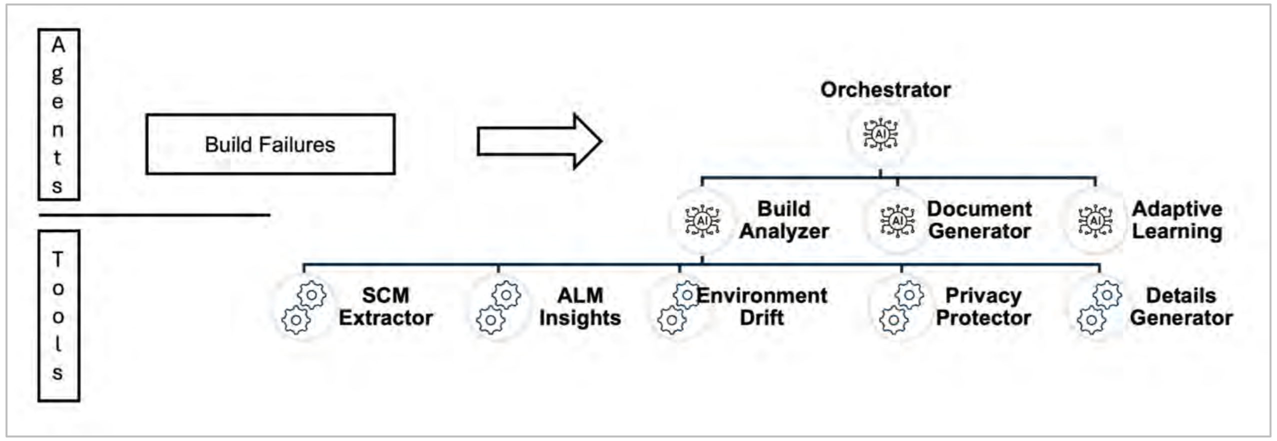

Generality
Performance

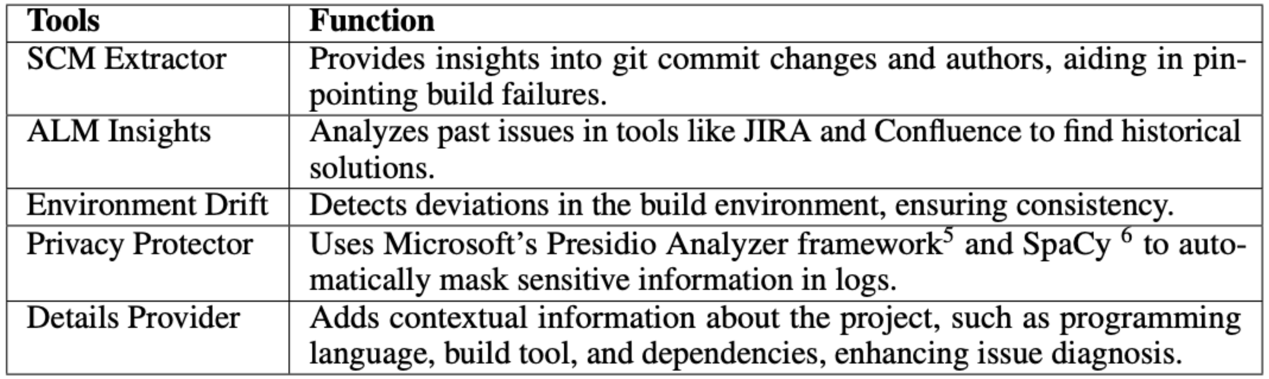
Level	Techniques	Performance	Capabilities	Key Characteristics	Use Cases	Narrow Domain	General Wide-Range Domain
5	LLM-based AI + Tools (Intent + Actions + Reasoning & Decision Making + Memory + Reflection + Autonomous Learning + Generalisation + Personality (Emotion + Character) + Collaborative behaviour (Multi-Agents)	Superhuman > 100% of Skilled Adults	True Digital Persona	Agent represents the user in completing affairs, interacts on behalf of user with others, ensuring safety & reliability.	Agent acts on behalf of user to complete tasks, interacting with others while ensuring safety & reliability.	Superhuman Narrow-AI AlphaFold, AlphaZero, StockFish	Artificial Super Intelligence (ASI) Not yet achieved
4	LLM-based AI + Tools (Intent + Actions + Reasoning & Decision Making + Memory & Reflection + Autonomous Learning + Generalisation	Virtuoso Equal to 99% of Skilled Adults	Memory & Context Awareness	Agent senses user context, understands user memory, and proactively provides personalised services at times.	A personalised virtual assistant enhances UX by understanding context & memory while acting proactively.	Virtuoso Narrow-AI AlphaGo, Deep Blue	Virtuoso AGI Not yet achieved
3	LLM-based AI + Tools (Intent + Actions) + Reasoning & Decision Making + Memory & Reflection	Expert Equal to 90% of Skilled Adults	Strategic task Automation	Using user-defined tasks, agents autonomously plan, execution steps using tools, iterates based on intermediate feedback until completion.	Agents autonomously plan and execute steps based on intermediate feedback	Expert Narrow-AI Purpose build, specific task orientated Agents	Expert AGI Not yet achieved
2	IL/RL-based AI + Tools (Intent + Actions) + Reasoning & Decision Making	Competent Equal to 50% of Skilled Adults	Deterministic Task Automation of Skilled Adults	Based on user description of deterministic task, agent auto-completes steps in predefine action.	User: "Check the weather in Beijing today".	Competent Narrow-AI Conversational AI build frameworks with LLM, RAG, etc.	Competent AGI Not yet achieved
1	Rule-Based AI + Tools (Intent + Actions)	Emerging Equal to Unskilled Humans	Simple Step Sequence	Agents complete tasks following exact steps, pre-defined by users or developers.	User: "Open Messenger" User: "Open the first unread email in my mailbox and read its content" User: "Call Alice".	Emerging Narrow-AI Single Rule-based systems, SHRDLU, GOFAI	Emerging AGI ChatGPT, Gemini, Llama 2, etc.
0	No AI Tools (Intent + Rules + Actions)	No AI	No AI	No AI	No AI	Narrow Non-AI UI Driven Software	General Non-AI Human-In-The-Loop Computing Mechanical Turk

