



Powering Innovation That Drives Human Advancement

Harnessing LLMs for Advanced EDA Platforms: Code Generation and Beyond

Akhilesh Kumar

EDPS, 2024

Outline

- Background
- LLMs for Code Generation
- RHSC Copilot
- Agent-Based RHSC Copilot Architecture
- Opportunities, Challenges, Summary

A Few Examples of Recent LLM EDA Work

ChipNeMo: Domain-Adapted LLMs for Chip Design

Mingjie Liu[§], Teodor-Dumitru Ene[§], Robert Kirby[§], Chris Cheng[§], Nathaniel Pinckney[§], Rongjian Liang[§]
 Jonah Alben, Himyanshu Anand, Sanmitra Banerjee, Ismet Bayraktaroglu, Bonita Bhaskaran, Bryan Catanzaro
 Arjun Chaudhuri, Sharon Clay, Bill Dally, Laura Dang, Parikshit Deshpande, Siddhant Dhodhi, Sameer Halepete
 Eric Hill, Jiashang Hu, Sumit Jain, Brucec Khailany, George Kokai, Kishor Kunal, Xiaowei Li
 Charley Lind, Hao Liu, Stuart Oberman, Sujet Omar, Sreedhar Pratty, Jonathan Raiman, Ambar Sarkar
 Zhengjiang Shao, Hanfei Sun, Pratik P Suthar, Varun Tej, Walker Turner, Kaizhe Xu, Haoxing Ren
 NVIDIA

Abstract—ChipNeMo aims to explore the applications of large language models (LLMs) for industrial chip design. Instead of directly deploying off-the-shelf commercial or open-source LLMs, we instead adopt the following domain adaptation techniques: custom tokenizers, domain-adaptive continued pretraining, supervised fine-tuning (SFT) with domain-specific instructions, and domain-adapted prompts. We evaluate the performance of three selected assistant chatbots on a wide range of tasks including design and report generation, and bug triage. In this study, we focus on these three specific LLM applications: an **engineering assistant chatbot** for GPU ASIC and Architecture design engineers, which understands internal HW designs and is capable of explaining complex design topics; **EDA scripts generation** for

ChatEDA: A Large Language Model Powered Autonomous Agent for EDA

Zhuolun He^{1,2†}, Haoyuan Wu^{2†}, Xinyun Zhang¹, Xufeng Yao¹, Su Zheng¹, Haisheng Zheng², Bei Yu¹
¹Chinese University of Hong Kong ²Shanghai Artificial Intelligent Laboratory

Abstract—The integration of a complex set of Electronic Design Automation (EDA) tools to enhance interoperability is a critical concern for circuit designers. Recent advancements in large language models (LLMs) have showcased their exceptional capabilities in natural language processing and comprehension, offering a novel approach to interfacing with EDA tools. This research paper introduces ChatEDA, an autonomous agent for EDA empowered by a large language model, AutoMage, complemented by EDA tools serving as executors. ChatEDA streamlines the design flow from the Register-Transfer Level (RTL) to the Graphic Data System Version II (GDSII) by effectively managing task planning, script generation, and task execution. Through comprehensive experimental evaluations, ChatEDA has demonstrated its proficiency in handling diverse requirements, and our fine-tuned AutoMage model has exhibited superior performance compared to GPT-4 and other similar LLMs.

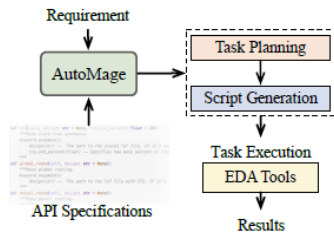


Fig. 1 Overview of AutoMage powered ChatEDA. With AutoMage as the controller and EDA tools as the executors, the workflow consists of three stages: 1) Task planning; 2) Script Generation; 3) Task execution;

OpenROAD-Assistant: An Open-Source Large Language Model for Physical Design Tasks

Utsav Sharma*
 New York University
 USA

Bing-Yue Wu*
 Arizona State University
 USA

Sai Rahul Dhanvi Kankipati
 Arizona State University
 USA

Vidya A. Chhabria
 Arizona State University
 USA

Austin Rovinski
 New York University
 USA

Abstract

Large language models (LLMs) have shown significant potential in serving as domain-specific chatbots. Recently, these models have emerged as powerful tools for chip design, providing both natural language responses and script generation for domain-specific inquiries in assistive design approaches.

design and the use of electronic design automation (EDA) tools. The necessary tool proficiency not only creates large barriers to entry for newcomers to the field, but also creates productivity challenges for experienced chip designers who transition to a new EDA tool or to an unfamiliar stage of the physical design flow. These challenges

RTLCoder: Outperforming GPT-3.5 in Design RTL Generation with Our Open-Source Dataset and Lightweight Solution

Shang Liu
 HKUST, sliudx@connect.ust.hk

Wenji Fang
 HKUST, wenjifang1@ust.hk

Yao Lu
 HKUST, yludf@connect.ust.hk

Qijun Zhang

Hongce Zhang

Zhiyao Xie*

HKUST, qzhangcs@connect.ust.hk

HKUST (GZ) & HKUST, hongcezh@ust.hk

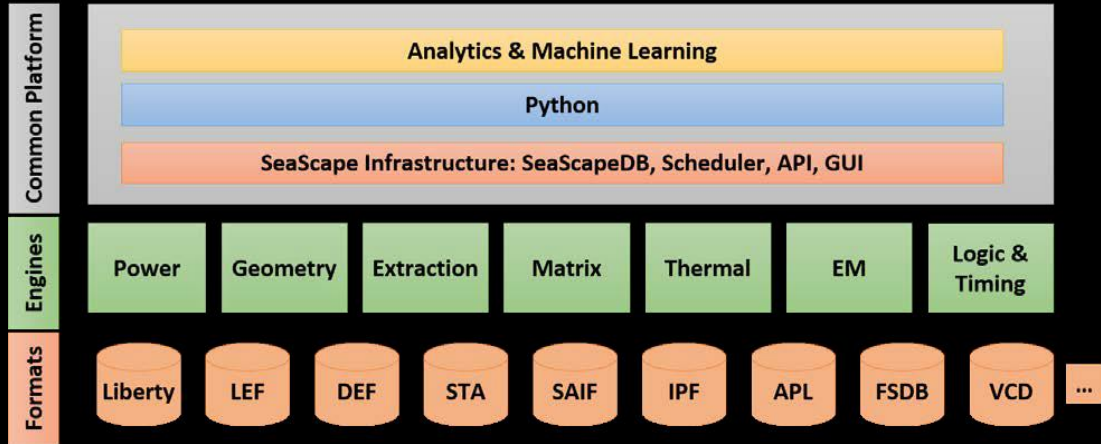
HKUST, eezhiyao@ust.hk

Abstract—The automatic generation of RTL code (e.g., Verilog) using natural language instructions and large language models (LLMs) has attracted significant research interest recently. However, most existing approaches heavily rely on commercial LLMs such as ChatGPT, while open-source LLMs tailored for this specific design generation task exhibit notably inferior performance. The absence of high-quality open-source solutions restricts the flexibility and data privacy of this emerging technique. In this study, we present a new customized LLM solution with a modest parameter count of only 7B, achieving better performance than GPT-3.5 on all representative benchmarks for RTL code generation. Especially, it outperforms GPT-4 in VerilogEval Machine benchmark. This remarkable balance between accuracy and efficiency is made possible by leveraging our new RTL code dataset and a customized LLM algorithm, both of which have been made fully open-source¹.

