Mila

18-03-2026

Coding Large Language Model (Code LLM) Evaluation

Houssem Ben Braiek, Ph. D.

Institution and Team

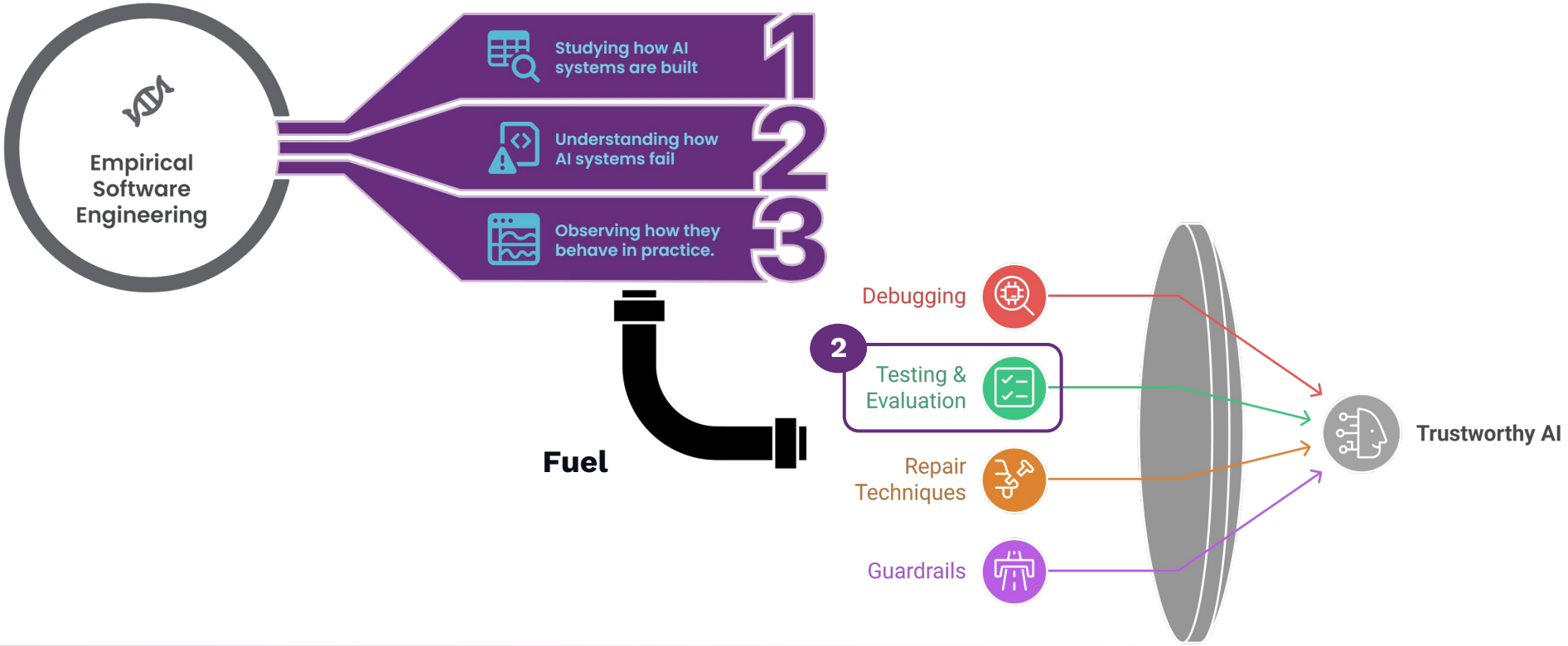


Foutse Khomh, SWAT Director, VP Research and **Innovation,** at

POLYTECHNIQUE MONTREAL



Trustworthy AI Research



XPJudge

Explanation-Aware Pairwise Judging for Cost-Effective LLM Selection

1st Ranked LLM ≠ What you need

XPJudge can be leveraged to *select the most suitable LLM for a given need*, instead of always choosing the highest-ranked model on static benchmarks.

XPJudge relies on a task-oriented evaluation setting built around triplets:

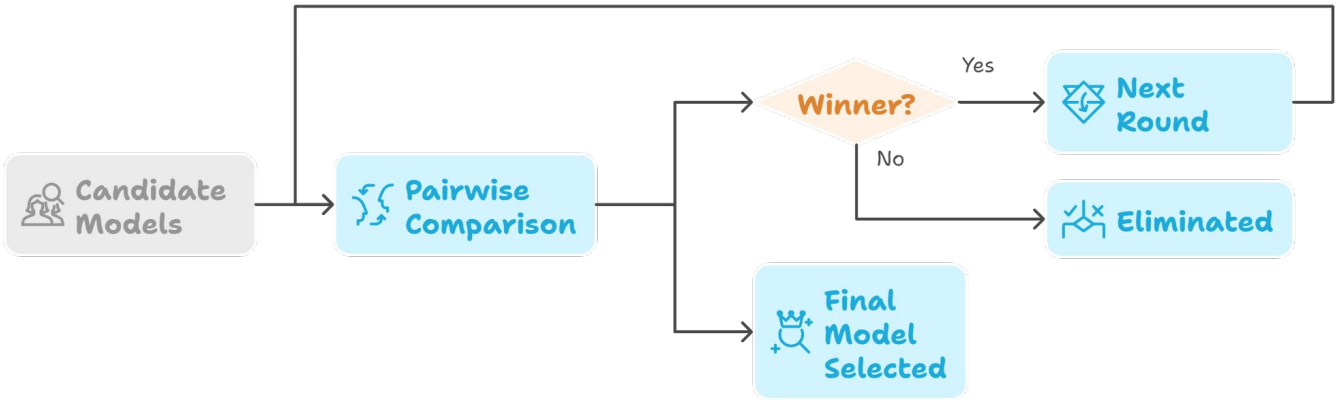
- **Task** → represents the requirement to satisfy
- **Solution** → generated by the LLM under evaluation
- **Explanation** → justification produced by the model