Adapted From: https://arxiv.org/pdf/2405.06643

COBUS GREYLING & AI

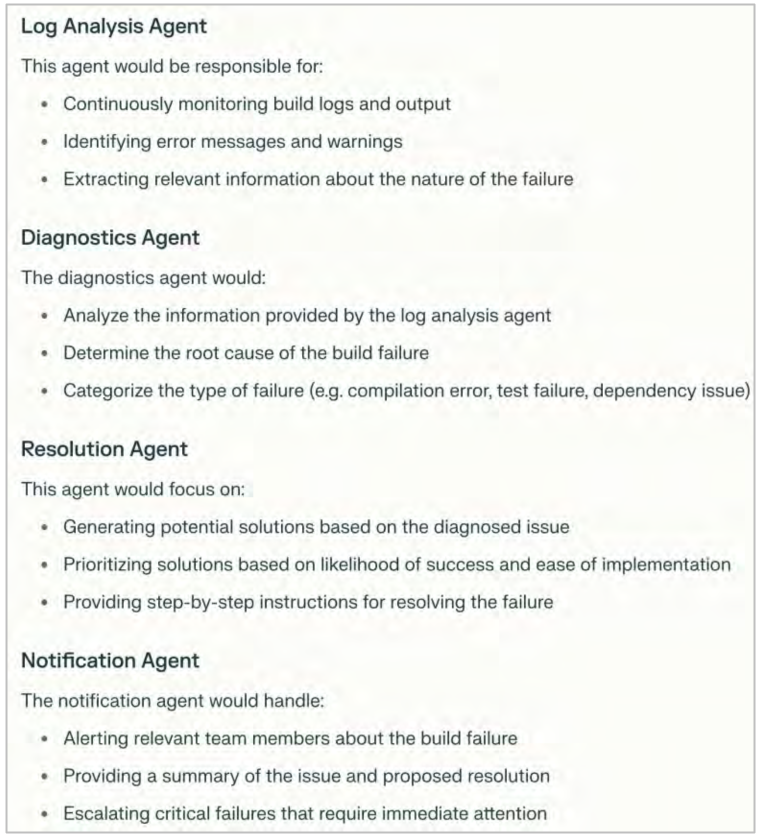
Multi-Agent System





Tools	Function
SCM Extractor	Provides insights into git commit changes and authors, aiding in pin-pointing build failures.
ALM Insights	Analyzes past issues in tools like JIRA and Confluence to find historical solutions.
Environment Drift	Detects deviations in the build environment, ensuring consistency.
Privacy Protector	Uses Microsoft's Presidio Analyzer framework[5] and SpaCy [6] to automatically mask sensitive information in logs.
Details Provider	Adds contextual information about the project, such as programming language, build tool, and dependencies, enhancing issue diagnosis.

