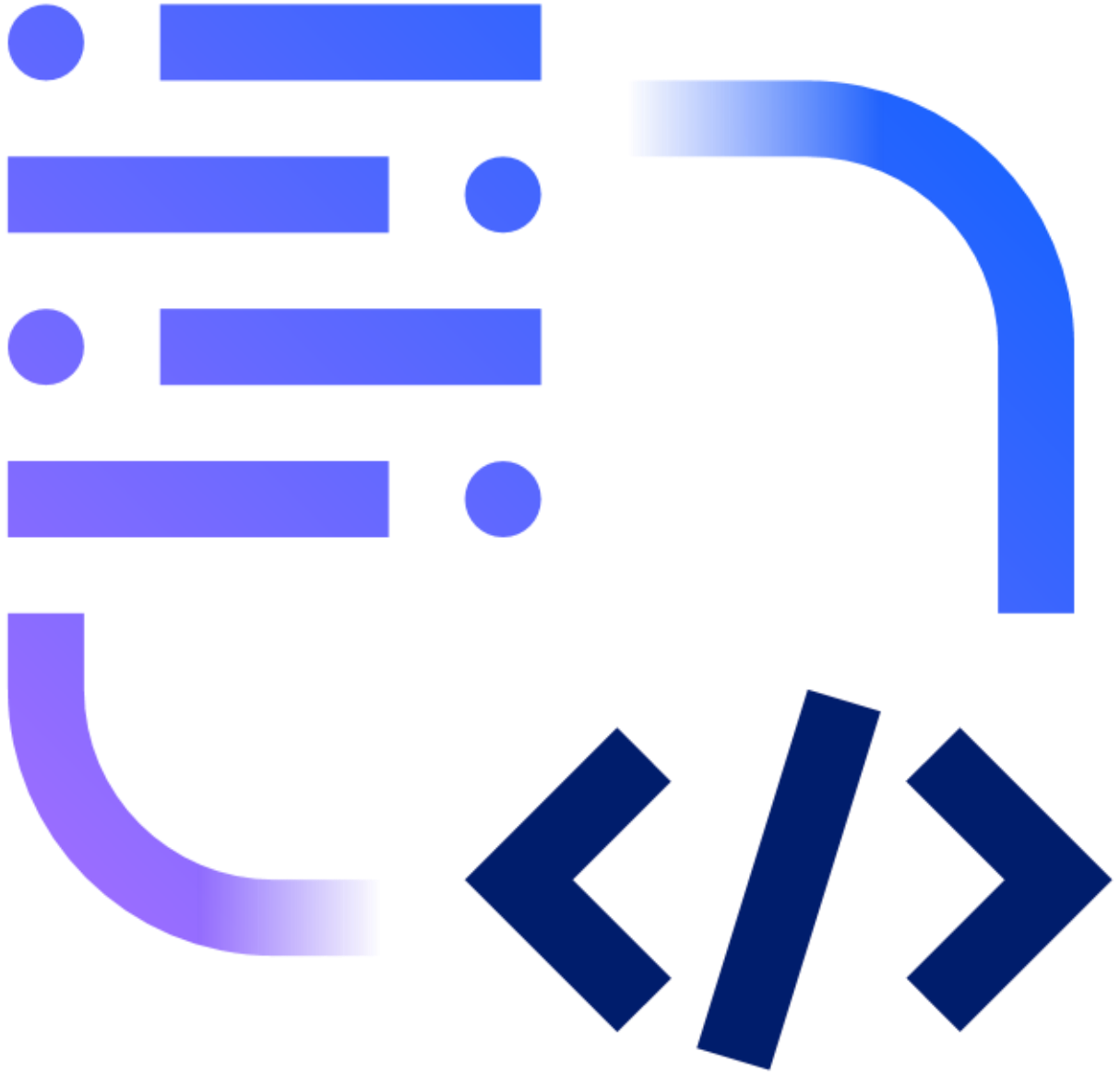
AI, Agents and Software Engineering -

What's Happening and Where Are We Heading?

Atul Kumar (IBM Research)

Sridhar Chimalakonda (IIT Tirupati)



Presenters

- Atul Kumar
  - AI for Code, IBM Research – India
  - Also worked at Microsoft, Accenture, ABB, Siemens
  - MTech and PhD, IIT Kanpur
  - kumar.atul@in.ibm.com
  - https://research.ibm.com/people/atul-kumar

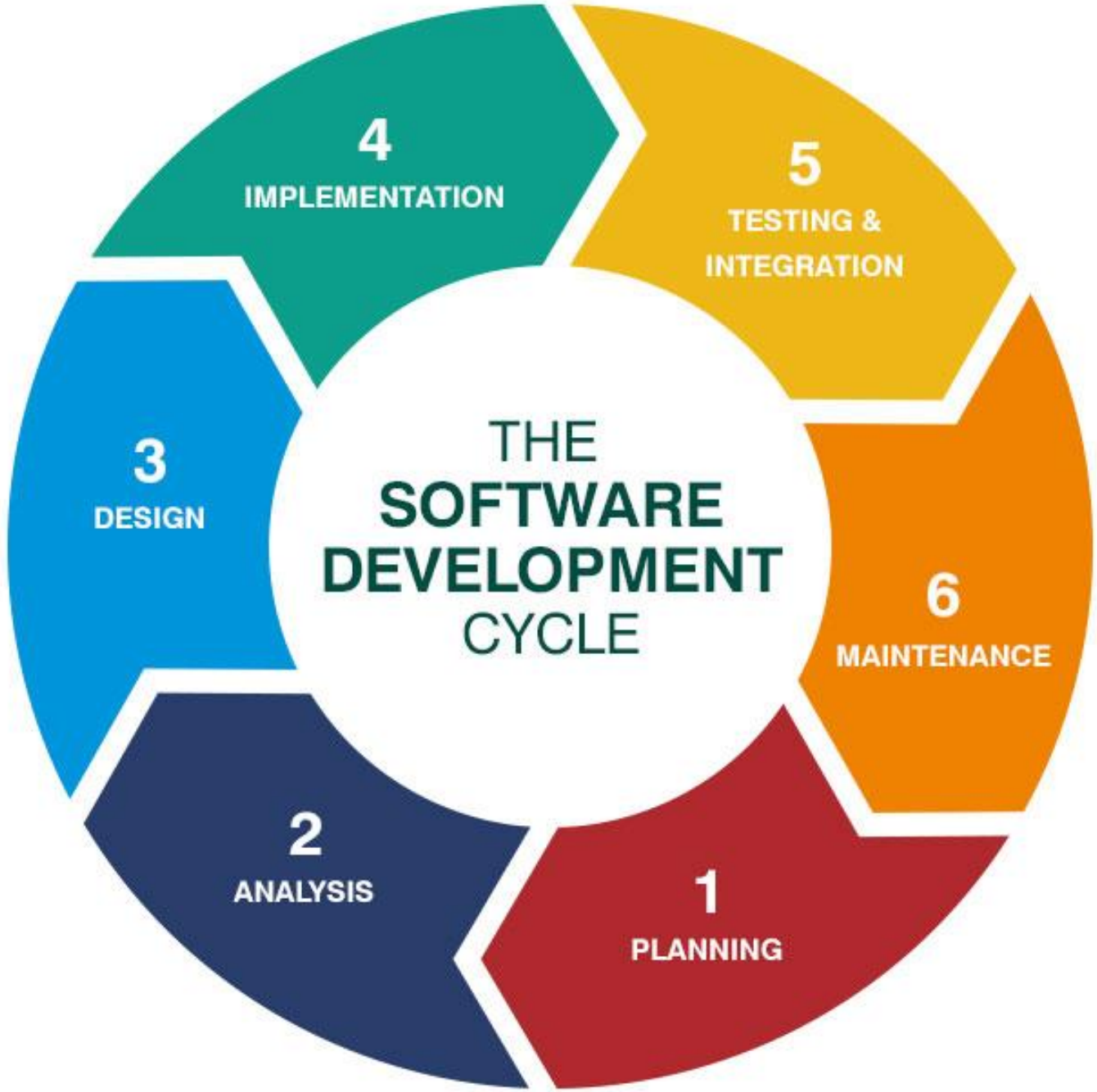
- Sridhar Chimalakonda
  - Head of the Department, CSE Dept., IIT Tirupati
  - Research in Intelligent Software & Human Analytics Lab (RISA Lab)
  - MS & PhD, IIIT Hyderabad
  - ch@iittp.ac.in
  - https://rishalab.in/

Tutorial Goals

- Provide an overview of the AI for SE landscape, including merits and demerits.
- Analyze state-of-the-art AI tools and agent-based systems.
- Present current limitations and industrial case studies.
- Discussion with experts who are leading/guiding the use of AI, Agents for enterprise SDLC
- Discuss open problems and emerging research opportunities.

Outline

- Introduction (Atul) – 20m
- Landscape of AI for Software Engineering Tasks (Sridhar) – 30m
- AI Agents in Software Engineering (Sridhar, Atul) – 20m
- Case Study / Demo (Atul) – 20m
- Panel Discussion: How is Industry looking at the recent disruptions caused by AI and Agents in SDLC – 45m
  - Panel members
    - RD Naik (Professor of Practice, COEP Tech, formerly Chief Scientist , TCS Research)
    - Prabhat Shankar (Industrial AI Solutions Lead at ABB)
    - Suman Roy (Consulting Member of Technical Staff, Health and AI Group, Oracle)
  - Moderator: Sridhar Chimalakonda
- Discussion and closing – 15m


4
IMPLEMENTATION
5
TESTING &
INTEGRATION
3
DESIGN
THE
SOFTWARE
DEVELOPMENT
CYCLE
6
MAINTENANCE
2
ANALYSIS
1
PLANNING

AI Assisted Software Development Life Cycle (AI SDLC)

LLMs changed how we do Software Engineering

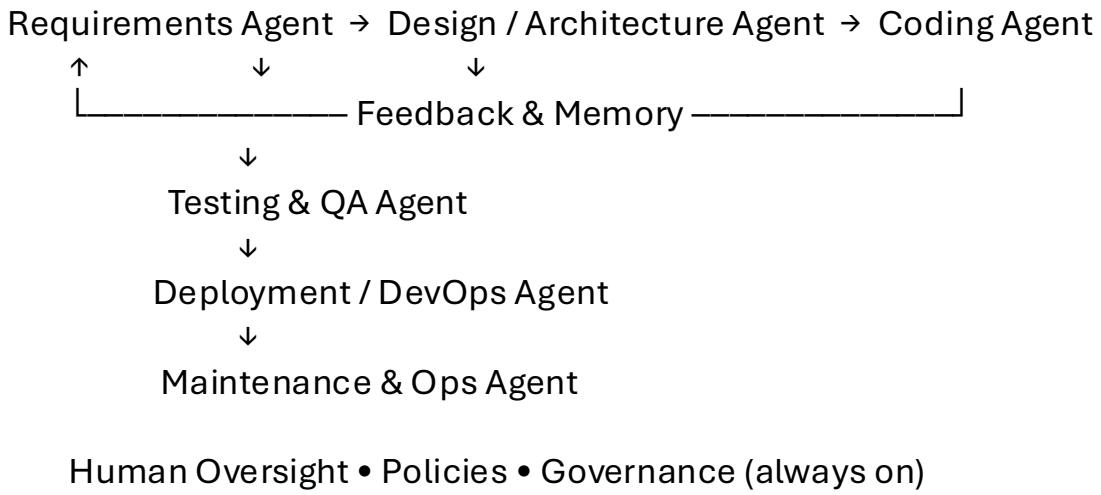
- Coding: completion, refactoring, translation
- Testing: test generation, bug localization
- Requirements: user stories, traceability
- System Design: architecture sketches, ADR drafts
- DevOps & Deployment: CI scripts, IaC, log analysis
- Maintenance: log analysis, code review

But...

❌ Fragmented point solutions
❌ One-shot interactions
❌ No end-to-end ownership

Agentic Software Development Lifecycle

- Requirements Agent
- Design / Architecture Agent
- Coding Agent
- Testing & QA Agent
- Deployment / DevOps Agent
- Maintenance & Operations Agent
- Human Oversight: Policies, Governance layer



Evolution of Automation in Software Engineering

- Rule-based -> ML -> DL -> LLMs -> Agents

SDLC: Before and After Agents

Before: LLMs as Tools

- Coding Assistant
- Test Generation Tool
- Design / Docs Helper
- DevOps Scripts Generator
- Human Integration & Validation

After: Agentic SDLC

- Coordinated AI Agents
- Shared Memory & Planning
- Execution + Feedback Loops
- Risk-based Autonomy
- Human Oversight & Governance

LLM (single prompt) vs Agents for SE

- Task
  Implement a function that reads a CSV of transactions and returns total amount per user.
  - Columns: user_id, amount
  - Ignore malformed rows

Write a Python function that reads a CSV file with columns:
user_id, amount
and returns a dictionary mapping user_id to total amount.
Ignore malformed rows.

```
import csv

def total_amounts(path):
    totals = {}
    with open(path) as f:
        reader = csv.DictReader(f)
        for row in reader:
            user = row['user_id']
            amount = float(row['amount'])
            totals[user] = totals.get(user, 0) + amount
    return totals
```

Issues with the generated code

- What if amount is empty?
- What if amount = "N/A"?
- What if user_id is missing?
- What if there’s a header mismatch?
- What if encoding is wrong?