Compare pairs to Select, then, Rank

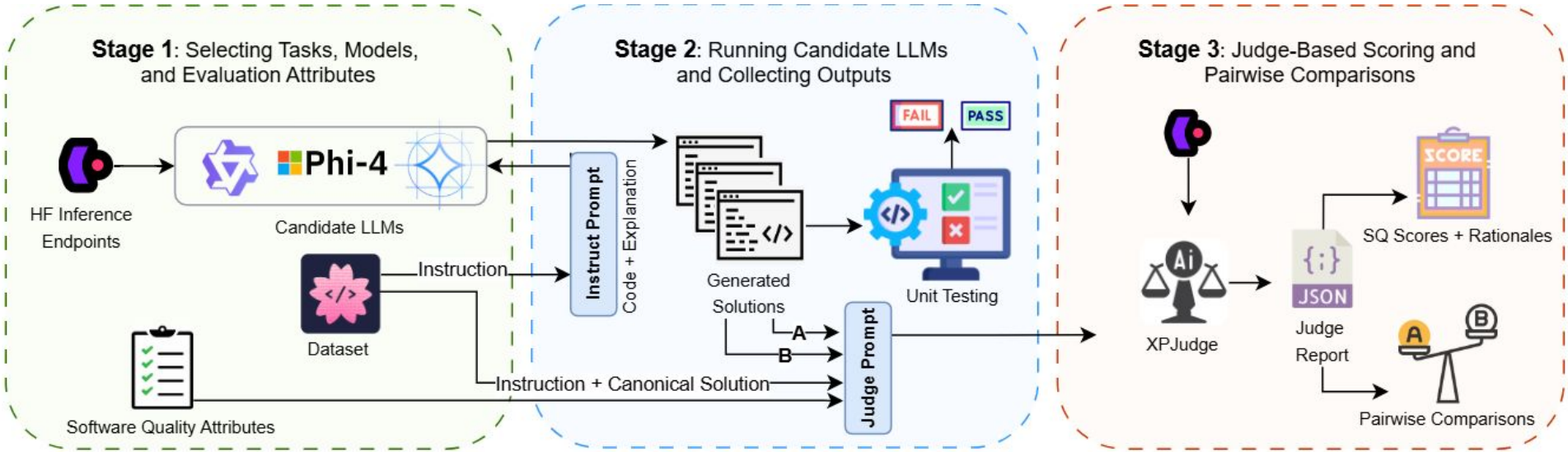
Pointwise judge-based ranking can lead to **overestimated evaluations.**

➔ Absolute scoring is often unreliable with LLMs due to **confirmation bias**, where the judge model tends to agree with the provided answer instead of critically evaluating it.

To address this, **XPJudge** uses an eXplainable Pairwise evaluation strategy.



Three Stages Workflow



- ➔ **Less sensitive** to judge bias than **pointwise** ranking
- ➔ Requires **less computation** than **full pairwise** ranking

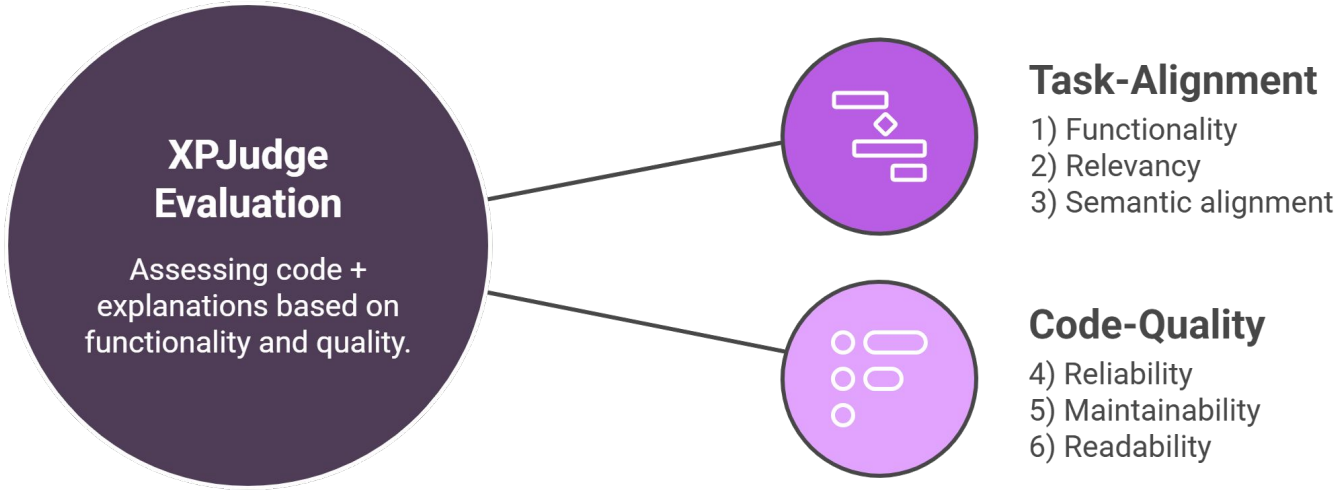
Software Engineering is our Domain

We used the **SWEBench** hard subset, which contains challenging real-world software engineering tasks.