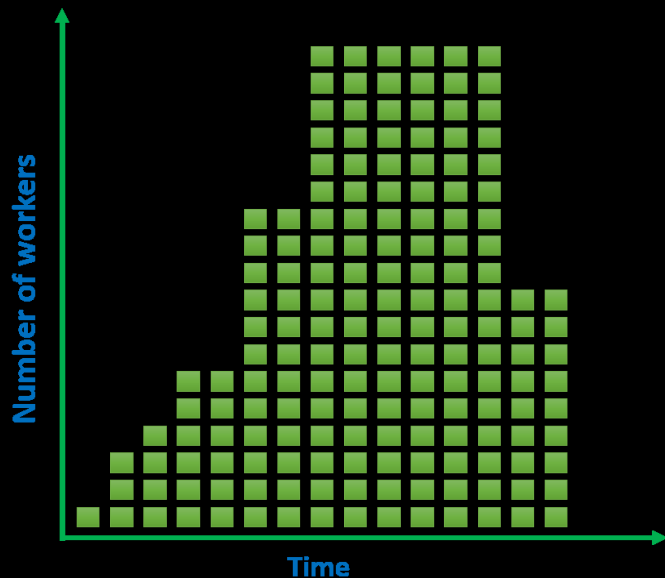
Works	New Training Dataset	New LLM Model	Outperform GPT-3.5
Prompt Engineering [9]–[11], [14], [15]	N/A	N/A	N/A
Thakur et al. [13]	Open-Source	Open-Source	No
VerilogEval [12]	Closed-Source	Closed-Source	Comparable
ChipNeMo [3]			
ChipGPT-FT [17]			
BetterV [18]			Yes
RTLCoder	Open-Source	Open-Source	Yes

TABLE I: LLM-based works on automatic design RTL (e.g., Verilog) generation based on natural language instructions.

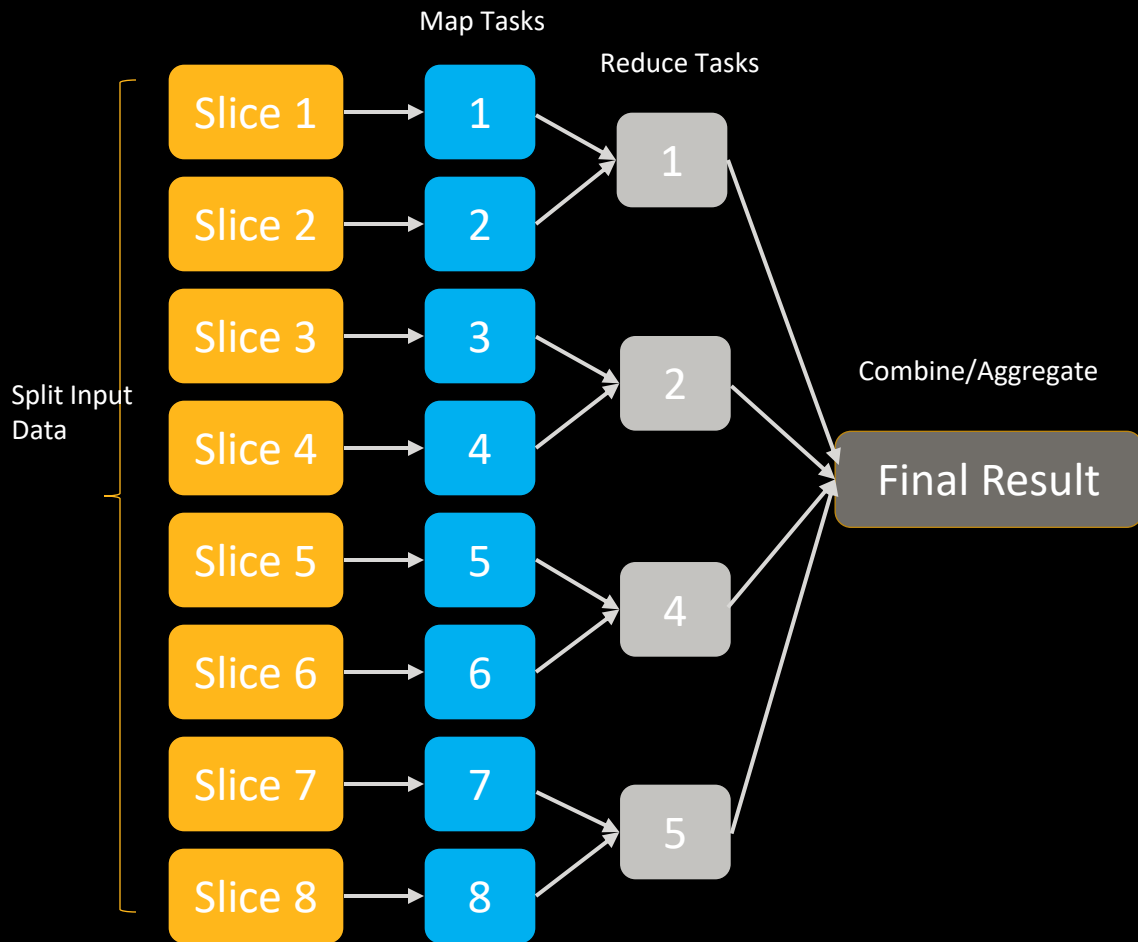
SeaScape Platform



- Power Noise and Reliability analyses framework for SoC/3DIC
- Main components of SeaScape Infra are Database, Scheduler, Python Interface and GUI
- Elastic Compute and big data analytics
- ML Support