Build Analyzer Agent



Log Analysis Agent

This agent would be responsible for:

- Continuously monitoring build logs and output
- Identifying error messages and warnings
- Extracting relevant information about the nature of the failure

Diagnostics Agent

The diagnostics agent would:

- Analyze the information provided by the log analysis agent
- Determine the root cause of the build failure
- Categorize the type of failure (e.g. compilation error, test failure, dependency issue)

Resolution Agent

This agent would focus on:

- Generating potential solutions based on the diagnosed issue
- Prioritizing solutions based on likelihood of success and ease of implementation
- Providing step-by-step instructions for resolving the failure

Notification Agent

The notification agent would handle:

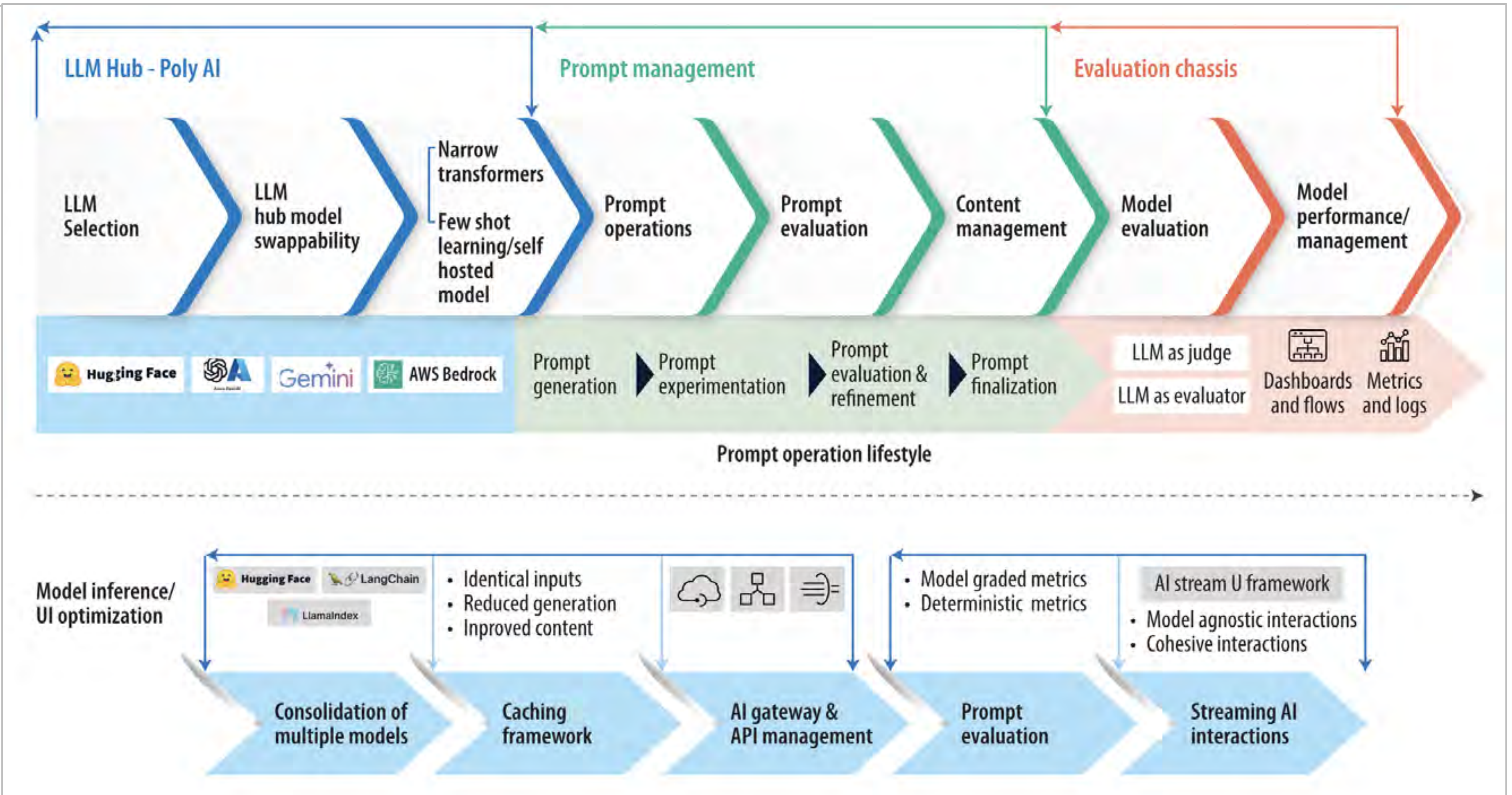
- Alerting relevant team members about the build failure
- Providing a summary of the issue and proposed resolution
- Escalating critical failures that require immediate attention

Challenges and addressing them..