- The code looks correct
- The prompt never enforced validation
- No tests were run
- No failures were observed

Agentic Workflow

- Developer Agent
  - Writes initial implementation
- Test Agent
  - Generates edge-case tests
- Execution Agent
  - Runs tests
- Fixing Agent
  - Patches failures

Test Agent Generates Tests:

```
def test_malformed_rows():
    input = "user_id,amount\nu1,10\nu2,abc\n,5\nu1,\n"
    assert total_amounts(input) == {"u1": 10.0}
```

Execution Agent Reports:

```
ValueError: could not convert string to float: 'abc'
```

Fixing agent updates code:

```
try:
    amount = float(row['amount'])
except (ValueError, TypeError):
    continue
```

The difference isn’t intelligence — it’s feedback.

Why single-prompt coding is insufficient?

- Single-prompt coding fails not because LLMs are weak, but because software engineering is not a single-step problem.
- Single prompt assumption: “One prompt → one correct solution”
- Reality from practice:
  - Requirements evolve
  - Tests fail
  - Integration breaks
  - Performance regressions appear
- Mismatch
  - Software quality emerges through cycles, not outputs

When was the last time any of you wrote production code that worked correctly on the first attempt?

Why Agents in SE?

- Correctness cannot be verified in the prompt
  - LLMs can:
    - Generate plausible code, Follow syntax and style
  - They cannot (from a prompt alone):
    - Execute code, Run tests, Observe runtime behavior, Measure performance
  - Consequence
    - Single prompts optimize for plausibility, not correctness
- No Feedback = No Learning
  - In traditional SE:
    - Compiler errors, Test failures, Code reviews, Runtime logs
  - In single-prompt coding:
    - Zero feedback loop, No correction signal
  - Without feedback, LLM output quality plateaus quickly.
- Cross-Artifacts reasoning
  - Real issues span:
    - Code + tests, Code + configuration, Code + environment
  - Single sees only what you paste
  - Agent can inspect repo, tests, configs, logs
- Roles: Analyst, developer, tester, reviewer
- Multiple agents can take different responsibilities - separation of concerns

Why AI, Agents + SE Now?

- Explosion of:
  - Code
  - Repositories
  - Requirements artifacts
- LLMs can reason across all of them
- Shift:
  - From tools -> assistants -> agents
- Extensive use of AI across the entire software development lifecycle (SDLC)
- New Capabilities: LLMs and autonomous agents creating new automation opportunities (eg GitHub Copilot)
- Shift from Tool-assisted -> AI-assisted -> AI-driven

Landscape of AI for Software Engineering Tasks – Status, Challenges and Future Directions

Sridhar Chimalakonda
Associate Professor & Head
Department of Computer Science & Engineering
Indian Institute of Technology Tirupati, India
Adjunct Associate Professor, University of Waterloo

Research in Intelligent Software & Human Analytics (RISHA) Lab

ch@iittp.ac.in

IIT Tirupati Campus (A glimpse)

Not generated by Generative AI/LLM

RiSHA LAB

RESEARCH IN INTELLIGENT SOFTWARE AND HUMAN ANALYTICS

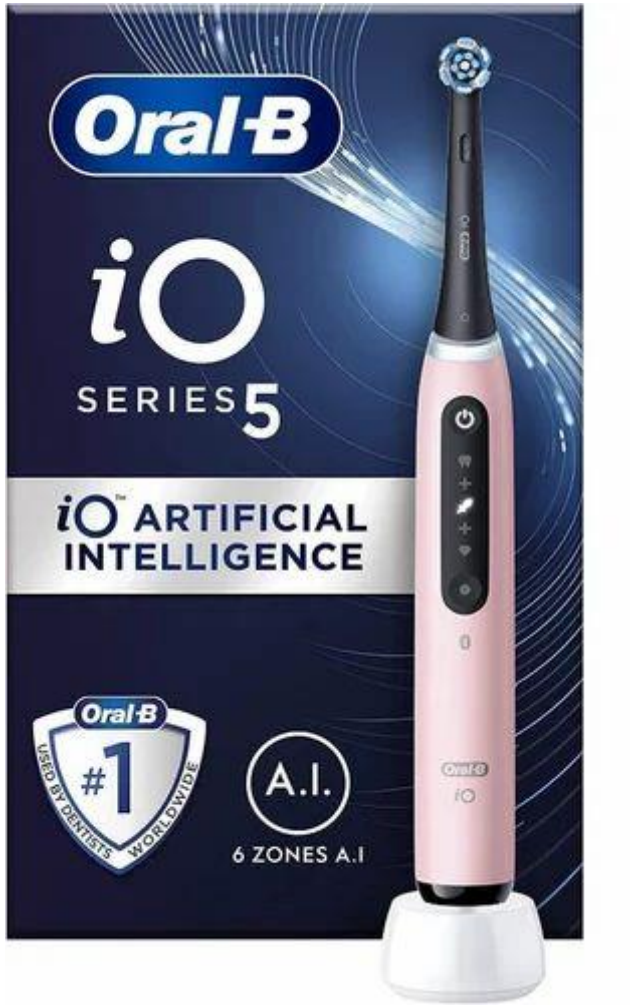
🎯 *15 research works with undergrads as first authors*

AR Storytelling for Śrīmad-Bhāgavatam: An Immersive Experience Through Time and Scale

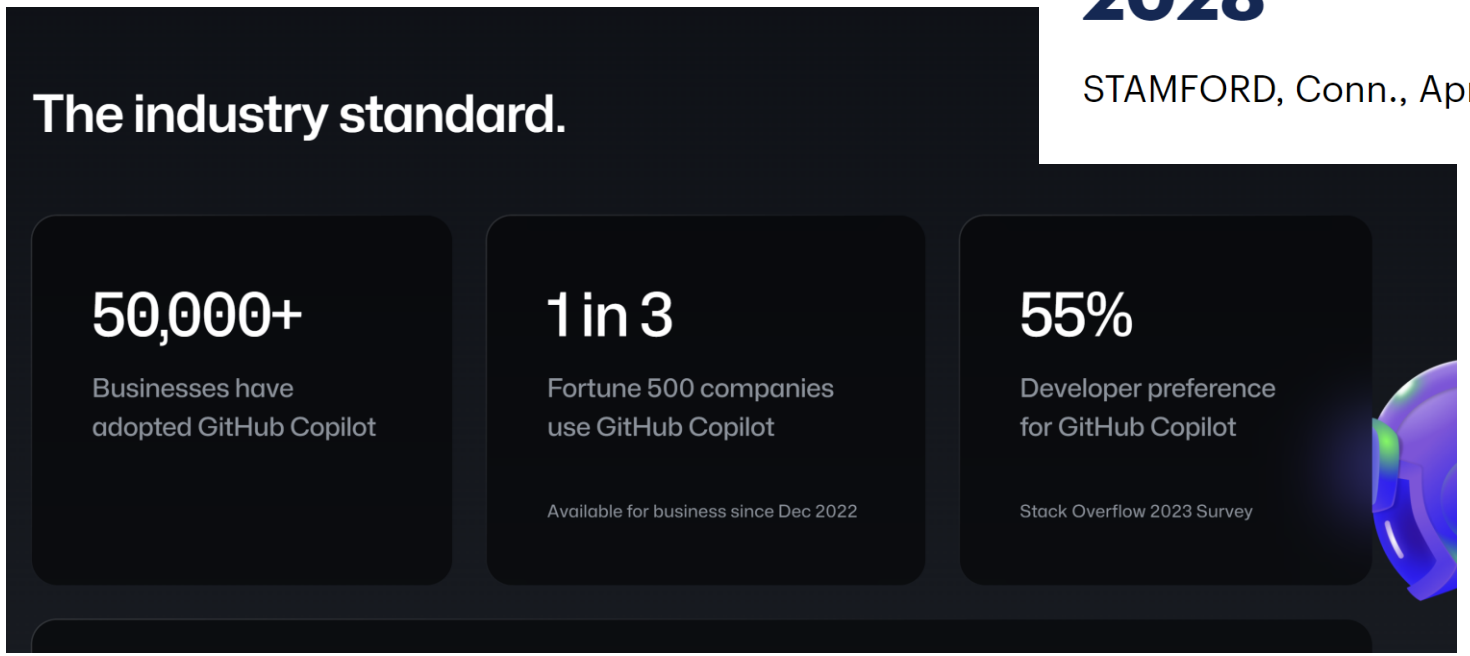
- ~4 Key Research Themes
  - Software Documentation & Legacy Modernization
  - Green Software Engineering
  - Source Code Representation, Bugs
  - Code Smells, Patterns, Anti-Patterns
- ~20+ Software Tools
- ~ Publications in multiple tracks - ICSE , FSE, ASE, MSR, EASE, ICSME…
- + Lots of explorations and unpublished ideas!

ICSE 2025 (Canada), FSE 2025 (Norway), EASE 2025 (Turkey)

Software (*with AI*) is Ubiquitous* Today!



Software Development (Industry) is Changing Today due to Advances in Generative AI



Newsroom / Information Technology / Press Release

Gartner Says 75% of Enterprise Software Engineers Will Use AI Code Assistants by 2028

STAMFORD, Conn., April 11, 2024

When to use and when not to use the capabilities of Generative AI for software development?

Source: https://github.com/features/copilot

Tech Industry > Artificial Intelligence

Google now uses AI to write 25% of its new code — Alphabet CEO Sundar Pichai underlines the company's role in the AI industry amidst strong Q3 24 financials

News By Hassam Nasir published October 31, 2024

Will AI completely displace entry-level software engineers?

TECH

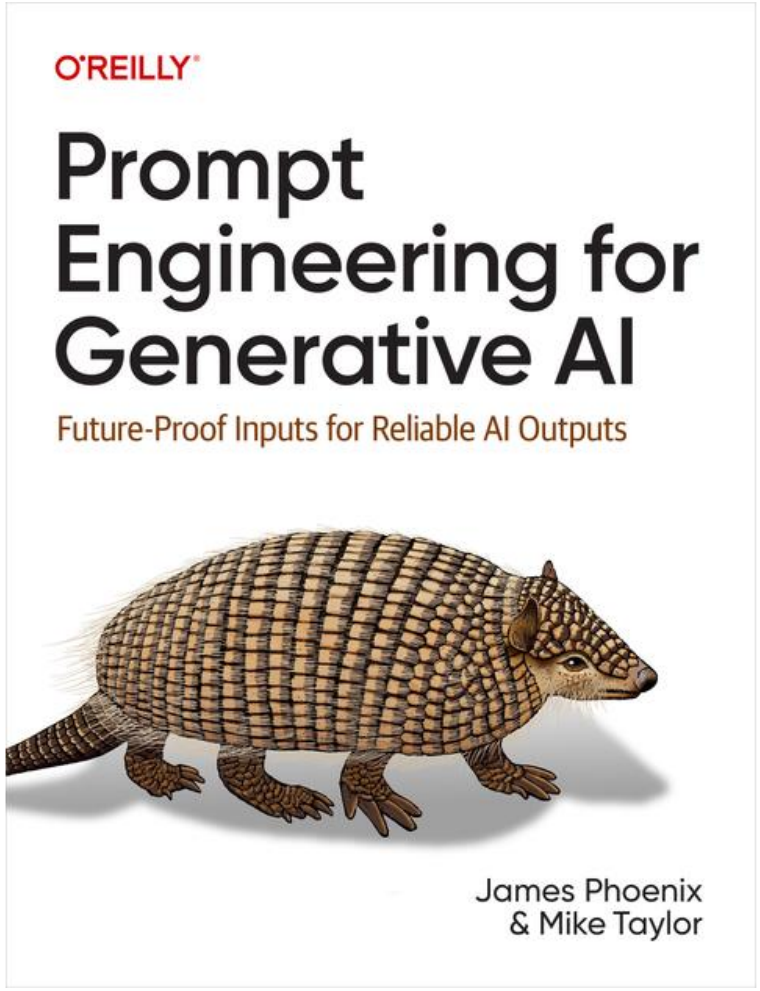
Satya Nadella says as much as 30% of Microsoft code is written by AI