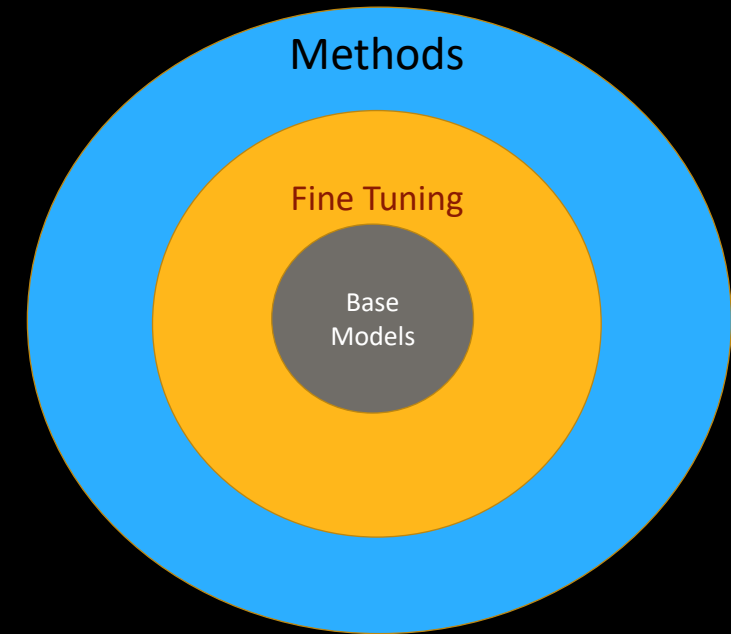
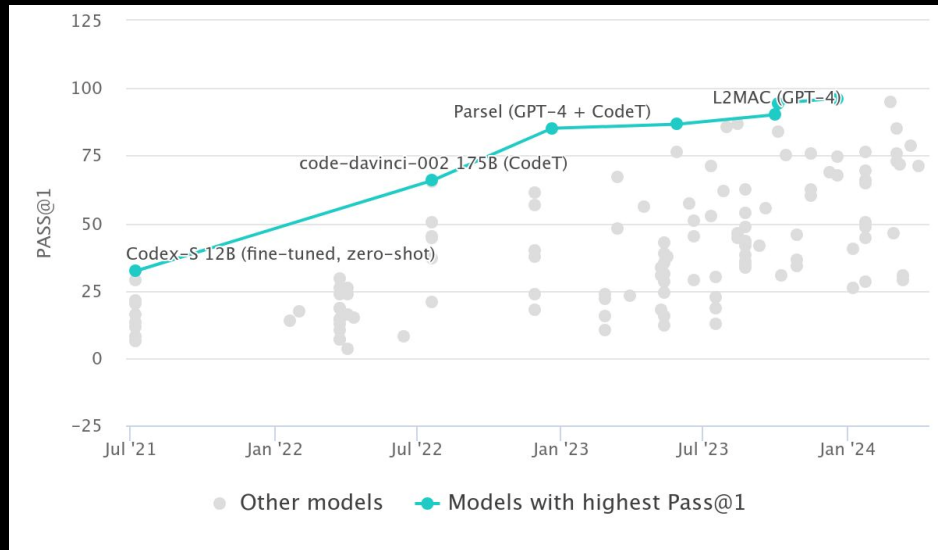


MapReduce



- The input data is split into smaller chunks
- On each of the smaller data chunks an operation *Map* is applied. This is parallelized.
- The above operation provides intermediate results.
- The intermediate results are combined and is called as *Reduce* operation.
- A final result is generated as the last step.