SWE-bench

For the evaluation, we defined **task-specific judging criteria**:



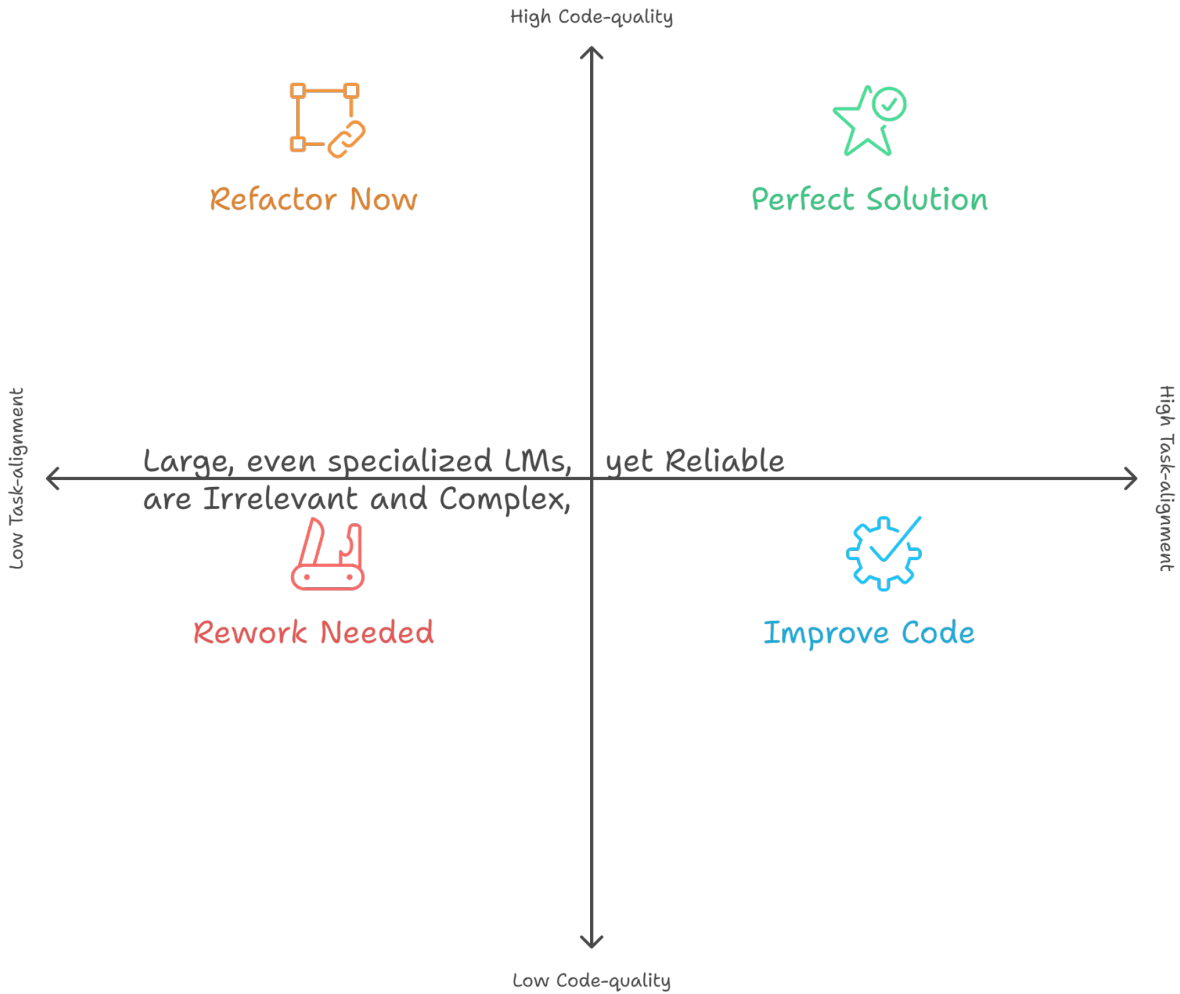
"The bigger the better" is not always True in LLMs

The highest-ranked models are often the largest ones, reinforcing **anchoring bias toward model size**.