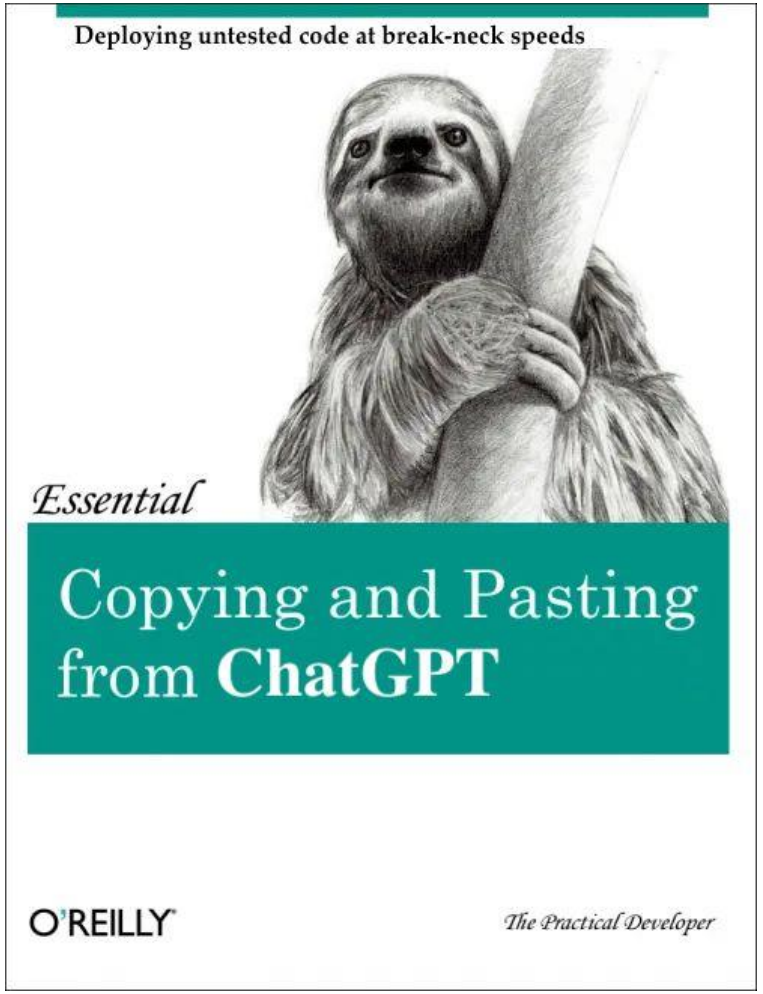
PUBLISHED TUE, APR 29 2025•9:33 PM EDT | UPDATED TUE, APR 29 2025•9:58 PM EDT

Jordan Novet @JORDANNOVET | Jonathan Vanian @IN/JONATHAN-VANIAN-B704432/

SHARE

The Common Tools of Software Professionals Today ☹





Why AI is able to support Software Engineering today?

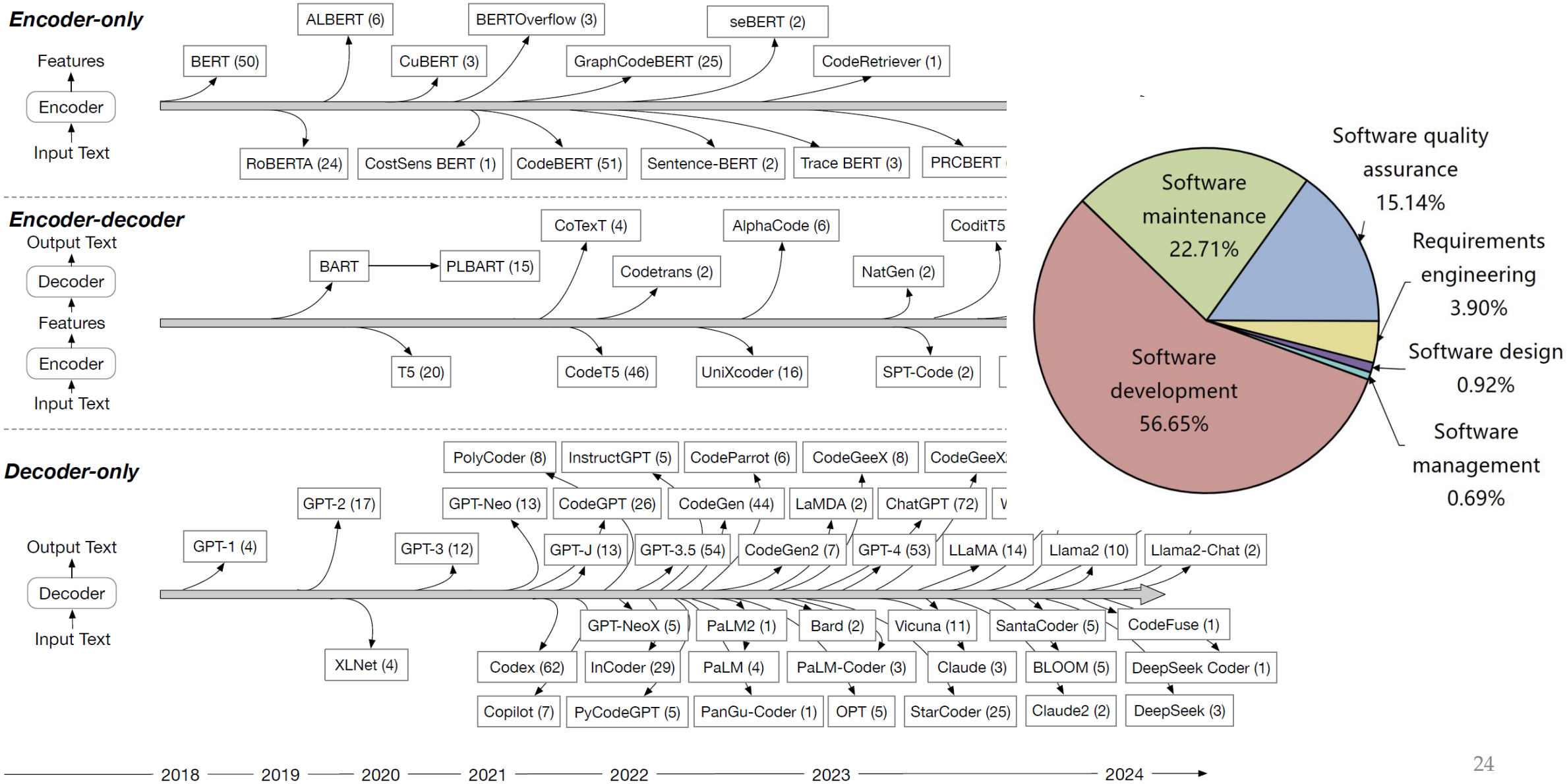
- Millions of software repositories
- Source code, comments
- Readme files, docs
- Pull Requests, Commits, Issues
- Metadata (stars, contributors, time data…)
- Stack Overflow, Issue Trackers...

Image Credit: https://www.nvidia.com/en-gb/deep-learning-ai/solutions/

Can Generative AI automate the software engineer's work?

Large Language Models for Software Engineering: A Systematic Literature Review

Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, Haoyu Wang



GenAI for Requirements

1. Elicit requirements and generate an initial requirements specification
2. Analyze requirements for consistency, completeness and remove ambiguities
3. Resolve conflicting requirements
4. Prioritize requirements

- *Are the extracted requirements from GenAI tools correct? Do they have bias?*
- *Lack of verification and validation of the outputs of the GenAI tools for requirements.*
- *Ethical and privacy concerns of requirements from a business perspective*
- *How do they fit in the context of mission-critical, safety-critical systems and domains such as healthcare, defense, finance?*

GenAI for Software Architecture and Design

1. Select architecture patterns for specific requirements, constraints and context.
2. Generate architecture and design diagrams
3. Update architecture based on codebase
4. Decision making wrt to ***quality trade-offs*** and prioritization of technology decisions

- Largely underexplored!
- Lack of availability of training data (such as architecture descriptions, design diagrams, decisions, case studies)
- Unavailability of domain-specific data and decisions at architecture level in the public domain

Can developers/architects share their code base and business requirements?

Agents for Coding

- *Code {Comprehension, Generation, Code Completion, Summarization, Search}*

- **Observations & Opportunities**
  - Significant productivity and quality gains!
  - Automate repetitive coding tasks
  - Accelerate debugging and bug fixing
  - ***Lack of domain and organization specific GenAI tools***

- Challenges
  - Unreliable and unmaintainable code
  - Hidden quality and technical debt issues
  - Security vulnerabilities
  - Lack of verification and validation tools for AI generated code
  - Licensing, ethical and privacy concerns
  - Lack of prompt standards – Which prompts work, when and why?
  - Integration with existing code and tools

GitHub CoPilot, CodeT5 (Salesforce), PaLM-Coder, Code Llama, Tabnine, Kite, CodeGuru, PyCharm with AI Assistant, Visual Studio IntelliCode, Codota, and OpenAI Codex.

Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you. (0% complete)

If you'd like to know more, you can search online later for this error: UNEXPECTED KERNEL MODE TRAP

AI-generated code is largely buggy, unsecure and unreliable!

A Survey of Bugs in AI-Generated Code

RUOFAN GAO, School of Mathematical and Computational Sciences, Massey University, New Zealand
AMJED TAHIR, School of Mathematical and Computational Sciences, Massey University, New Zealand
PENG LIANG, School of Computer Science, Wuhan University, China
TEO SUSNJAK, School of Mathematical and Computational Sciences, Massey University, New Zealand
FOUTSE KHOMH, SWAT Laboratory, Polytechnique Montréal, Canada

Developers are widely using AI code-generation models, aiming to increase productivity and efficiency. However, there are also quality concerns regarding the AI-generated code. The generated code is produced by models trained on publicly available code, which are known to contain bugs and quality issues. Those issues can cause trust and maintenance challenges during the development process. Several quality issues associated with AI-generated code have been reported, including bugs and defects. However, these findings are often scattered and lack a systematic summary. A comprehensive review is currently lacking to reveal the types and distribution of these errors, possible remediation strategies, as well as their correlation with the specific models. In this paper, we systematically analyze the existing AI-generated code literature to establish an overall understanding of bugs and defects in generated code, providing a reference for future model improvement and

"You still have to study" - On the Security of LLM generated code

Stefan Götz
Offenburg University of Applied Sciences
andreas.schaad@hs-offenburg.de

Andreas Schaad
Offenburg University of Applied Sciences
andreas.schaad@hs-offenburg.de

ABSTRACT

We witness an increasing usage of AI-assistants even for routine (classroom) programming tasks. However, the code generated on basis of a so called "prompt" by the programmer does not always meet accepted security standards. On the one hand, this may be due to lack of best-practice examples in the training data. On the other hand, the actual quality of the programmers' prompt appears to influence whether generated code contains weaknesses or not.

In this paper we analyse 4 major LLMs with respect to the security of generated code. We do this on basis of a case study for the Python and Javascript language, using the MITRE CWE catalogue

caution when using such tools. The right questions (i.e. "prompts need to be asked to avoid the AI assistant generating code wit potential weaknesses.

The idea of educating (student) developers in following a "shif left" security approach now also needs to be extended to includ carefully crafting prompts. Therefore, we analyse to which degre the quality of a prompt impacts the security of generated code. W apply different prompt engineering techniques as well as specific i structions regarding security-related problems. In this way, we ca evaluate the extent to which the various AI assistants independentl consider security-related aspects, at what level of complexity the require support in the form of hints or questions, and when the

Generative AI for Software Engineering! – The Critical Lens

- *Requirements Engineering*
- *Software Architecture and Design*
- *Software Implementation (Coding)*
- *Software Testing*
- *Software Maintenance*
- *…*
- *GenAI for Practitioners*

- ***Key Observations and Challenges***
  - *Non-deterministic and Unreliable outputs*
  - *No standardized inputs, process and outputs*
  - *Licensing and reliability issues*
  - *Lack of guards for developers on usage of AI tools*
  - *Lack of verification and validation of AI tools*

[1] Nguyen-Duc, A., Cabrero-Daniel, B., Przybylek, A., Arora, C., Khanna, D., Herda, T., ... & Abrahamsson, P. (2023). Generative Artificial Intelligence for Software Engineering--A Research Agenda. *arXiv preprint arXiv:2310.18648*.