LLM-Based Code Generation: Benchmarking and Methods

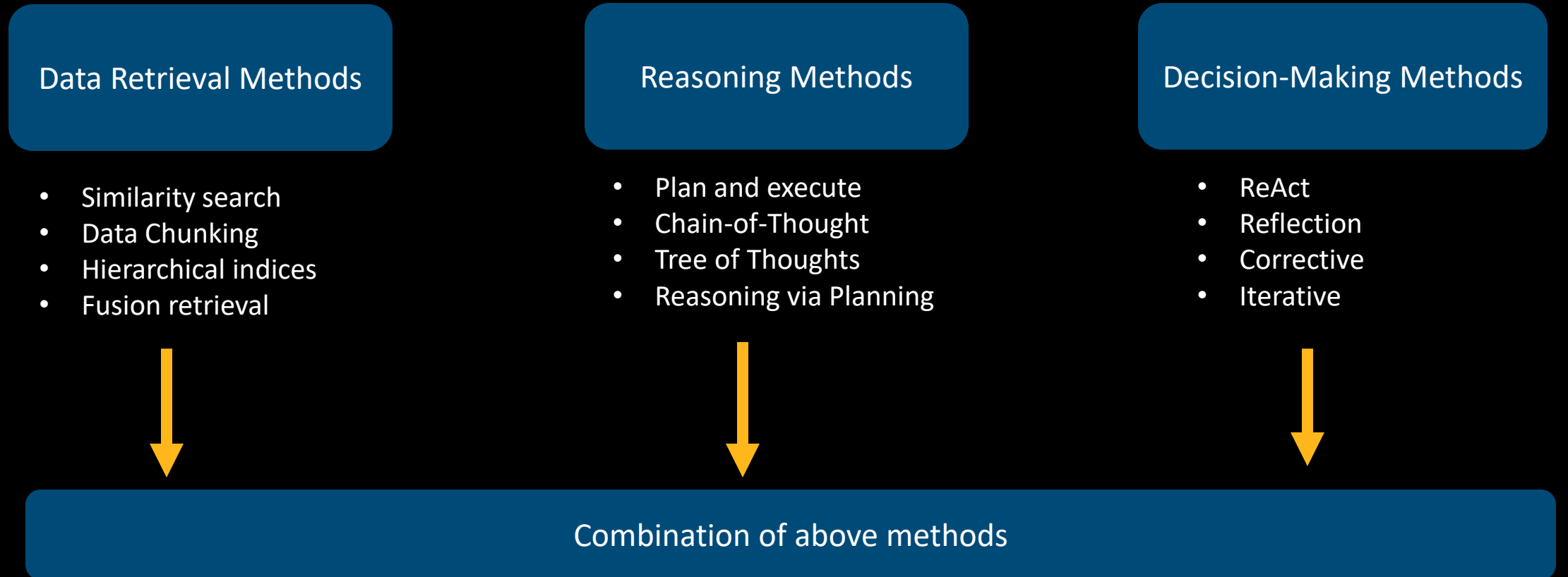


Benchmarking

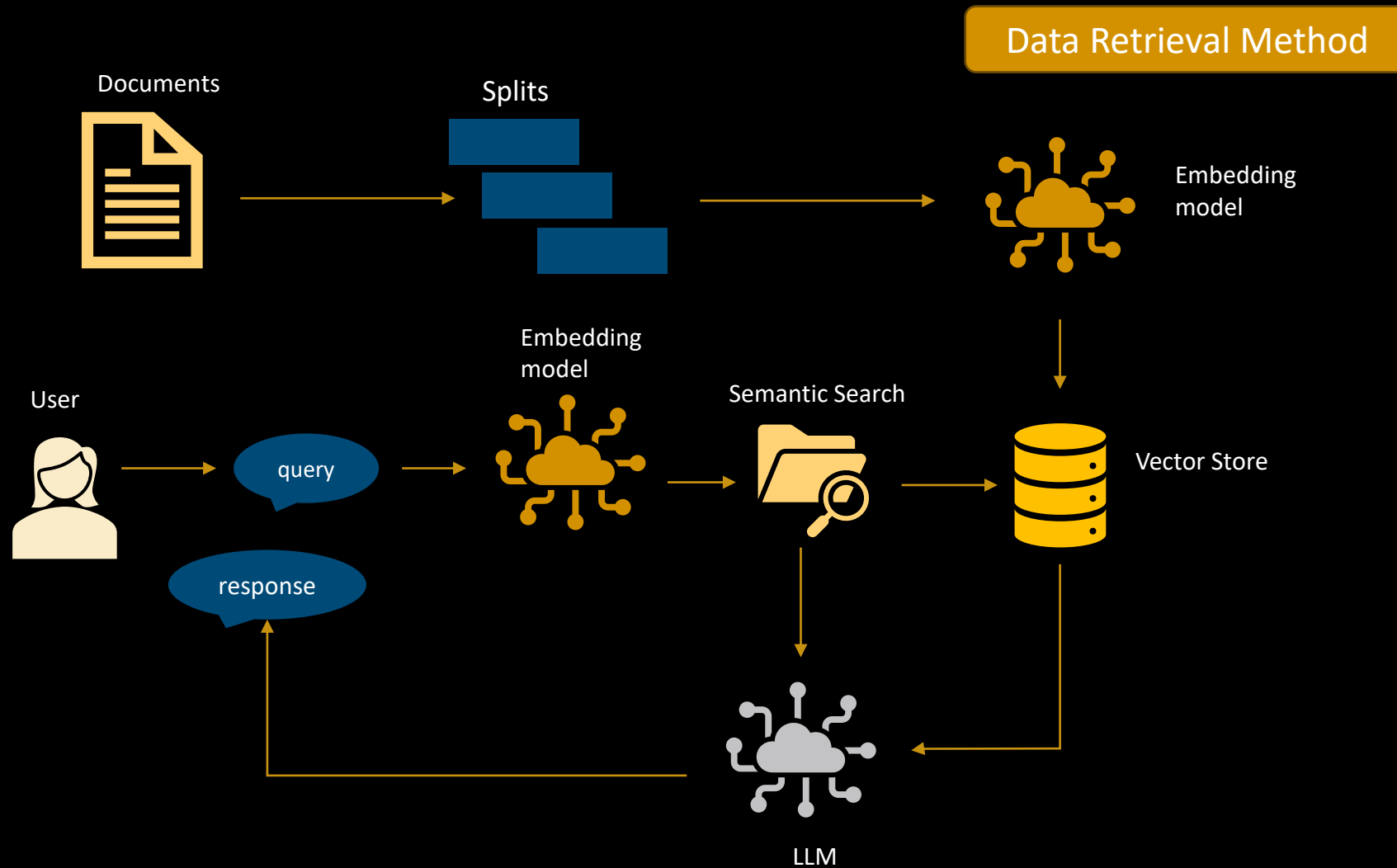
- ❑ HumanEval is the most widely recognized benchmark for code generation (164 handwritten programming problems with unit tests)
- ❑ MultiPL-E is HumanEval with 18 programming languages
- ❑ HumanEval-X is 820 high-quality human written code problems
- ❑ Proprietary benchmarks
 - ❑ Natural2Code (Google)
 - ❑ TestGen-LLM (Meta)

- ❑ Base Models such as GPT-4, Code Llama, Mistral directly used for code generation
- ❑ Fine-tuning can be performed for domain specific application
- ❑ Several methodologies can be applied to augment code generation capabilities of LLMs