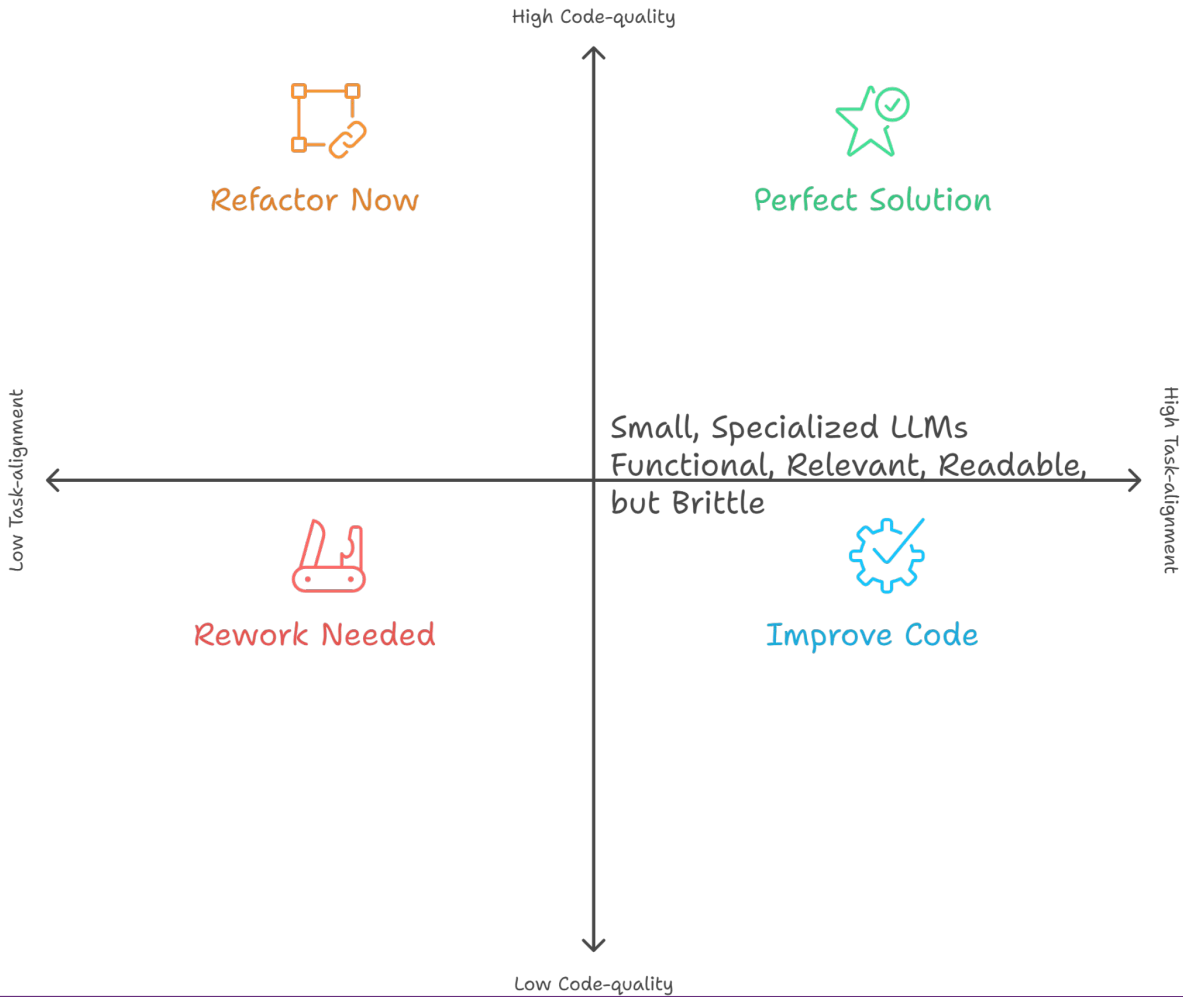
	Model	Pass@1	Params (B)
3	Qwen2.5-Coder-32B-Instruct	27.7	32
	Gemma-3-27B-Instruct	25.0	27
1	Phi-4-14B	24.3	14
2	Qwen2.5-Coder-14B-Instruct	23.6	14
	Qwen2.5-14B-Instruct	20.9	14
4	Qwen2.5-Coder-7B-Instruct	20.3	7

- ➔ With XPJudge, the winning models were not the usual suspects.
- ➔ Smaller, more specialized language models often produced **cleaner, well-aligned, and fully functional solutions.**