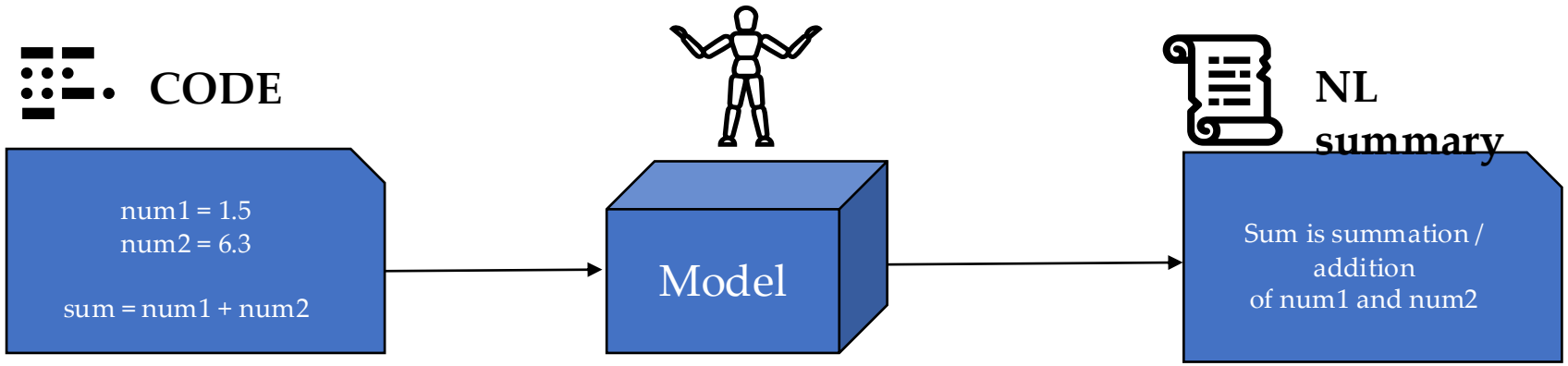
1

Code Summarization without Direct Access to Code - Towards Exploring Federated LLMs for Software Engineering

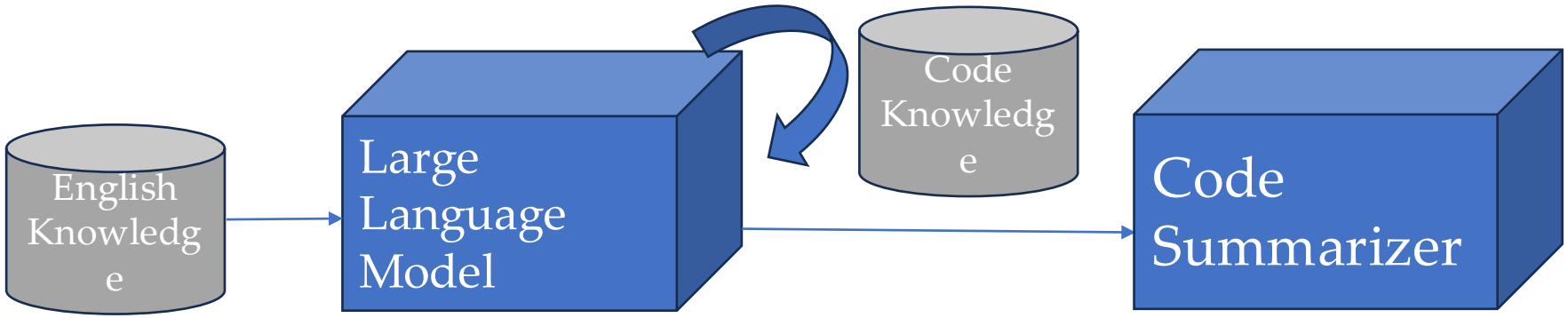
FSE 2022, EASE 2024

Task @ hand: Code Summarization

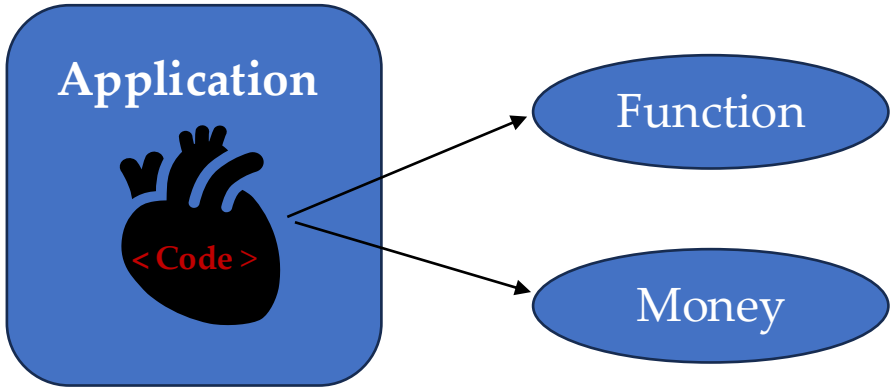
What?

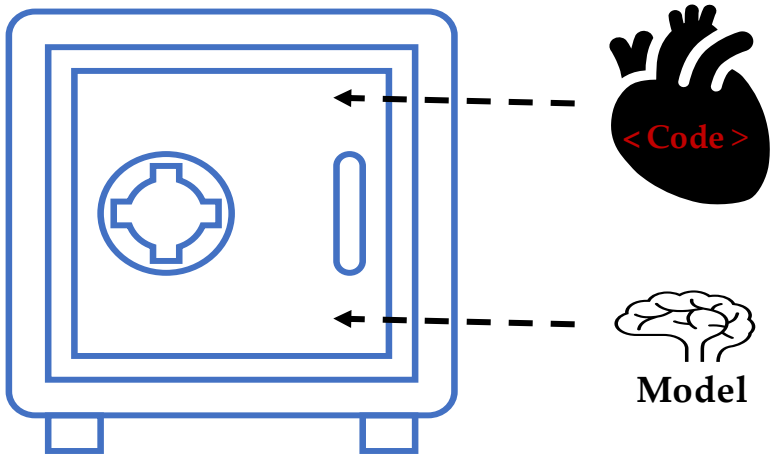


How?



Code is sensitive (private)

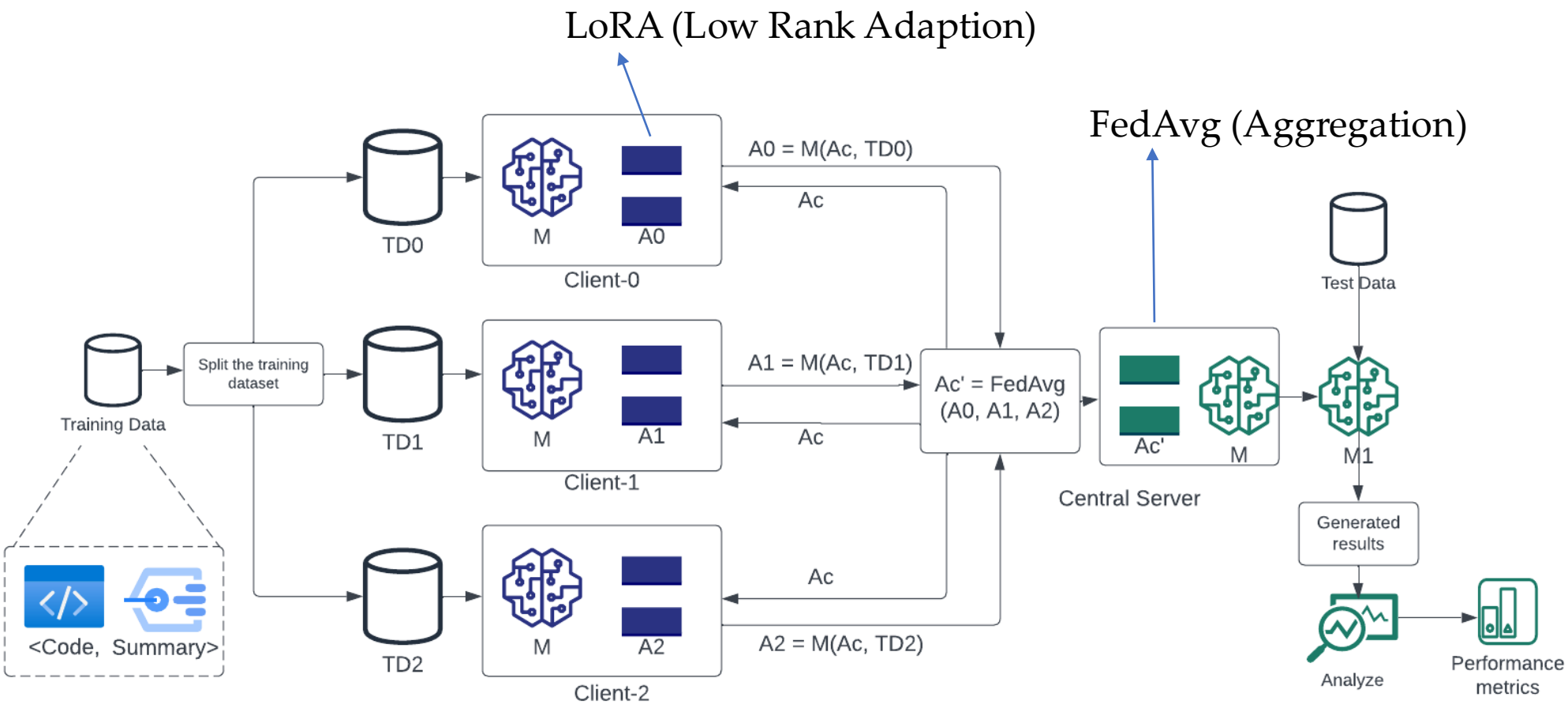




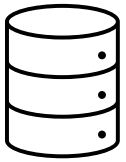
- Code is private!
- What if I don't have enough data to train an LLM ? Can I collaborate with another partner?

Can we generate natural language summary from code without sharing code?

A FedLLM Approach for Code Summarization



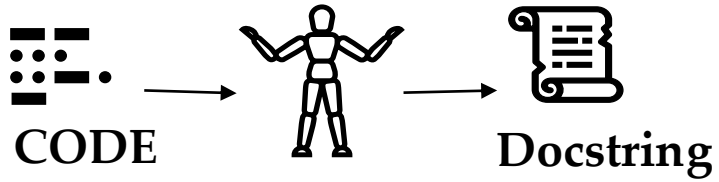
Evaluation



Code Summarization Dataset:

Python [code-docstring-corpus]

109,108 training triples, 2,000 validation triples, and 2,000 test triples



★★☆

Results:

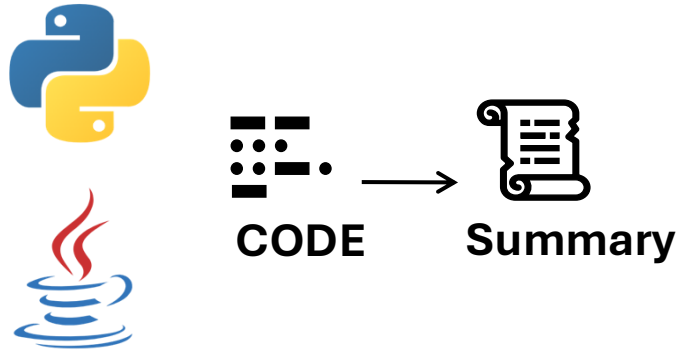
- Optimal hyperparameters: LoRA of rank **8** targeted on key and query projections.
- Performance: **Fed** > Centralized > Vanilla
- Illustrative example: **Fed is closer to ground truth**.
- Fed@BEST (7) shows 3%, 12%, 16% increase in BLEU, METEOR, ROUGE-L compared to PreTrained model.

Model:

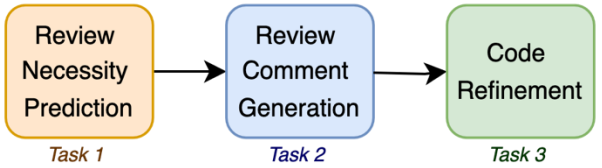
- Llama-2 LLM
- Fine-tune on chosen Dataset: Fed, Non-Fed (for comparison)

Does it work for other code tasks?

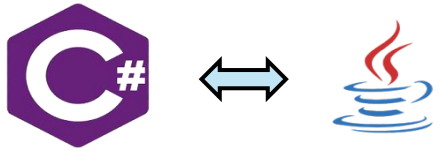
1. Code Summarization without Direct Access to Code - Towards *Exploring* Federated LLMs for Software Engineering (EASE 2024)
2. Collaborative Code Summarization Via FedLLM -An *Empirical* Study



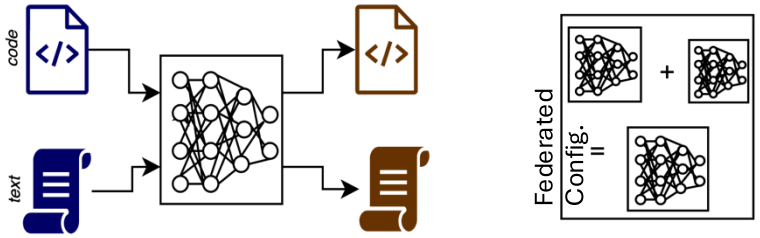
3. Code Review Automation Via Multi-task Federated LLM - An *Empirical* Study



4. Code Translation through Federated LLM - An Empirical Study



5. LLM2FedLLM - A *Tool* for Simulating Federated LLMs for Software Engineering Tasks (ICPC 2025)



2

This article was published on: 02/11/22

Home / Enterprise IT / COBOL code base in production hits 800+ billion lines

Enterprise IT Read This

COBOL code base in production hits 800+ billion lines

Legacy Software Systems

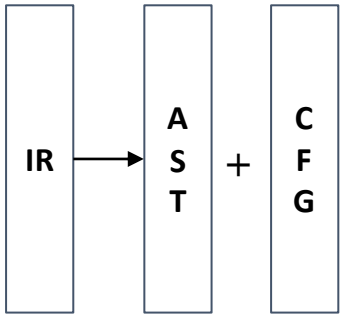
- 93% of ATMs still use COBOL
- Android Lolipop version (2014) has 2.8 million active users
- Python 2.7 to Python 3
- Multiple versions in parallel

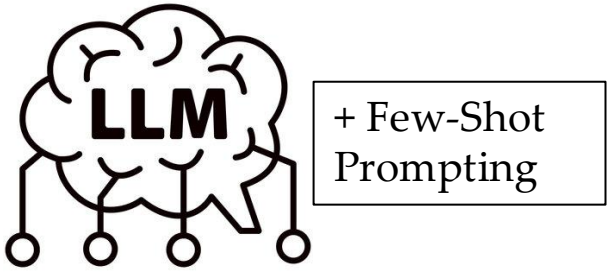
Challenges

- Difficult to understand and debug
- No design diagrams, comments
- Obsolete dependencies
- Poor documentation
- Software developers move on!