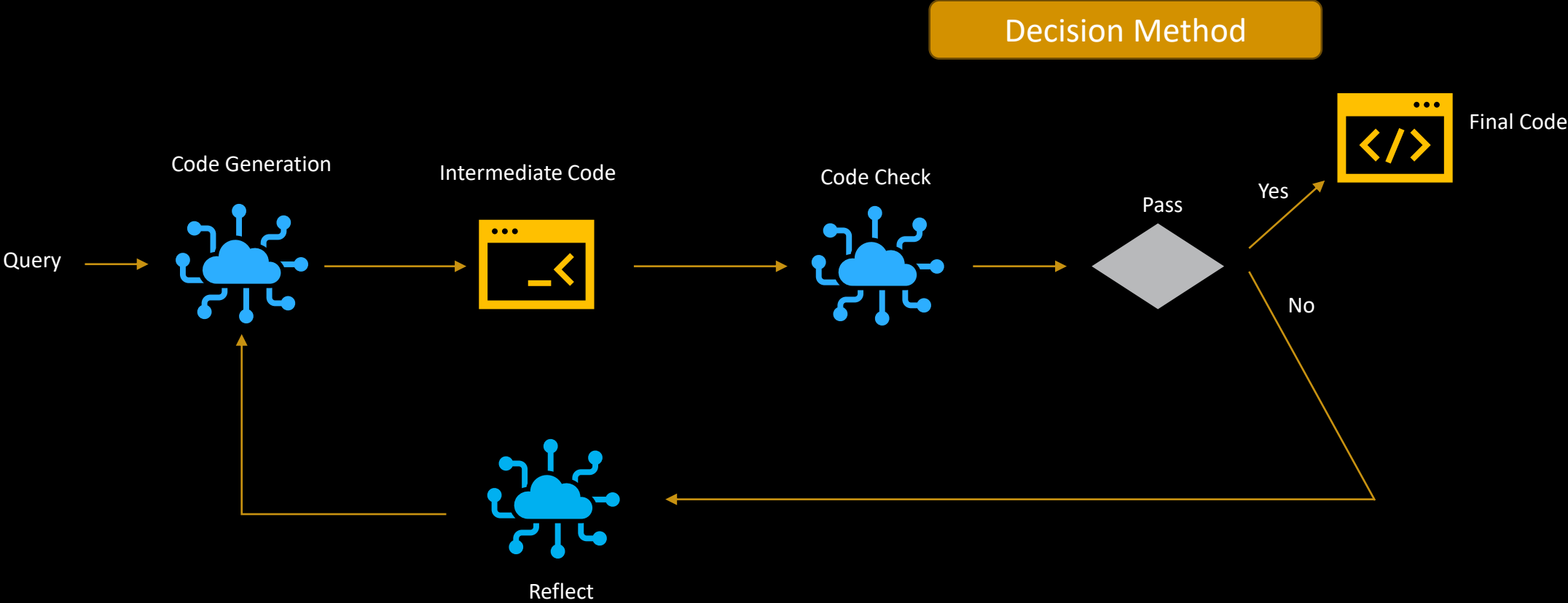
Methods for LLM-Based Code Generation



RAG Framework

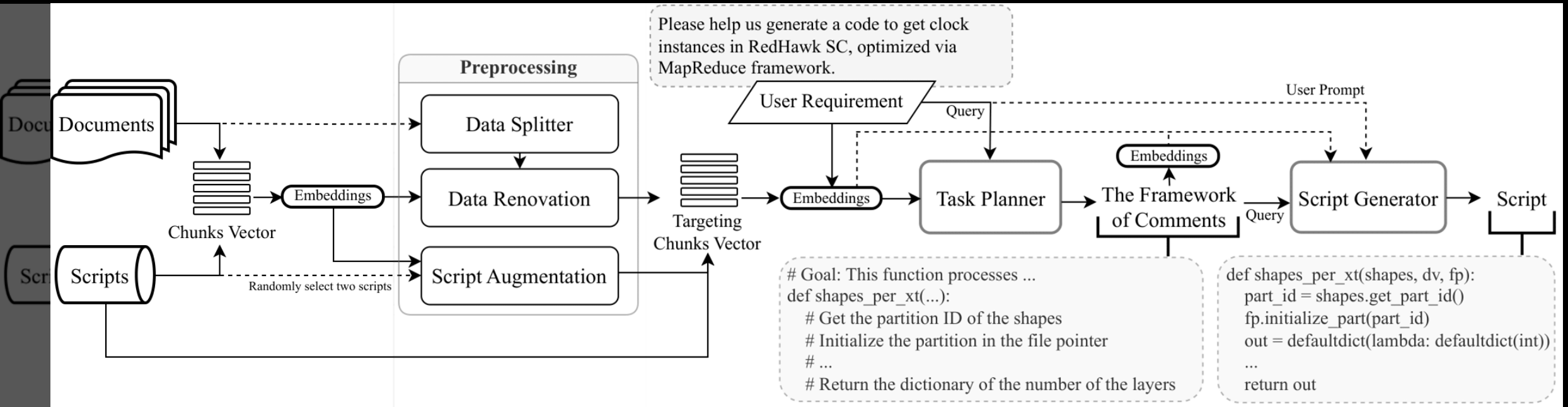


Self-Reflection for Code Generation



RedHawk-SC Code Generation

Flowchart - Proposed Methods



Data:

- RedHawk-SC Documents, Scripts
- MapReduce Documents

Y. Lin, A. Kumar, N. Chang, W. Zhang, M. Zakir, R. Apte, H. He, C. Wang, J. Roger Jang, ISLAD 2024, Novel Preprocessing Technique for Data Embedding in Engineering Code Generation Using Large Language Model.

Evaluation – Ansys RedHawk-SC Code Generation Arena

182 votes from 28 experts

STEP 1: Please fill your name

Owen Lin

Submit

Randomly Choose One of Our 20 User Requirements (Test Cases)

arena

leaderboard

leaderboard-testcase

User Requirement

```
1  ## Task: Given two SwitchingActivityView objects, develop a script to get the worst NUM instances with largest total toggles_
2  ## 1. the first SwitchingActivityView object
3  ## 2. the second SwitchingActivityView object
4  ## 3. number of instances to be returned, which is by default 10.
5
6  ## The return value should be a list of NUM dicts. For each dict, it has the following keys/values:
7  ## - 'delta' : the total toggles_per_second difference in two SwitchingActivityView objects
8  ## - 'instance' : the instance with the largest diff
9  ## - 'cell' : the cell name of the instance
10
11
12  ## To implement it using map-reduce
13  ## 1. A map function could be called to get the worst NUM instances with largest total toggles_per_second in the partition
14  ## 2. reduce_sorted could be used to reduce the results from different partitions
```

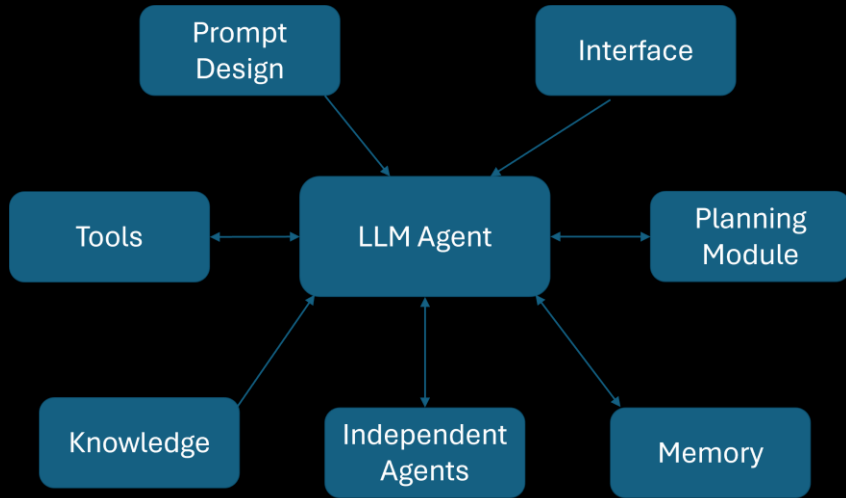
Evaluation – Ansys RedHawk-SC Code Generation Arena

Rank	Approach	Arena Elo	95% CI (testcase)	Votes	Unique ID
1	RAG_IKEC_CoT_Splitter_similarity_top_k_6_ChatEDA	1054	57.0 / -59.0	35	19
2	RAG_Augmentation_Splitter_Renovation_similarity_top_k_6_ChatEDA	1045	59.0 / -62.0	28	12
3	RAG_Augmentation_Splitter_similarity_top_k_6_ChatEDA	1013	59.0 / -61.0	24	9
4	RAG_IKEC_Splitter_similarity_top_k_6_ChatEDA	1010	60.0 / -60.0	31	20
5	RAG_similarity_top_k_6_ChatEDA	994	64.0 / -57.0	26	5
6	RAG_IKEC_Augmentation_Splitter_Renovation_similarity_top_k_6_ChatEDA	980	59.0 / -57.0	19	14
7	RAG_IKEC_Augmentation_Splitter_similarity_top_k_6_ChatEDA	955	61.0 / -57.0	23	8
8	RAG_IKEC_similarity_top_k_6_ChatEDA	947	59.0 / -56.0	23	11