Insights from our XPJudge evaluations



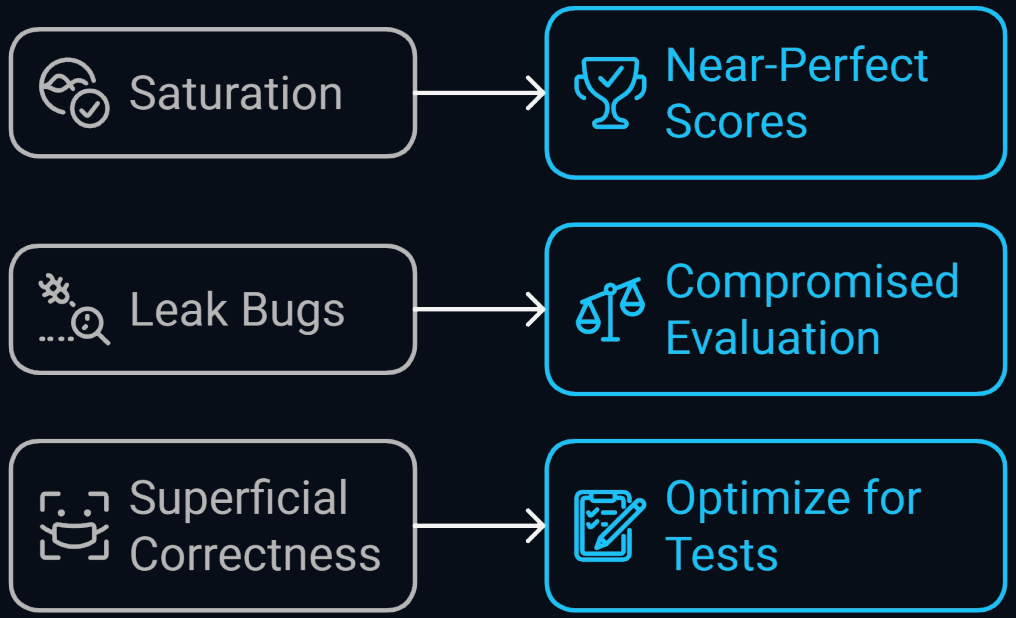
Insights from our XPJudge evaluations



PrismBench

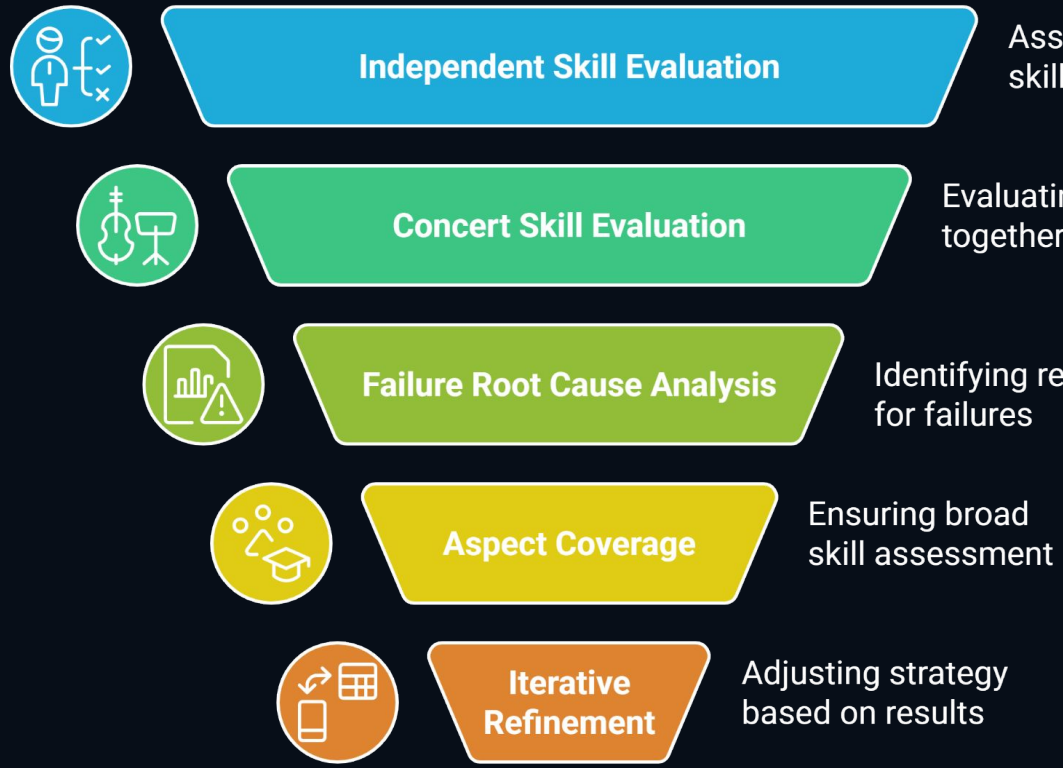
A dynamic benchmarking framework for LLMs using Monte Carlo tree search and multi-agent coordination

Why Static Benchmarks Fall Short

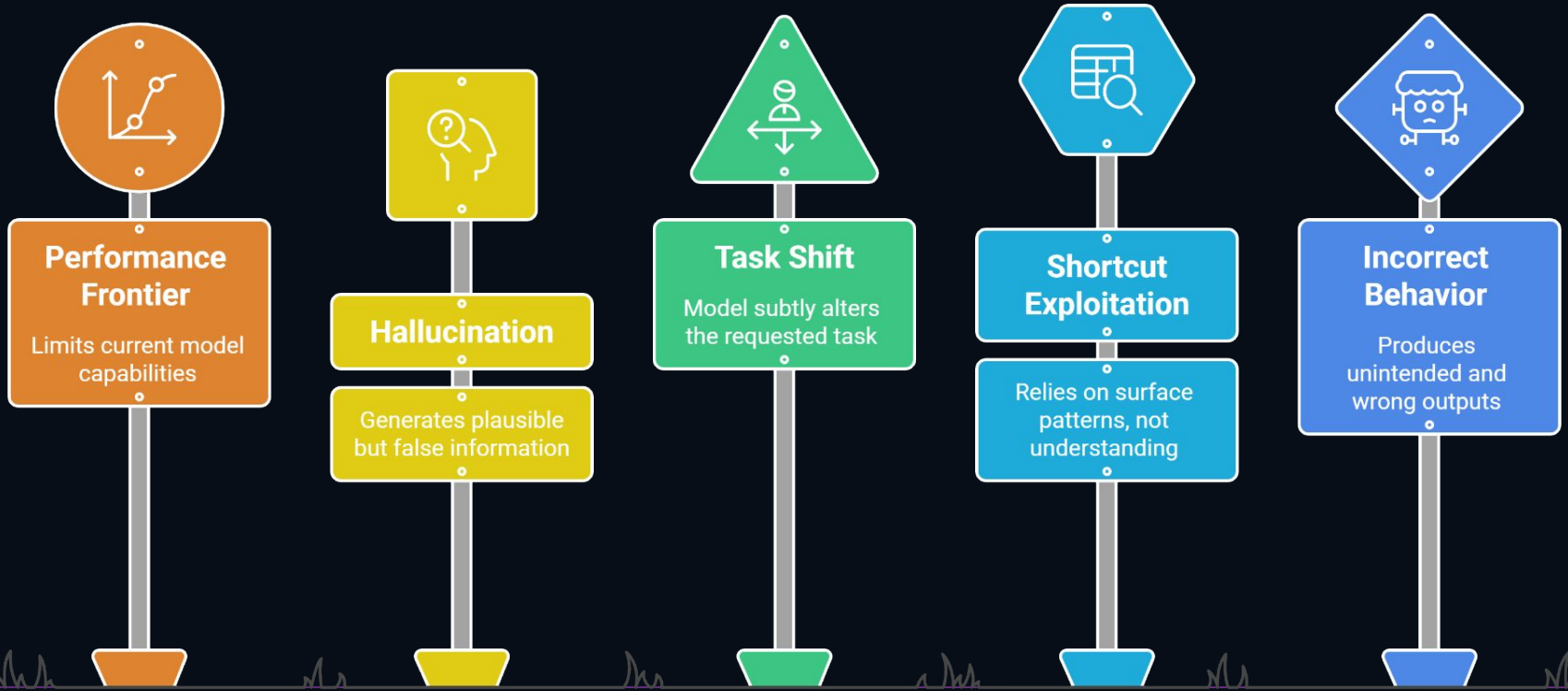


➔ In Short, Static scores (like pass@1) mask brittle model's behavior.