How to comprehend and maintain Legacy Software Systems (e.g. Banking, Finance?)

Our COBOL Tool Landscape





COBREX: AST + CFG Rule-Based Method

A-COBREX: COBREX + Intermediate Representation of Business Rules

COBRAIN: Can LLMs help?

1 ICSME 2022

2 ICSE 2025

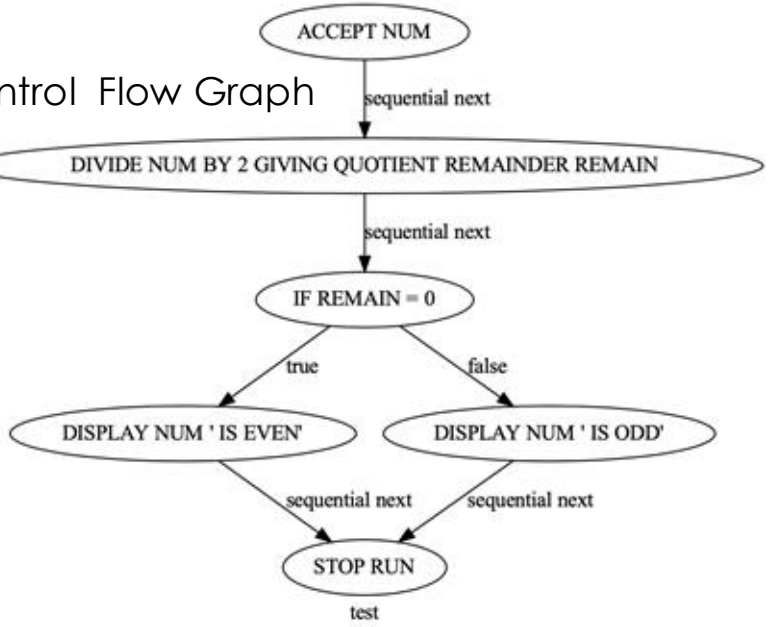
3 EASE 2025

COBREX

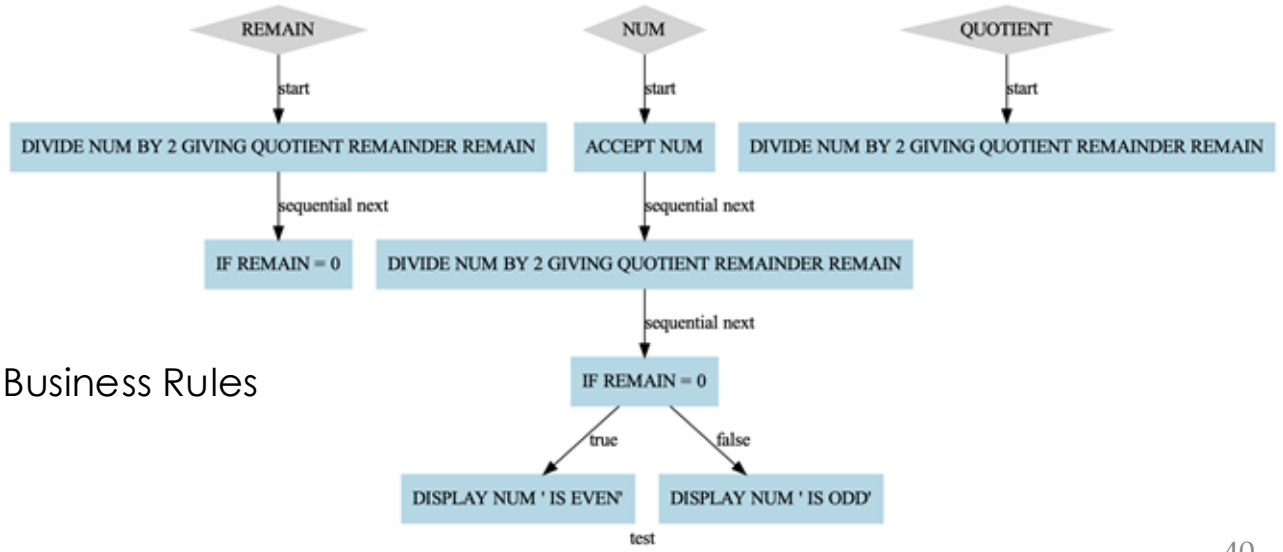
Source code (.CBL or .COB)