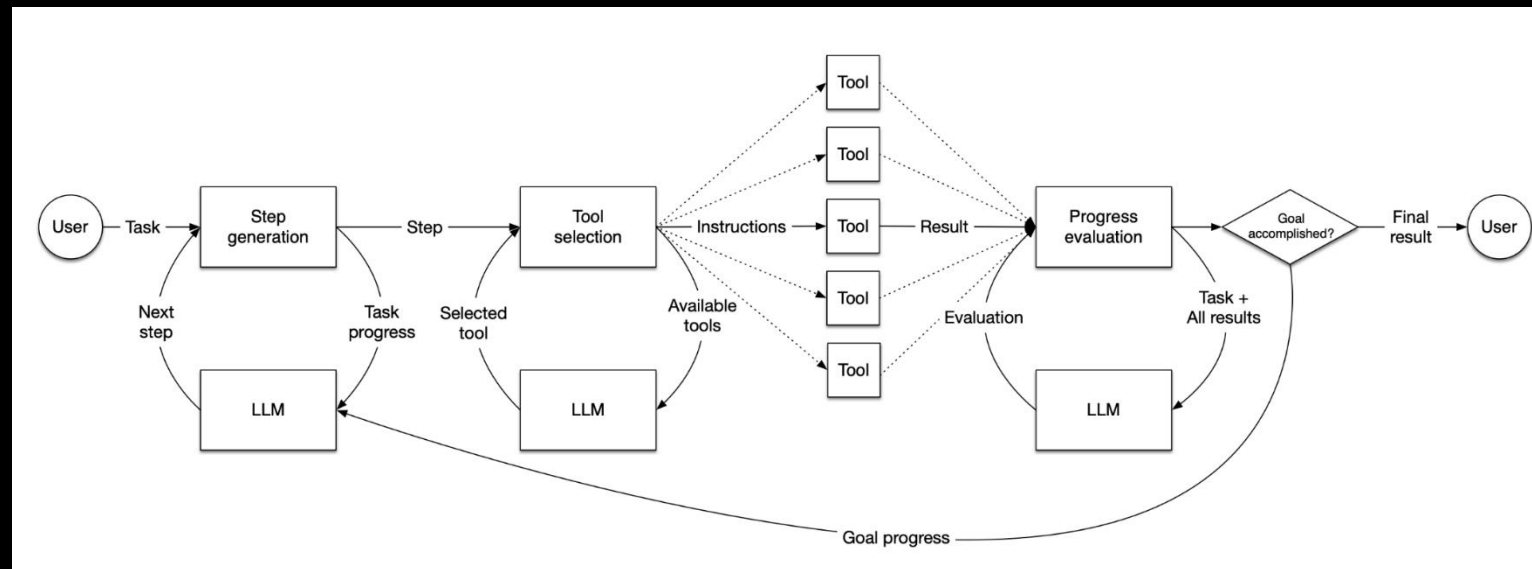
Total #approaches: 20 / #votes: 182 / #users: 28
(Update: 2024-03-26 23:05:16.370746, GMT+8)

EDA Platform Copilot Vision RHSC As An Example

LLM Agents Architecture



- Provide response
- Perform specific tasks
- Work collaboratively
- Use tools



Source:
<https://www.llamaindex.ai/>

RHSC Copilot Goals

- Enhancing productivity and efficiency
 - Examples: answering a query on tool usage, commands, general guidelines
- Automating tasks on SC
 - Examples: Automatically performing a high-level task described in natural language by breaking it down into smaller sub-tasks and invoking the relevant APIs, MapReduce code generation.
- Intuitive Assistance
 - Examples: Suggesting a correction in the script, suggesting next steps, providing a feedback on the results, filling in the skills gaps
- Optimization Tasks
 - Optimization at various levels

Operational Modalities

- Chat – Answer questions about the tool, product etc.
- Assistance: Generate Text-to-Code, help with completing tasks efficiently, fill in the skills-gaps
- Automation (agents-based): Planning and reason based and capability to react to the environment, decide which APIs to use, perform tasks in multiple steps.
- Debugging and Analysis

RHSC Copilot Components

User Interface

Chat

Assistance

Automation

Debugging

Task Orchestration with Planning and Reasoning (Agent Based)

Copilot Engine

Models

Embedded RHSC Knowledge base

Tool Plugins

Workflows

Guardrails

RHSC Application

RHSC APIs

RHSC Data

Components Description

User Interface

- Enables the user to interact with the RHSC copilot
- Interfaces embedded at multiple places across the platform in an intuitive fashion

Task Orchestration

- Planning and reasoning agents
- Determines how to respond to a query or perform tasks
- Can react to the environment feedbacks
- Can use multiple models and tools to perform a task

Knowledge Sources and Tool Plugins

- Embedded and indexed RHSC documents and wiki
- Example documented and commented code scripts
- RHSC API plugins
- Plugins for accessing RHSC simulation data

RHSC APIs and Data

- APIs provided by RHSC which can be used in the current context
- Data generated by the current simulation and input data provided by the user, including design data