Proposed Solution: PrismBench



The Failure Modes we Hunt

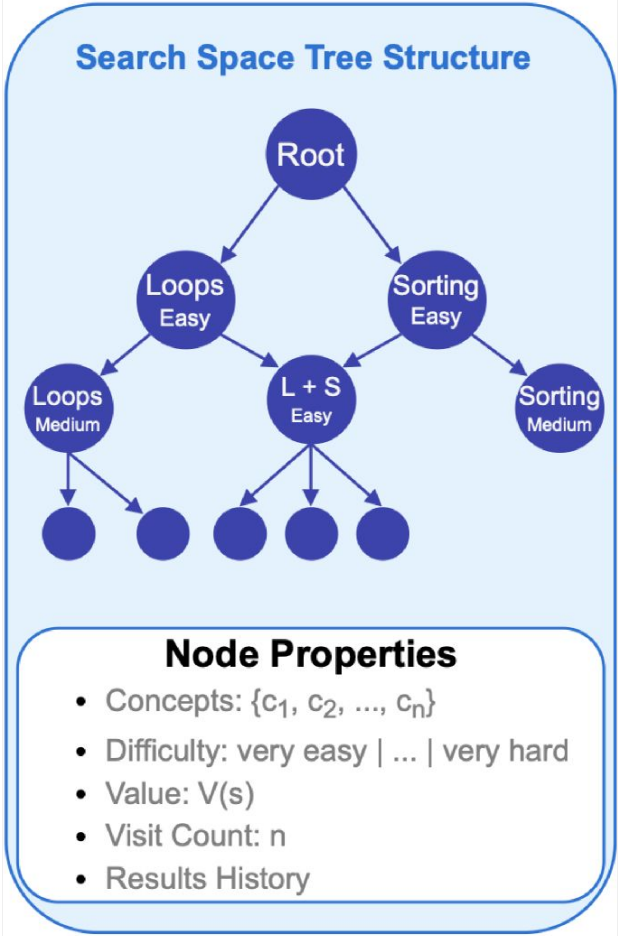


Majdinasab, V. et al., Prism: Dynamic and Flexible Benchmarking of LLMs Code Generation with Monte Carlo Tree Search, Arxiv (2025)

Search Space

Encode all possible evaluation scenarios as an MDP, which can be modeled as a search tree:

- Each state corresponds to a unique scenario.
- Actions at each state represent traversing the MDP.
- Transitions between states dependent on success.
- Transition probabilities dependent on previous received rewards.



Monte-Carlo Tree Search

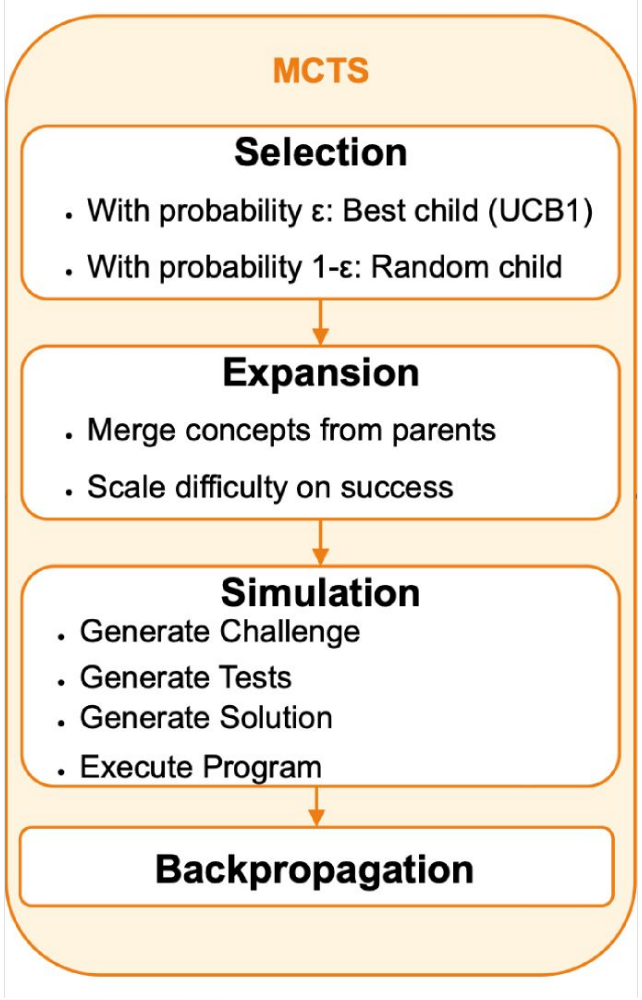
Traversal requires a policy.

The policy needs to be external to the model being evaluated.