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EXAMPLE.
DATA DIVISION.
WORKING-STORAGE SECTION.
77  NUM          PICTURE 99.
77  QUOTIENT     PICTURE 99.
77  REMAIN       PICTURE 9.
PROCEDURE DIVISION.
    ACCEPT NUM.
    DIVIDE NUM BY 2 GIVING QUOTIENT REMAINDER REMAIN.
    IF REMAIN = 0
            DISPLAY NUM ' IS EVEN'
    ELSE
            DISPLAY  NUM ' IS ODD'
    END-IF.
    STOP RUN.
```

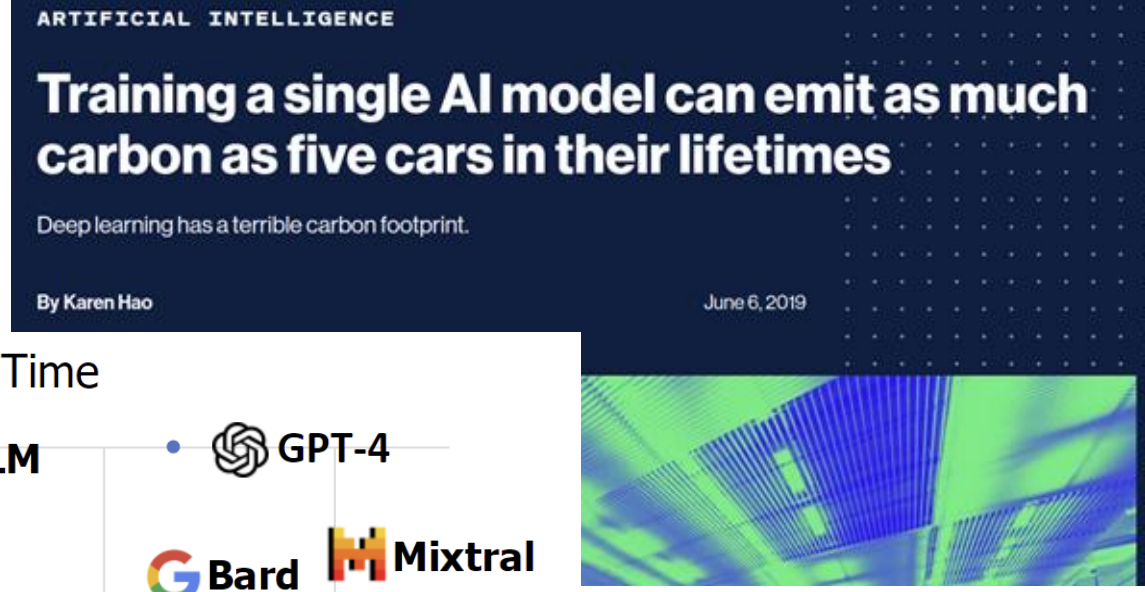
Control Flow Graph

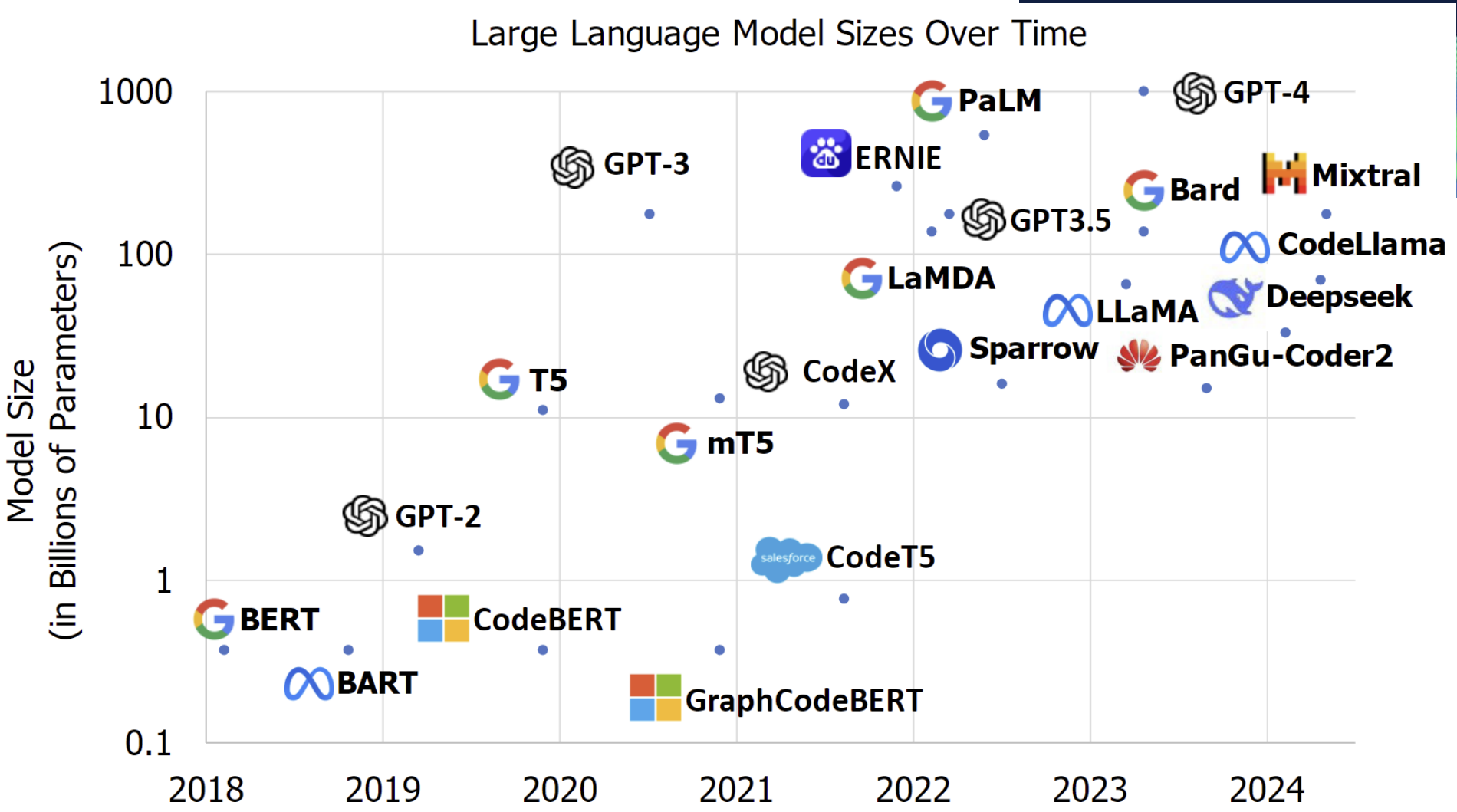


Business Rules



3 *The Grand Challenge* Energy-Hungry LLMs



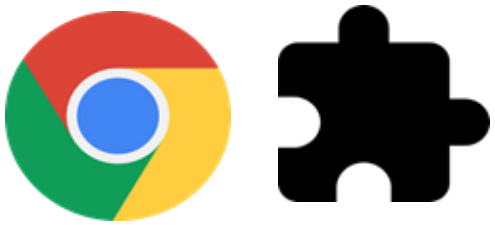


The Software Lens!

Deep Learning, Dataframe Libraries, Energy ⚡

8 energy patterns in deep learning development

1 EASE 2022



DENT - A Tool for adding energy tags to Stack Overflow questions on DL

2 FSE 2023

GreenNet catalog of 12 energy patterns in deep learning development

3 TOSEM*

Energy comparison of Dataframe Libraries

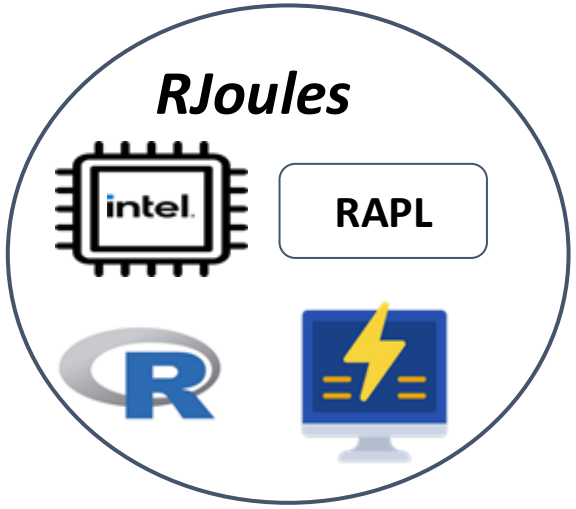
4 **MSR 2023**

NLP, Databases, PLs and Energy ⚡

V/s

spaCy

V/s



V/s

DBJoules

V/s

NLP Libraries, Energy Consumption, and Runtime - An Empirical Study	*RJoules: An Energy Measurement Tool for R*	*R vs Python - An Exploratory Study on Energy Consumption of Machine Learning Algorithms*	*Towards Comprehending Energy Consumption of Database Management Systems - A Tool and Empirical Study*
1 FSE 2025	2 ASE 2023	3	4 EASE 2024

When to use and when not to use the capabilities of Generative AI for software development?

The AI Regulation/Policy Landscape

Who will take responsibility for software systems that are built using AI?

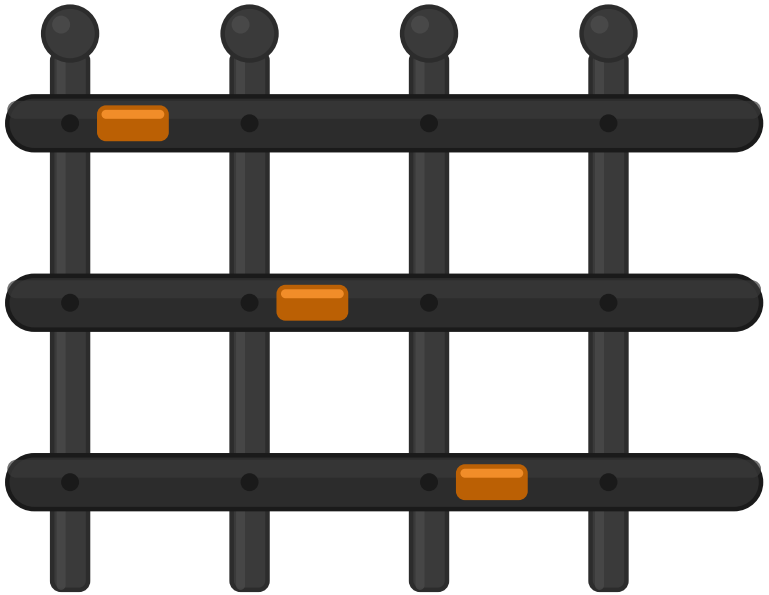
NIST AI Risk Management Framework

सत्यमेव जयते

INDIAai
A MEITY INITIATIVE

Standardization!

- ISO/IEC SC42 – Artificial Intelligence
- ISO/IEC SC7 – Software and Systems Engineering

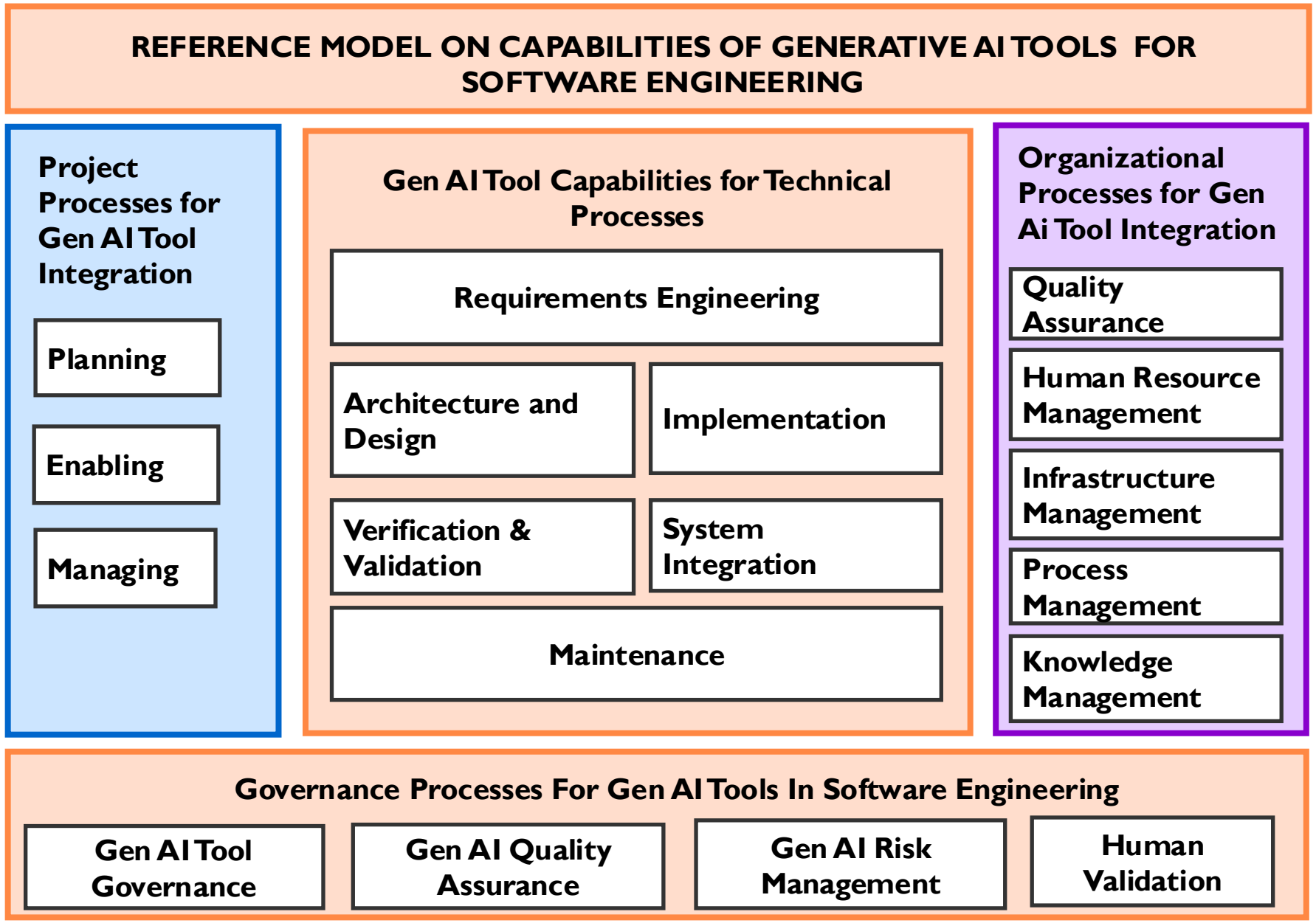


ISO/IEC JTC1/SC7
AHG9 (AI-Assisted Software Development)

Do we have standards for AI-Assisted Software Development?

- **No standards for GenAI tool makers in SE context**
  - Tool vendors (GitHub, OpenAI, Amazon, etc.) have no guidance!
  - How should tools integrate with SDLC processes and existing SE tools?
- **No standards for GenAI tool users in SE context**
  - Developers/organizations have no guidance!
  - **How to systematically evaluate** and use or not use GenAI tools for SE use cases?
  - **How to measure the effectiveness** of GenAI tool adoption?

New Project in ISO/IEC SC7 (Software and Systems Engineering)



Can Generative AI automate the software engineer's work?

1. Improve software development process, quality and support software professionals throughout the lifecycle
2. Decision making support
3. 10x, 100x, 1000x productivity?