- LLM selection impacts accuracy
- When LLM changes, existing prompts are not effective anymore

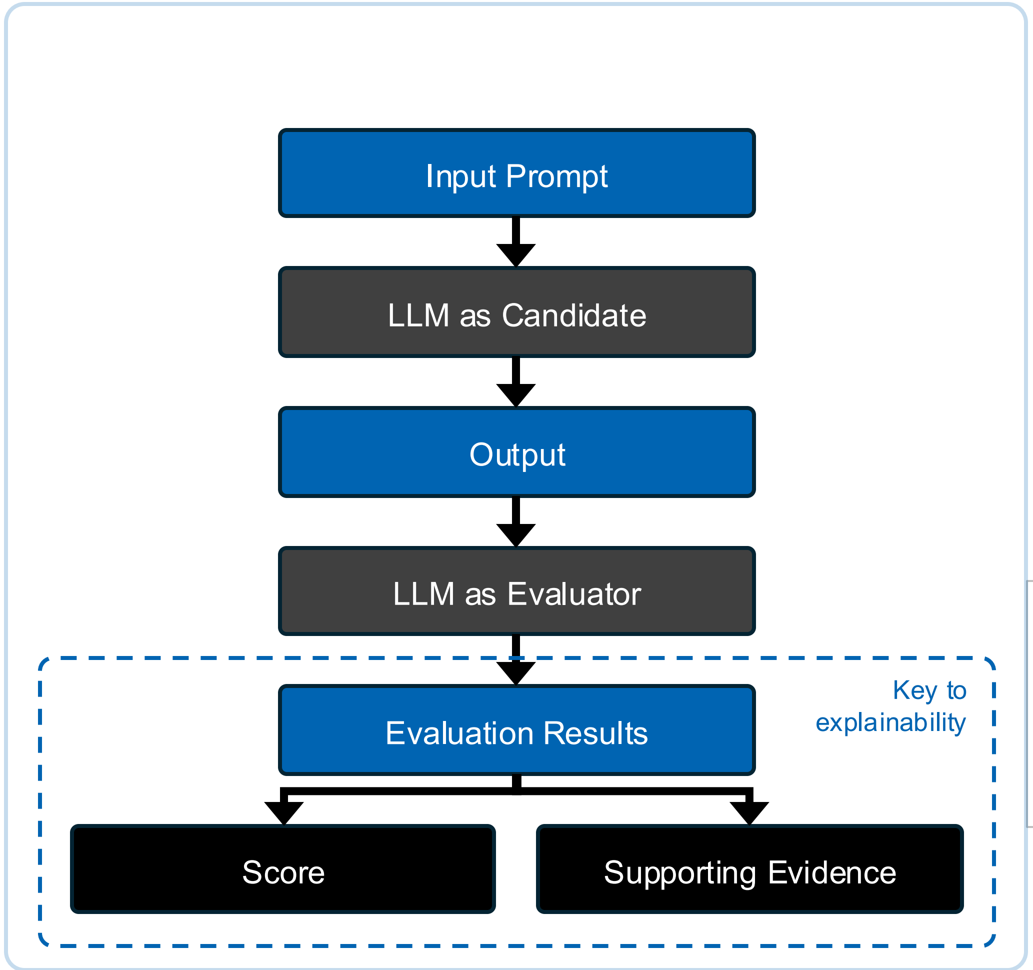
LLM Version	Accuracy
GPT-3.5	82%
GPT-4.0	90%
GPT-4o Mini	68%