Policy needs to select the state which maximizes information gain:

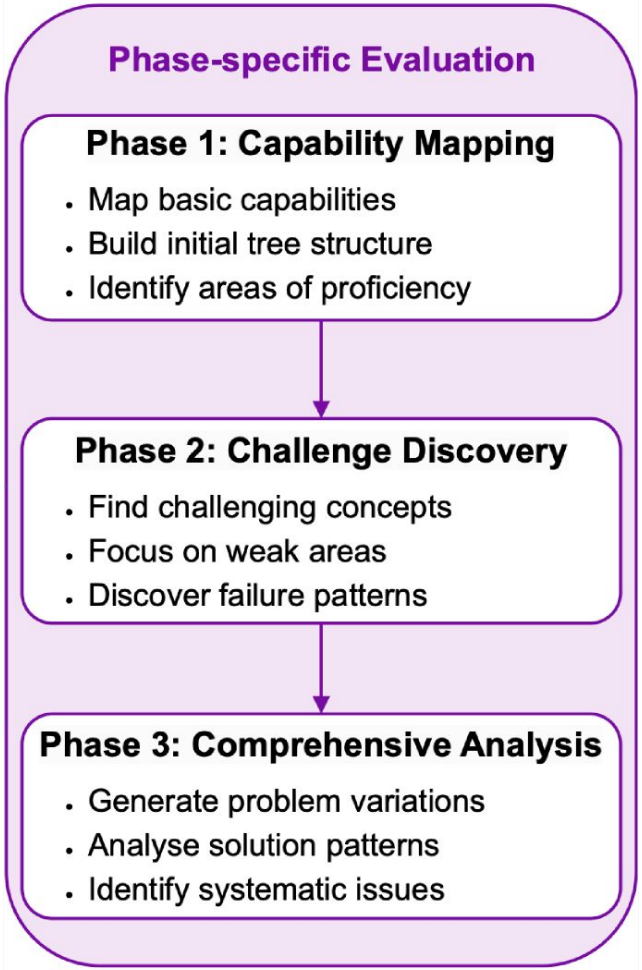
MCTS selects the best possible node at each iteration.

ϵ-greedy policy balances exploration/exploitation.

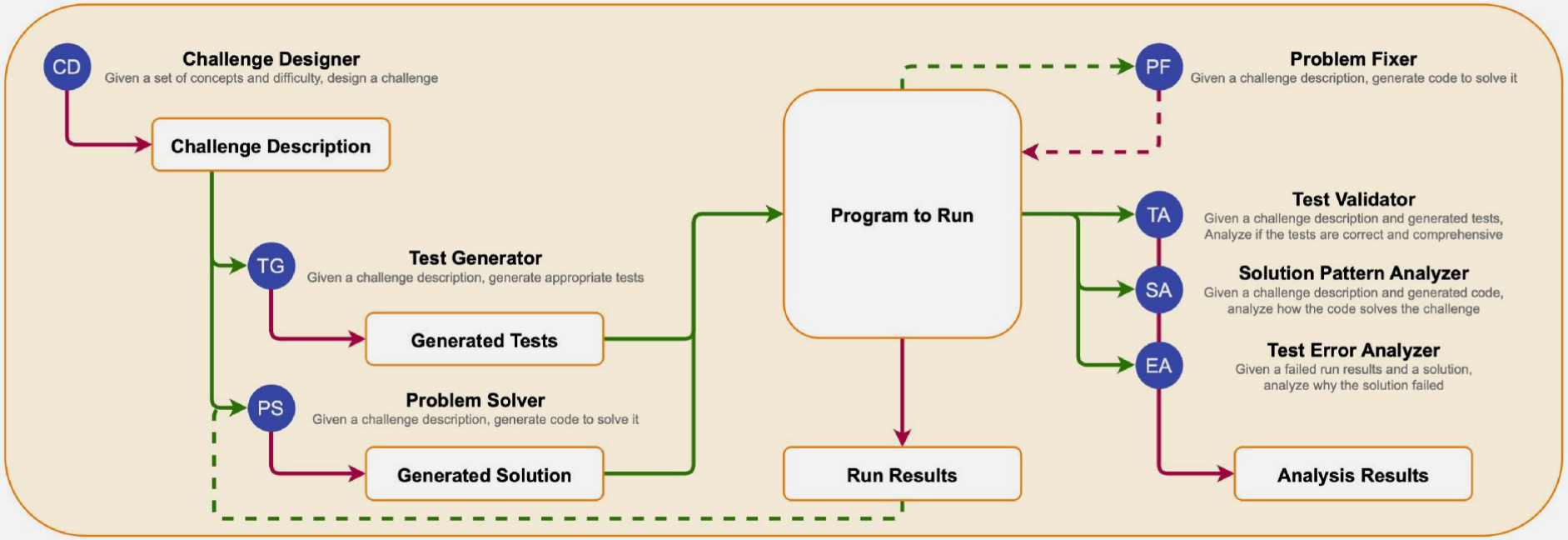


Multi-Phase Evaluation

Comprehensive evaluation requires optimizing differing objectives:
Optimizing for task success reveals capability boundaries.
Optimizing for task failure reveals consistent failure modes.