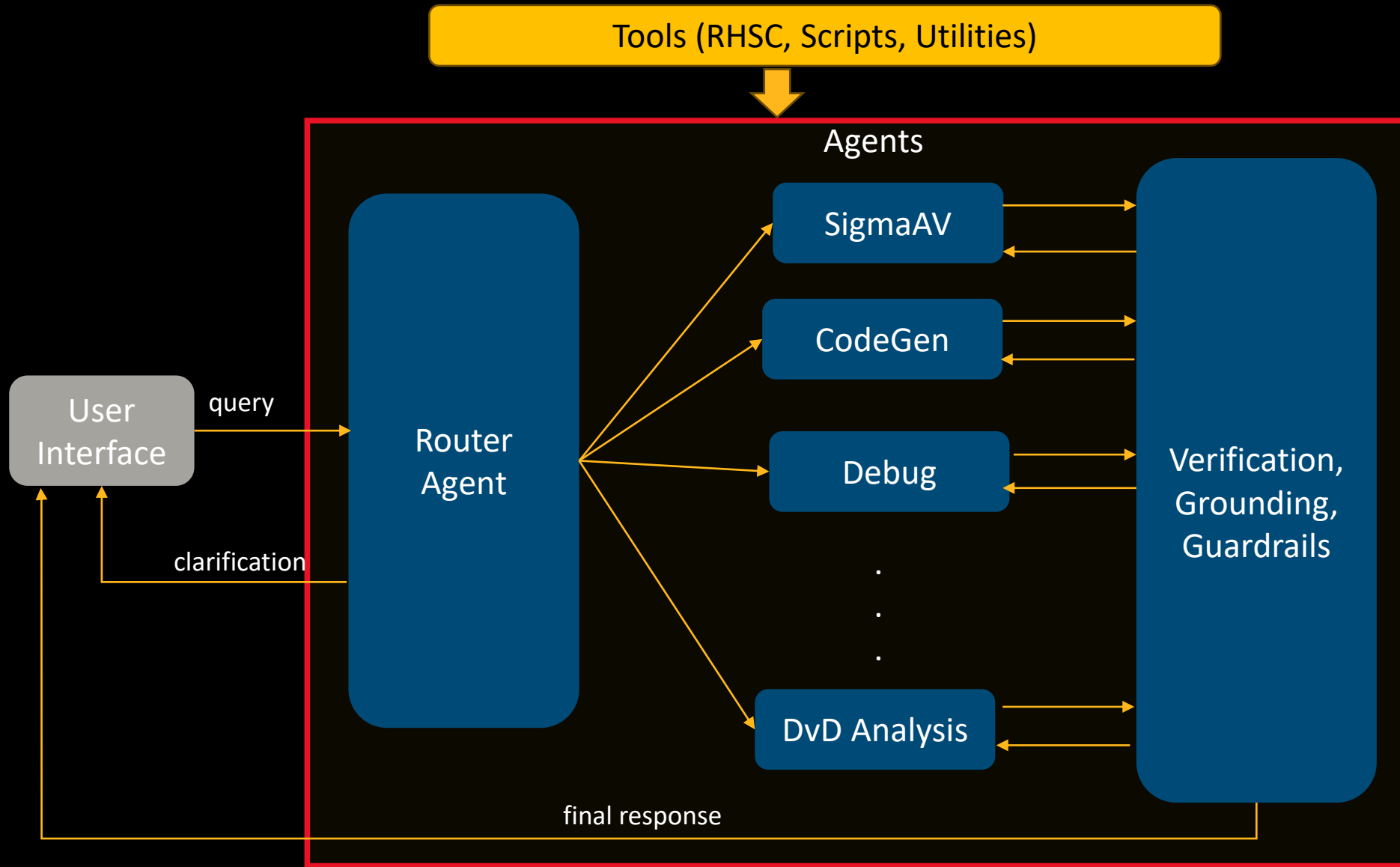
Models

- Refers to the set of LLMs
- May be composed of multiple LLMs each suited for specific tasks
- The models will act agents and work cooperatively in a multi-agent system

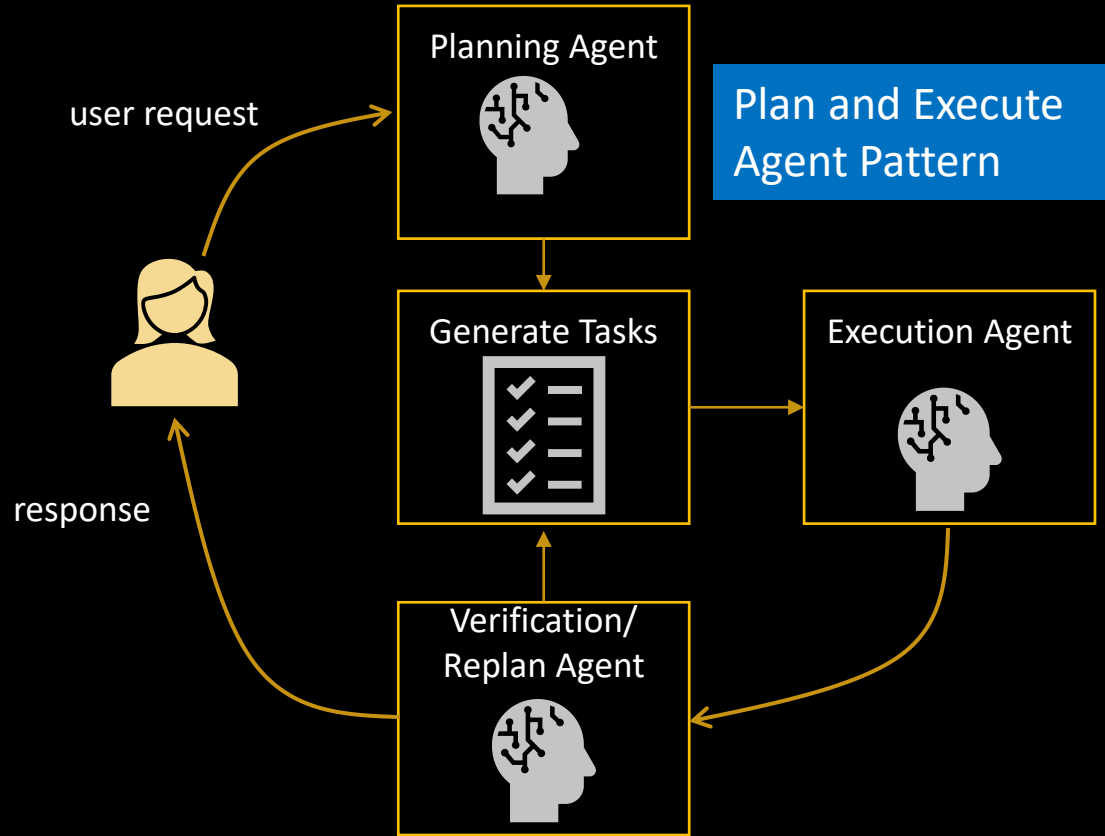
Workflows and Guardrails

- Workflow recipes such as early grid analysis, EM/IR, Ramp Up analysis, sigmaAV, CPM, PFSC
- Generates a guided step by step instruction
- Guardrails for grounding the responses to the RHSC application