What AI and Agents cannot do for SE? (yet!)

- Explainability, Fairness, Accountability, Transparency, Bias, Ethics, Legal, Social Issues!
- High Computation Resources
- Lack of consensus on privacy and security issues
- Human-driven software engineering tasks
- Generalized vs Domain-specific!

Ethics and Information Technology (2024) 26:38
https://doi.org/10.1007/s10676-024-09775-5

ORIGINAL PAPER

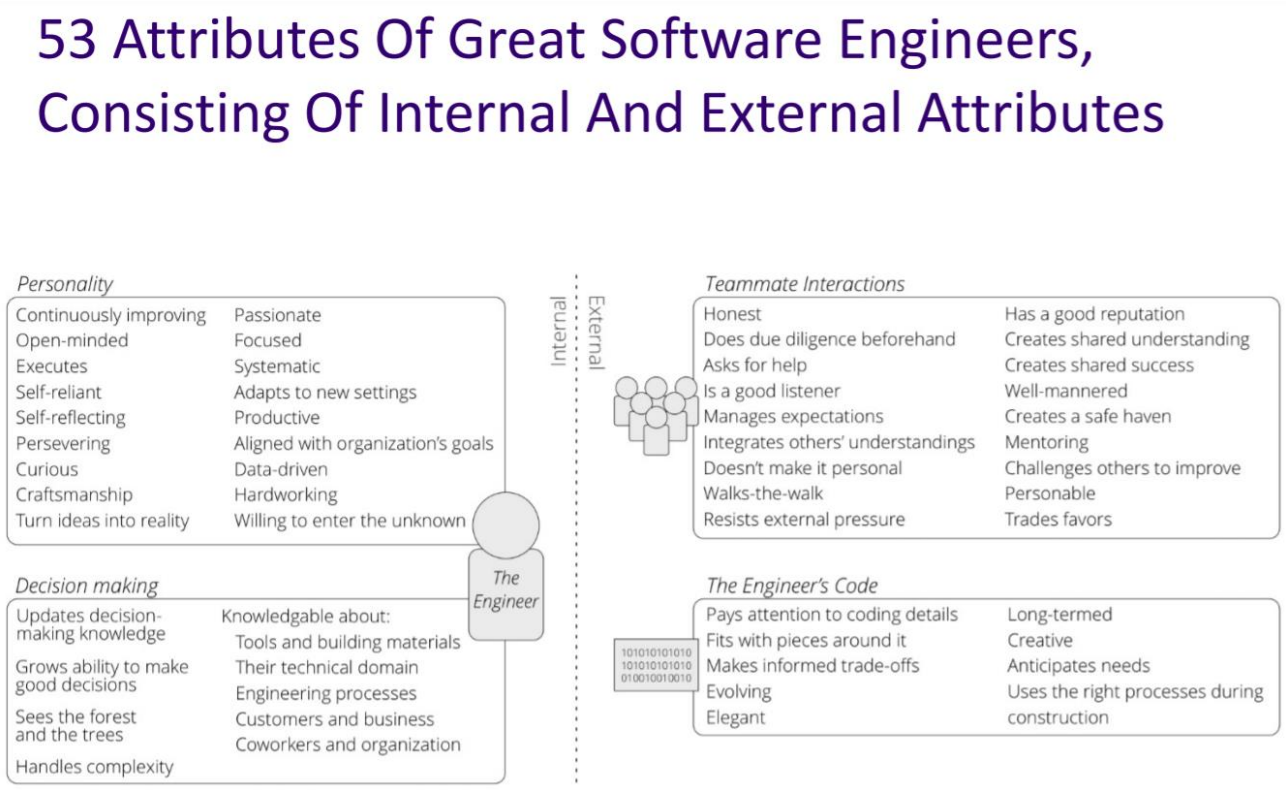
ChatGPT is bullshit

Michael Townsen Hicks[1] · James Humphries[1] · Joe Slater[1]

Abstract
Recently, there has been considerable interest in large language models: machine learning systems which produce human-like text and dialogue. Applications of these systems have been plagued by persistent inaccuracies in their output; these are often called "AI hallucinations". We argue that these falsehoods, and the overall activity of large language models, is better understood as *bullshit* in the sense explored by Frankfurt (On Bullshit, Princeton, 2005): the models are in an important way indifferent to the truth of their outputs. We distinguish two ways in which the models can be said to be bullshitters, and argue that they clearly meet at least one of these definitions. We further argue that describing AI misrepresentations as bullshit is both a more useful and more accurate way of predicting and discussing the behaviour of these systems.

Why is it hard?

- *Ill*-formed Vs Well-Formed problems
- No physical artifacts
- Lack of clarity - **Imprecise and Uncertainty is common!**
- Mind boggling complexity
- Failures often but not tolerable
- Change is expected rapidly

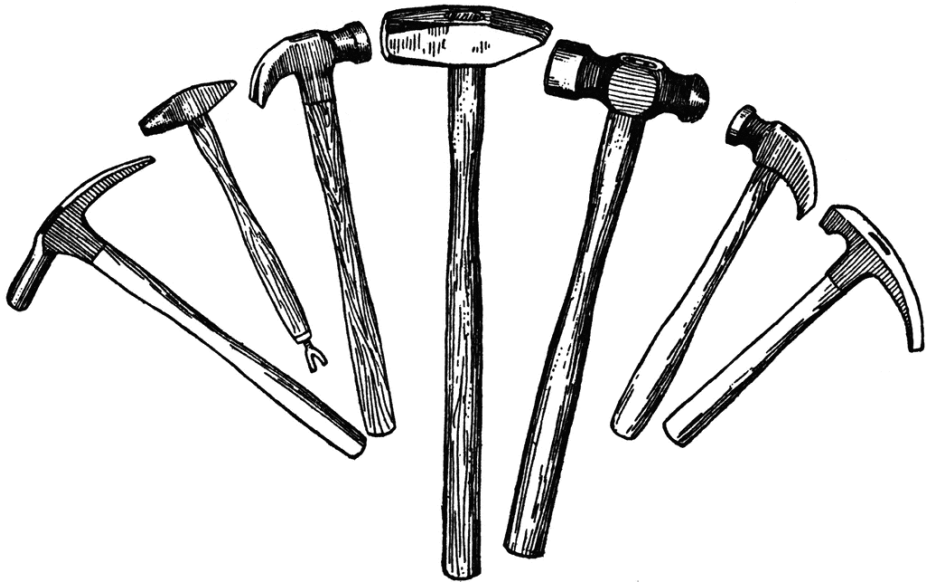


What are the key take-aways?

1. Agents are good but cautiously used!
   - Guard Rails, V&V, Agentic Maturity Models!
2. Non-AI
   AI
   Non-AI + AI
   - LLMs for Society (with caution)
3. Can AI and Agents replace jobs?
   - 10x, 100x, 1000x productivity? But still many open challenges!

An Important Takeaway

- "If you have a hammer, you tend to see every problem as a nail" – Abraham Maslow
- Can we use the hammer of AI and Agents only if necessary?



Can we design Responsible and Ethical AI+SE?

"Necessity is the mother of invention"

Creativity is the father

Passion, Curiosity & Originality are siblings

Capability & Copability are cousins

Inventions & Innovations are heirs!!!

[while luck is the best friend]

Thank you

☺ ☺ ☺ ☺

We are hiring! (Faculty, MS/PhD)

RISHA Lab ↠

Research in Intelligent Software & Human Analytics Lab

भारतीय प्रौद्योगिकी संस्थान तिरुपति

IIT TIRUPATI

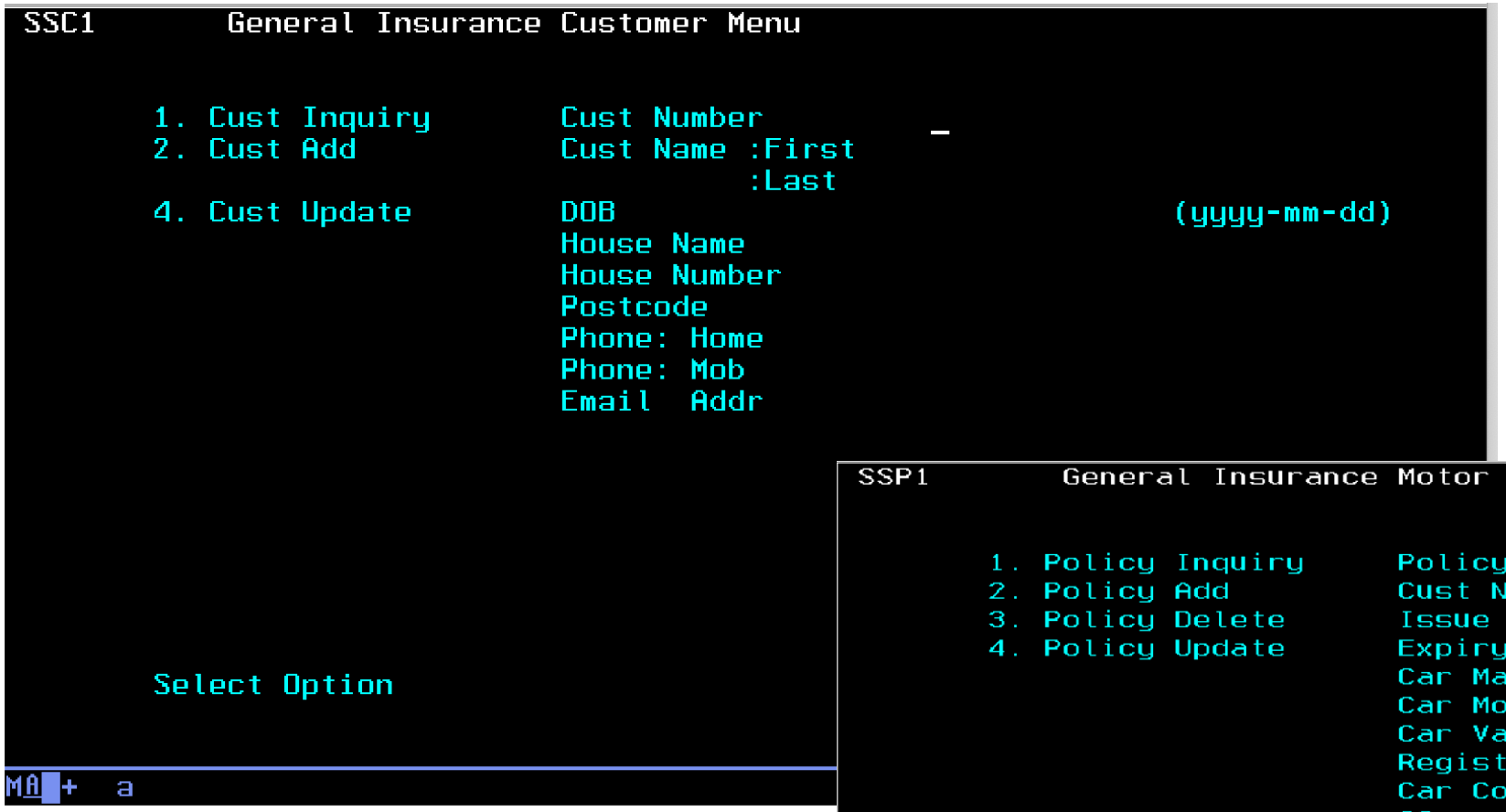
Comments & Collaborations

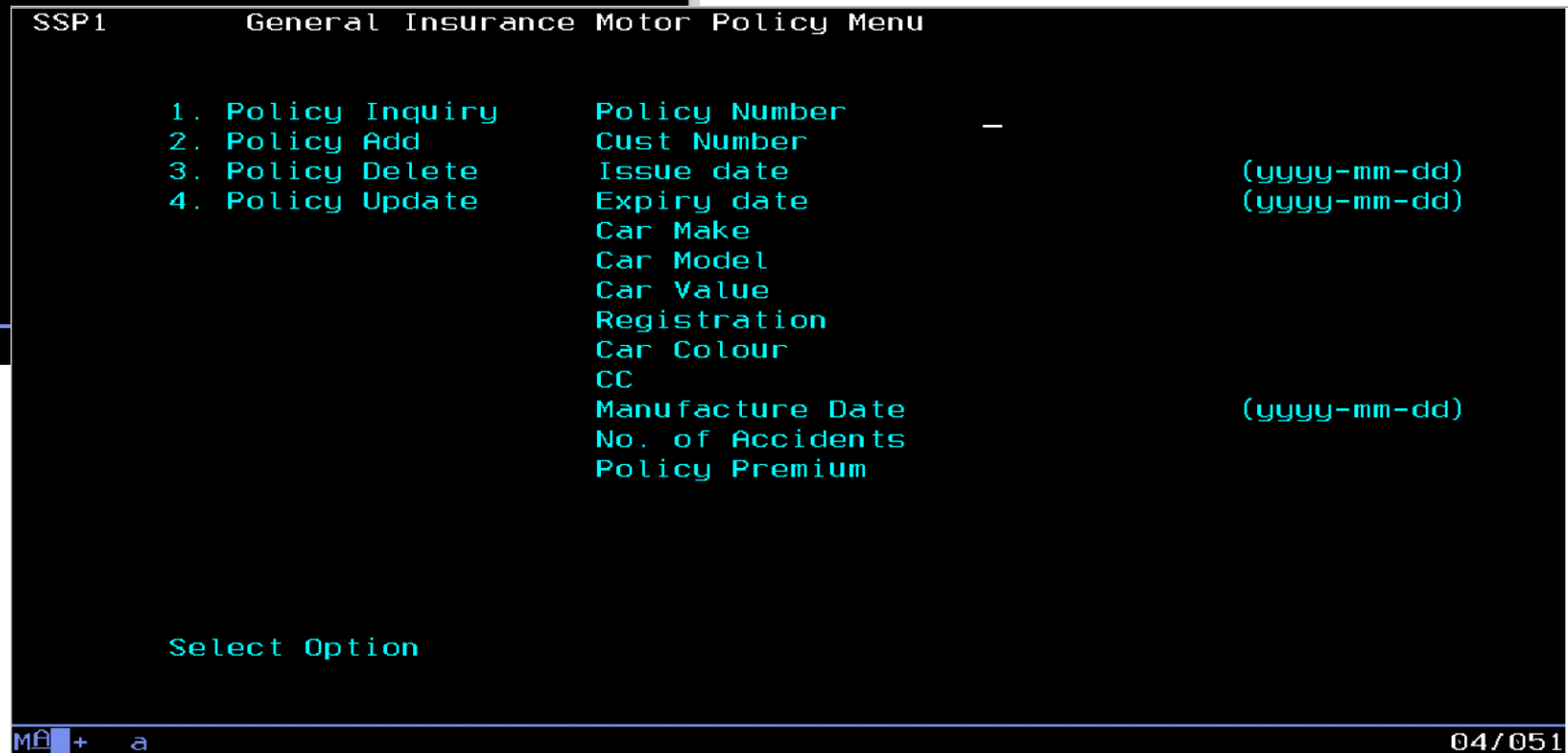
ch@iittp.ac.in | rishalab.in

Case Studies

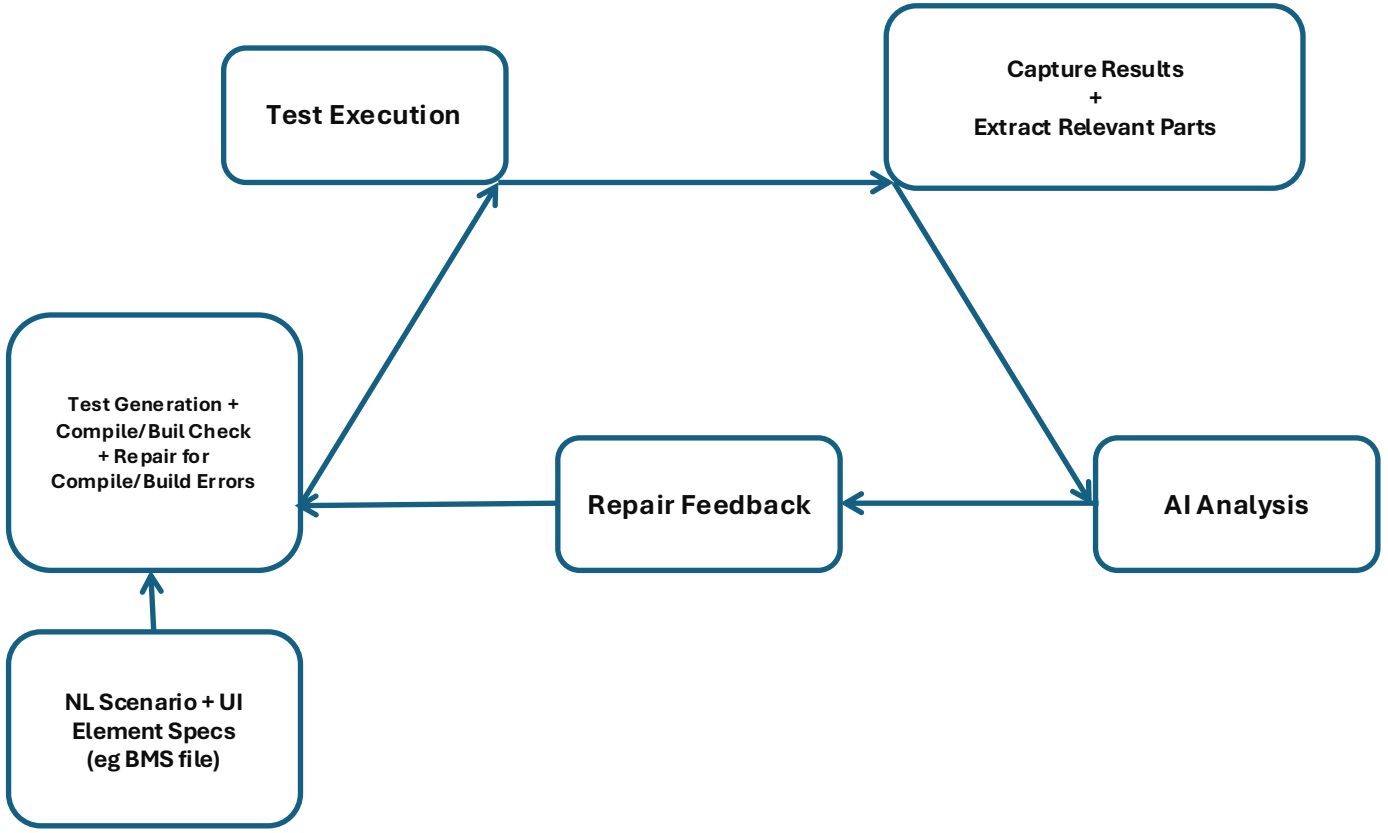