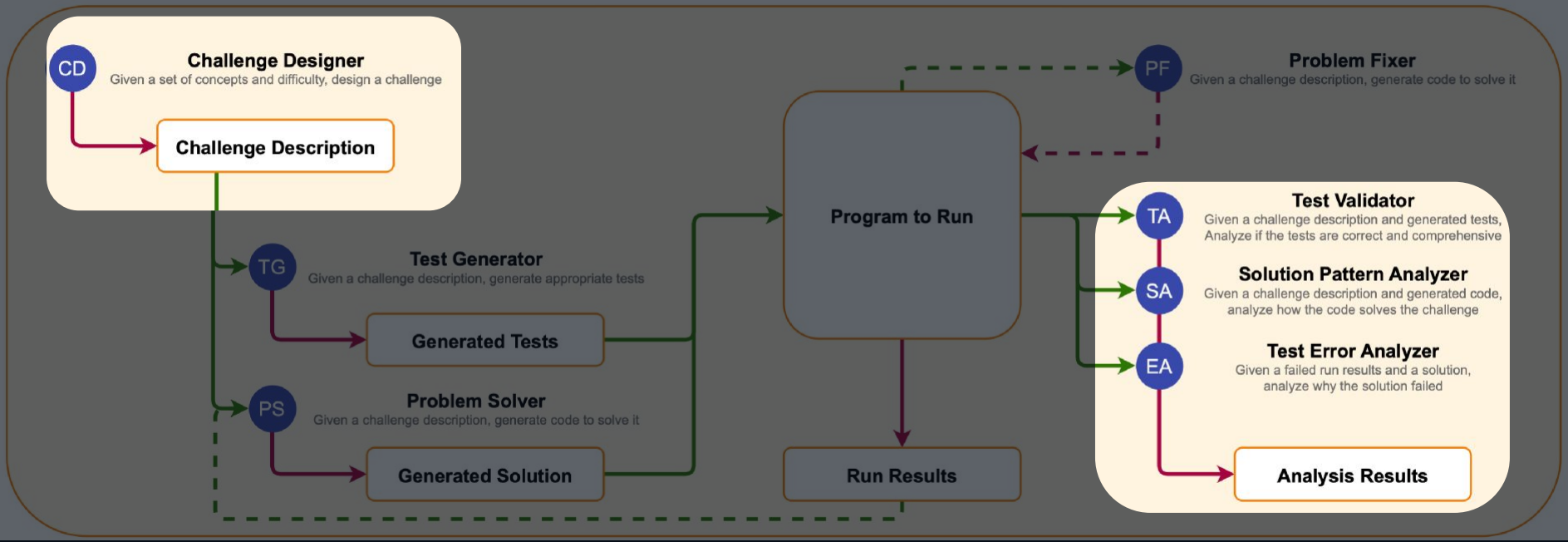


PrismBench



Multi-agent Interactive Sandbox for LLM Coder Dynamic Benchmarking

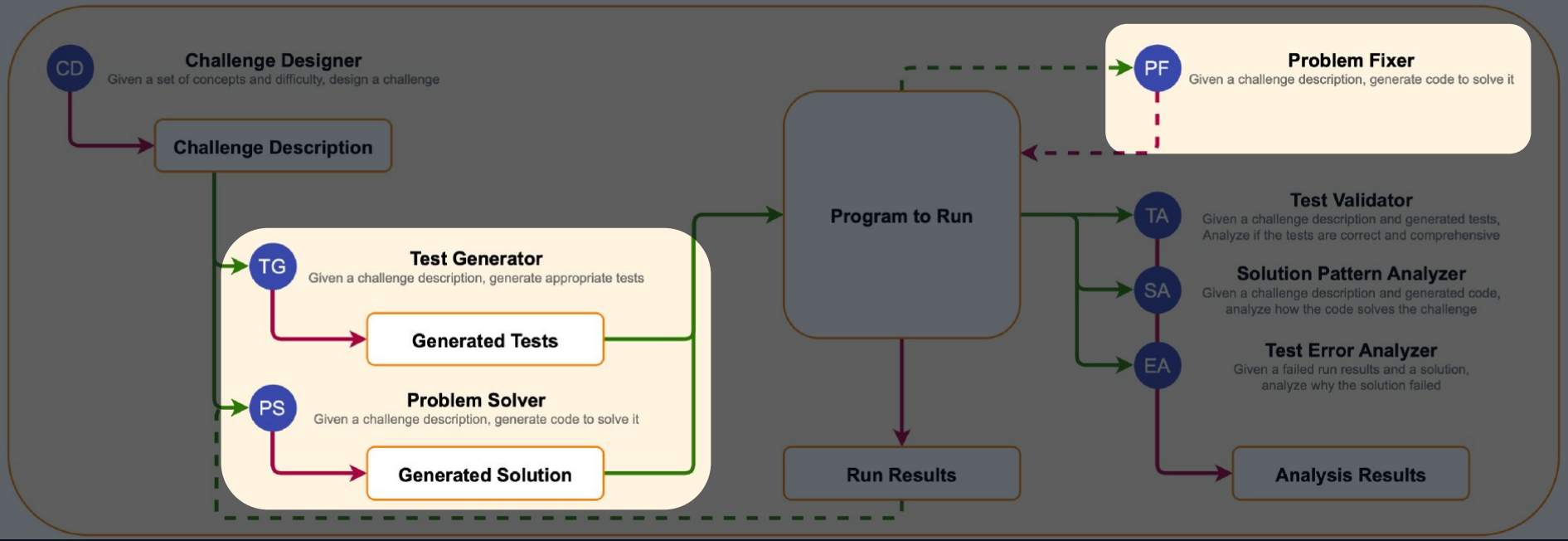
PrismBench



Challenge Designer: Generates a new coding problem for the node's concepts/difficulty

Analysis Agents: Inspect execution results, identify errors, and analyze failure patterns

PrismBench



Problem Solver: Attempts to solve the problem without knowing what the tests are

Test Generator: Creates tests to evaluate possible solutions without knowing the solution

Problem Fixer: Attempts to fix a solution or test that consistently fails

Assigning Rewards to Actions

Every agent action can be evaluated with a custom reward:

- **Test Generator:** Reward for correct tests
- **Solver:** Reward for correctness, efficiency
- **Fixer:** Reward for effective repairs

These rewards can be customized for any measurable quality.

➔ This allows us to **pinpoint which skills the model excels at and where it struggles**.

Concept Mastery Evaluation Results



Majdinasab, V. et al., Prism: Dynamic and Flexible Benchmarking of LLMs Code Generation with Monte Carlo Tree Search, Arxiv (2025)

Check LLMs Evals with PrismBench

Leaderboard

Thank you for attending



Questions ?