Example Agent-Based RHSC Copilot Architecture

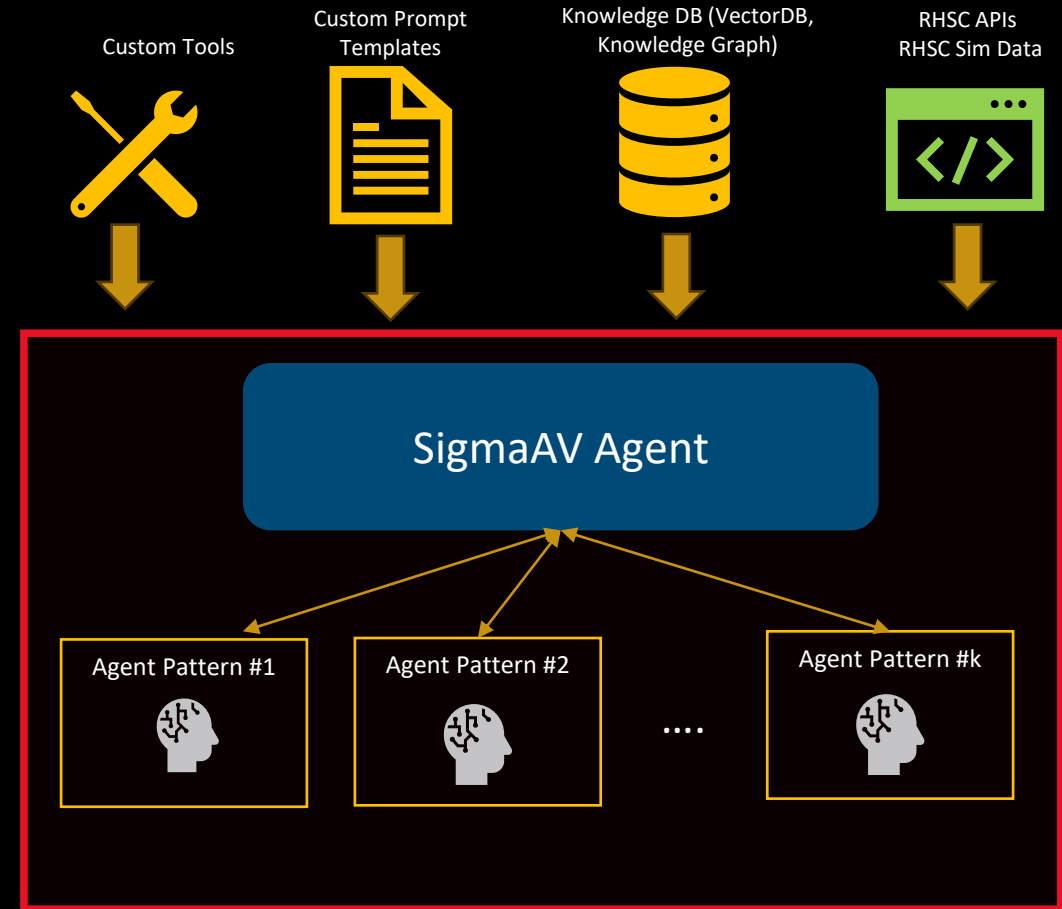


Building Individual Agents with Agent Patterns



Agent Patterns

- Collaboration
- Reflection
- ...other patterns



Sample Response – Debug Assistant

```
FATAL<DIS.999> No communication socket to file service at 10.50.68.172 from workpony121 [Socket error: retry_count 30 for /volume/xt-wk30/alokpandey/bump_def/xt_macsec_ptoc_wrapper/redhawk_sc/static_bump/db_common/dv0::settings(Cannot connect main socket (Connection refused))].
```

QUESTION: Can you provide the logs for all workers that have tried to access the file 'dv0::settings' or any file on the host with IP 10.50.68.172 during the time frame of 10:45 to 10:50?

THOUGHT: This will help us determine if the issue is specific to Worker7 or if it's a broader issue affecting other workers as well. If other workers are also unable to access files on the same host, it could indicate a problem with the host or its file service. If other workers are able to access the file, it could indicate a problem specific to Worker7.

QUESTION: Can you provide the logs for all workers on the host with IP 10.50.68.172 (workpony170) during the time frame of 10:45 to 10:50?

THOUGHT: This will help us determine if the issue is specific to the file service on host workpony170. If workers on the same host are also unable to access files, it could indicate a problem with the file service on that host. If workers on the same host are able to access files, it could indicate a problem with the network connection between hosts.

```
{  "start_time": "10:44:00",
  "end_time": "10:51:00",
  "keywords": ["dv0::settings", "10.50.68.172"],
  "workers": [] }

{  "start_time": "10:44:00",
  "end_time": "10:51:00",
  "hosts": ["10.50.68.172", "workpony170"] }
```

FINAL_ANSWER:

Based on the logs, the issue seems to be with the network connection between the hosts workpony121, workpony171 and the host host170 (IP 10.50.68.172). Multiple workers (Worker5, Worker6, Worker7, Worker8, Worker13, Worker14, Worker15) from the hosts workpony121 and workpony171 are unable to connect to the file service at IP 10.50.68.172 to access the file 'dv0::settings'. They are all receiving the error "Cannot connect main socket (Connection refused)". However, workers on the host workpony170 itself are not reporting any issues, suggesting that the file service on this host is functioning correctly. Therefore, the issue is likely due to a network problem between these hosts.

Opportunities and Challenges for EDA Copilots

Opportunities

- Intuitive natural language interface for the user
- Automation of commonly used workflows
- Debugging assistance in EDA tools
- Assist in results analyses and interpretation
- Providing suggestions on next steps in the EDA tool
- Enable and assist optimization workflows
- Provide guardrails for the users against mistakes
- Can be highly customized for specific EDA tasks
 - Specific *personas*

Challenges

- Compute costs associated with LLMs usage
 - Fine-tuning and usage
- Lack of high-quality and volume of training/context data in EDA domain
- Hallucinations
- Lack of specialized semiconductor/EDA domain knowledge in off-the-shelf models
- Evaluation metrics for generated responses
- Deployment challenges, cloud vs on-prem
 - Data privacy issues
- Trust in LLMs
 - Accuracy, robustness, privacy, safety, ethics

Summary

- Agentic LLM frameworks can simplify complex tasks on EDA platforms
 - Improved productivity and user experience
 - Help with debug and results analyses
 - Reduce errors in setup
- Smaller LLMs can be more easily fine-tuned for specific roles enabling
- Emerging multimodal models and frameworks can be effectively used for EDA domain applications
- Several challenges need to be addressed carefully while developing LLM based EDA solutions

Acknowledgements

- Thanks to Owen Lin, Norman Chang, Wenliang Zhang, Haiyang He, Muhammad Zakir, Rucha Apte, Mallik Vusirikala, Shlok Zanwar, Subrata Nandy for the innovative contributions and valuable discussions on this work!

The image features the Ansys logo on the left, which consists of a yellow slanted bar followed by the word "Ansys" in white. On the right, there is a large, stylized letter 'A' composed of a yellow slanted bar and a white slanted bar. The background is black.

Ansys