Functional Tests Generation for IBM Z Mainframe Applications (online 3270 Screen)

Example Mainframe 3270 Screens – Genapp





Test Generation And Improvement with AI Agents

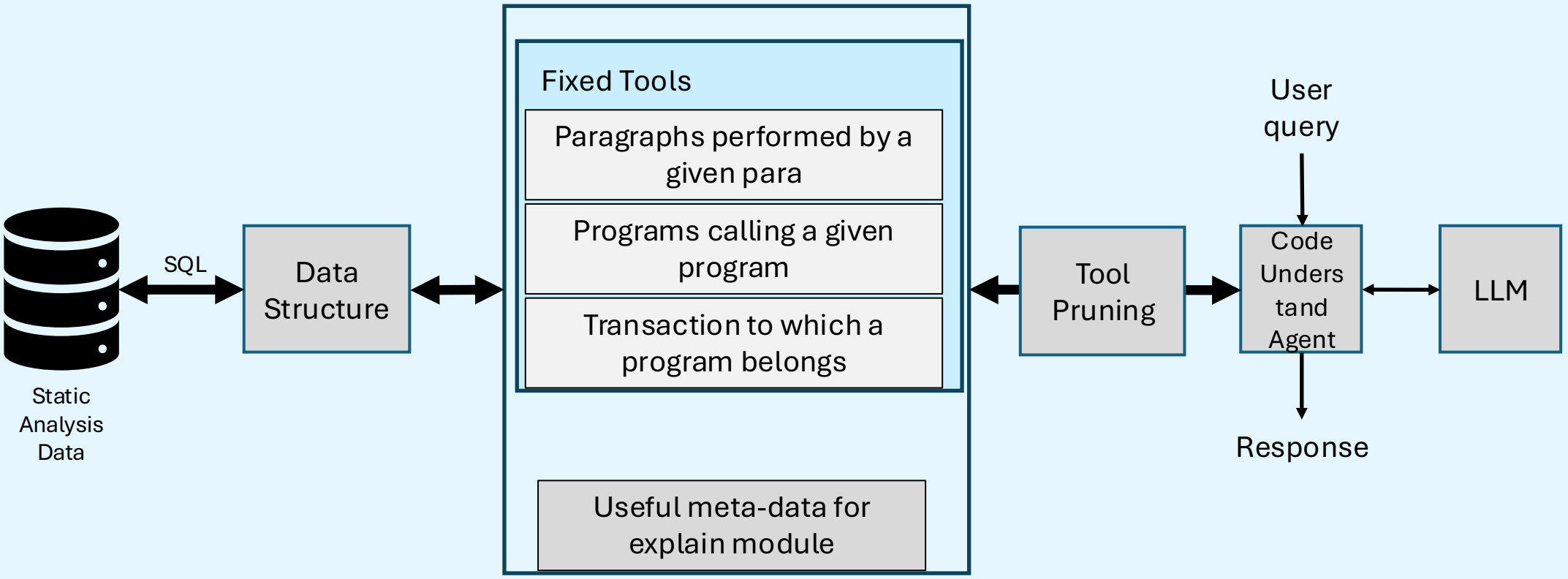


Path Guided Queries for Debugging And Testing

- What if questions
  - What is the value of X at line N is ‘a’
- What value of X will force the program to run line no. N
- How is the value of Y computed at line N
- ….
- ….
- ….

Application Understanding

Code Understanding Agent



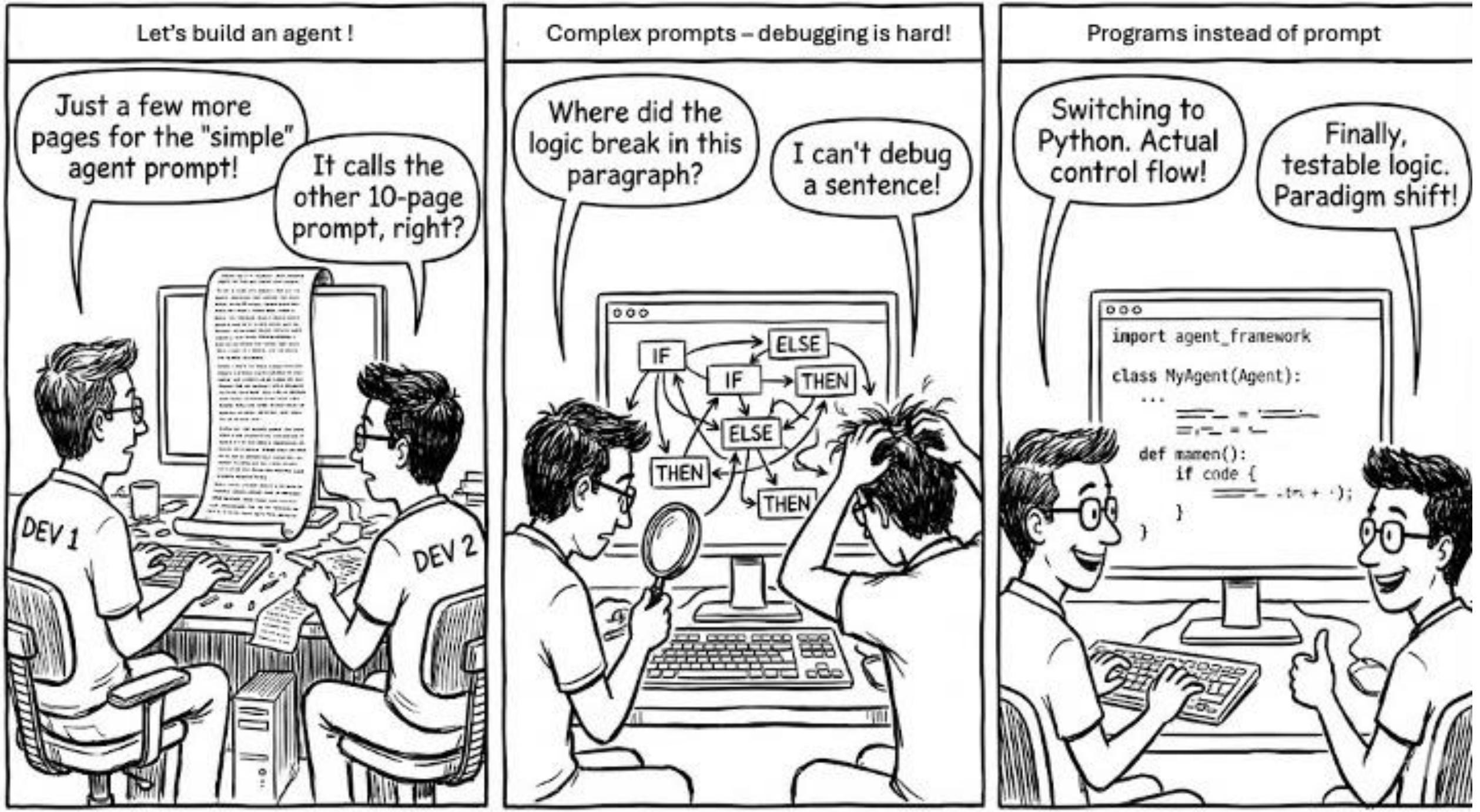
Business Rules

NL business rule generation problem

- Call graphs, entry points, paths, code slices, variables, …
- LLMs are very good at NL understanding and NL generation

- Classical program analysis methods as tools to agents

Software Engineering for GenAI


Let's build an agent !
Just a few more pages for the "simple" agent prompt!
It calls the other 10-page prompt, right?
DEV 1
DEV 2
Complex prompts – debugging is hard!
Where did the logic break in this paragraph?
I can't debug a sentence!
IF
ELSE
IF
THEN
ELSE
THEN
THEN
Programs instead of prompt
Switching to Python. Actual control flow!
Finally, testable logic. Paradigm shift!
import agent_framework
class MyAgent(Agent):
def

Project Mellea

https://github.com/generative-computing/mellea

AssetOpsBench: Benchmarking AI Agents for Task Automation in Industrial Asset Operation and Maintenance

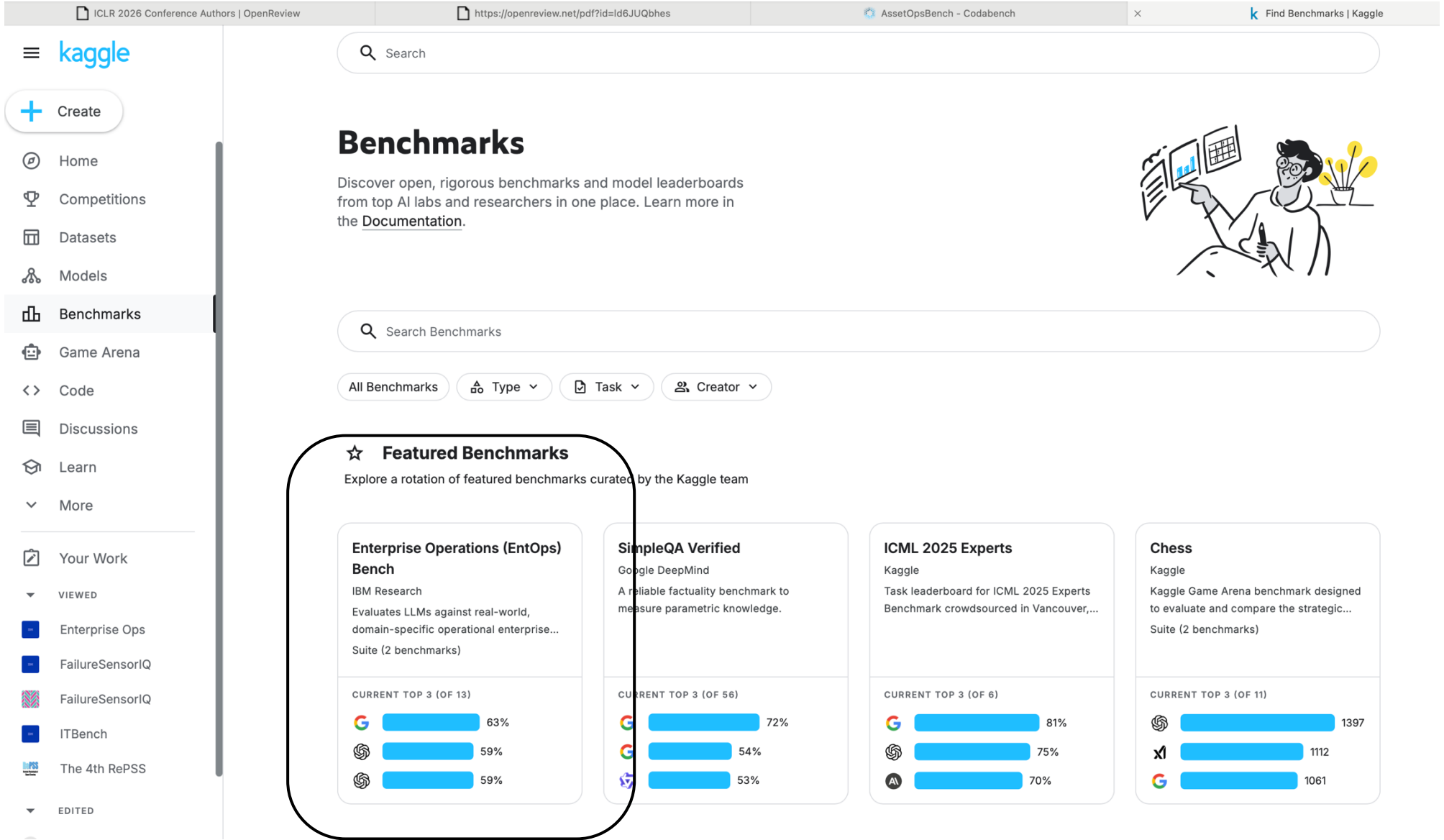
IBM Research

https://github.com/IBM/AssetOpsBench

Rise of Enterprise Benchmark

https://www.kaggle.com/benchmarks

Special Thanks to Kaggle and IBM Research Team



GenAI, AI Agents in Industry – Early Experiences

RD Naik

RD recently retired from TCS Research as Chief Scientist after an illustrious career spanning nearly four decades. He is now a Professor of Practice at COEP Technological University (COEP Tech), formerly the College of Engineering Pune which is also his alma mater. Over the years, RD has worked across a wide spectrum of Software Engineering and related areas.

Suman Roy

Suman is a Consulting Member of Technical Staff in Oracle's Health and AI group in Bangalore, where he provides technical leadership for Healthcare AI initiatives on Oracle OCI. He brings over three decades of industry experience, having worked at Infosys, Optum, Mercedes-Benz, Satyam, GE, and others. He received his BE from Jadavpur University and his ME and PhD from IISc.

Prabhat Shankar

Prabhat is the Industrial AI Solutions Lead at ABB, Bangalore, where he leads the development of Industrial Automation and Predictive Maintenance solutions by building in-house ML models. He holds a BTech and MTech from IIT Kharagpur and a PhD from the RIKEN Center for Developmental Biology and Hiroshima University, Japan.

Discussion