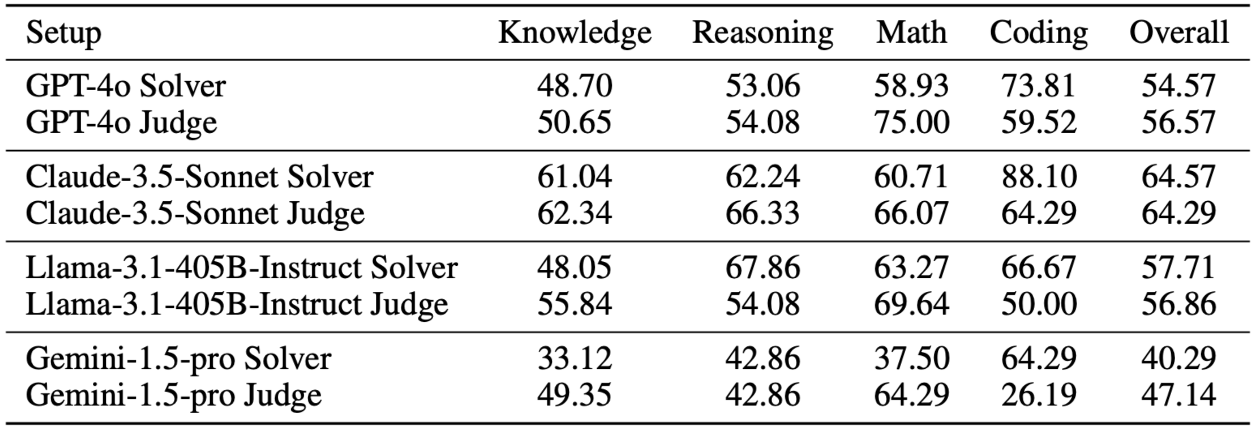
- Human evals are not scalable
- Reasoning is key
- Frontier models cannot be fine-tuned



Source: https://www.infosys.com/iki/research/top10-ai-imperatives-2025.html

LLM-as-an-Evaluator





Setup	Knowledge	Reasoning	Math	Coding	Overall
GPT-4o Solver	48.70	53.06	58.93	73.81	54.57
GPT-4o Judge	50.65	54.08	75.00	59.52	56.57
Claude-3.5-Sonnet Solver	61.04	62.24	60.71	88.10	64.57
Claude-3.5-Sonnet Judge	62.34	66.33	66.07	64.29	64.29
Llama-3.1-405B-Instruct Solver	48.05	67.86	63.27	66.67	57.71
Llama-3.1-405B-Instruct Judge	55.84	54.08	69.64	50.00	56.86
Gemini-1.5-pro Solver	33.12	42.86	37.50	64.29	40.29
Gemini-1.5-pro Judge	49.35	42.86	64.29	26.19	47.14

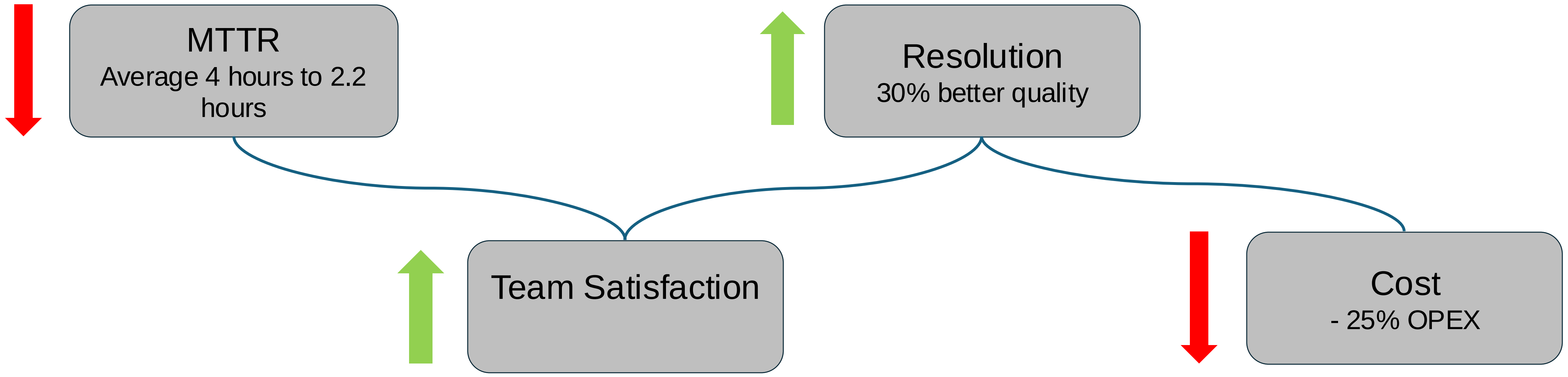
Source https://openreview.net/pdf?id=G0dksFayVq

Key observation –

Ability of the judge to verify the solution pairs is highly correlated with its ability to solve the problem itself.

Key Conclusions and Results

January 2024 to September 2024 in over 100 applications


MTTR
Average 4 hours to 2.2 hours
Resolution
30% better quality
Team Satisfaction
Cost
- 25% OPEX