Promoting Effective Computer Science Education Through Interaction Testing for Requirement Engineering

Summary of Work

Introductory computer courses are **inaccessible** to non-computing majors. AI coding assistants enable previously non-technical students to **pursue computer science** through **"vibe coding,"** necessitating new teaching methodologies for requirement engineering, which improves AI-generated code. Thus, this work **uses software testing as a pedagogical vehicle to teach requirements engineering.** We propose using Large Language Models (LLMs) to **simulate help-seeking behaviors**, enabling students to learn software testing by teaching the LLM. The LLM iteratively **probes requirements** by asking the real student to develop and define test-case inputs and outputs from ill-defined system descriptions, which **scaffolds requirement engineering.**

Contribution

Review of **120+ papers** on computing education, requirement education, and software testing to develop a schema of testing-for-requirement engineering. To better equip end-user programmers with the necessary skills to engage with AI-coding assistants, this research found that educators should consider using **software testing as a pedagogical scaffold for requirement engineering.** By using LLMs to facilitate multi-user software testing, we may not just **improve requirement engineering**, but we may also help to increase **equity and diversity** in the computing field. Many students taking computer science courses, particularly those who are not majoring in computer science, care more about **developing a computer science vocabulary**, and LLMs now enable CS education to better serve these populations while teaching them core **computational thinking** skills that will improve their end-user programming abilities with AI.

Gap

Related work	Software testing	Multi-user system	Teachable agent	Teaches RE	AI-assisted prog.	Ill-defined specs	Pedagogical scaffold
ROPE *Ma et al., 2025*	✗	✗	✗	✓	Partial	✓	✓
Probeable Problems *Pawagi & Kumar, 2024*	✓	✗	✗	✗	Partial	✓	✓
POPT *Lustosa Neto et al., 2013*	✓	✗	✗	✗	✗	✓	✓
HypoCompass *Shen et al., 2024*	✓	✗	✓	✗	✗	✓	✓
Teach AI How to Code *Jin et al., 2024*	✗	✗	✓	✗	✓	✗	✗
ChainForge *Arawjo et al., 2024*	✗	✗	✗	Partial	✓	✗	Partial
Web-CAT *Edwards, 2003*	✓	Partial	✗	Partial	✗	Partial	✓
This work *Mitchell-Pullman, 2026*	✓	✓	✓	✓	✓	✓	✓

✓ = Yes Partial = Partial ✗ = No

Previous work **fails at incorporating Large Language Models (LLMs)** into a testing pedagogy that scaffolds requirement engineering. Additionally, lots of **AI-assisted pedagogies** depend too much on LLMs and **ignore past research** on teaching requirement engineering and testing.

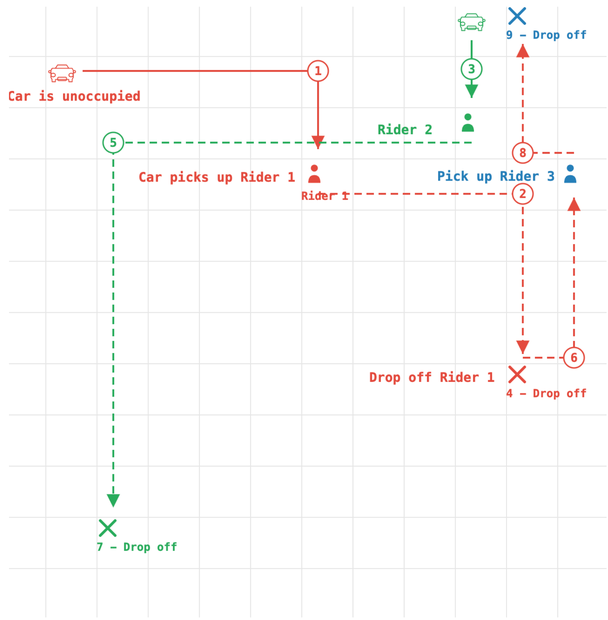
Ride Share Matching for Requirement Engineering

- **LLM simulates a student** asking for help on **writing test cases and requirements** for their ride share matching program. The **student acts as a TA** to the LLM student by proposing **test cases** to evaluate the quality of the code.
- Students define data types, interactions across users, and evaluate edge cases from **poorly-defined** specifications.
- If a novice were to write requirements for a ride share matching system, we expect them to focus on the **structure** of the matching system without considering the system's **behavior** and **function**, which would cause errors.
- Instead of writing technical **test cases**, the student defines system behavior using **natural language** for accessibility.

	NOVICE	EXPERT
Goals	0	3
Attributes	3	9
Requirements	2	9
Total	5	21

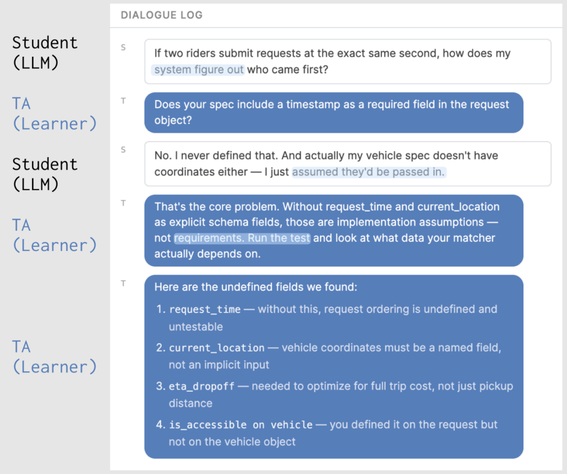
TEST CASE SCENARIO

There is one vehicle currently active in the system. It is unoccupied. Rider 1 requested a ride at 13:00 and is located four blocks east of the vehicle. Rider 2 requested a ride at 13:05 and is located three blocks south of Rider 1. Rider 3 requested a ride at 13:10 and is located five blocks east of Rider 2. The vehicle picks up Rider 1 (1), drives south to Rider 2 (3), picks up Rider 2 (5), drops off Rider 1 (4), picks up Rider 3 (8), drops off Rider 3 (9), and drops off Rider 2 (7). The routing must account for request time priority, vehicle occupancy during transit, and optimal chaining of pickups and dropoffs across all three riders.



Testing Paradigm for Requirement Engineering in AI-Assisted Programming

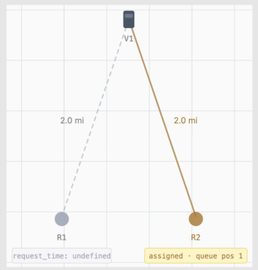
Messages Between LLM-Simulated Student and Real Learner



An **LLM simulates a help-seeking student** to **probe requirements** from the **real learner**, who acts as a computer science **class TA.**

The learner uses **software testing** to **identify and iterate upon** new **system requirements** and **teaches** them to the **simulated student.**

Students View the Testing Simulation for Visual Testing

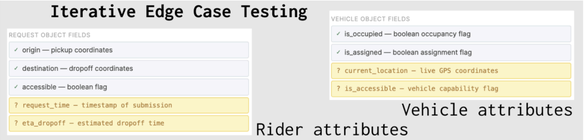


Rider 1 (left) and rider 2 (right) are **equidistant** from the ride share vehicle (top center).

However, we see **V1 has been matched with R1** in our test scenario.

The student must act as a TA and **guide the simulated student** to add additional **data classes** to make the **matching unambiguous.**

Iterative Edge Case Testing



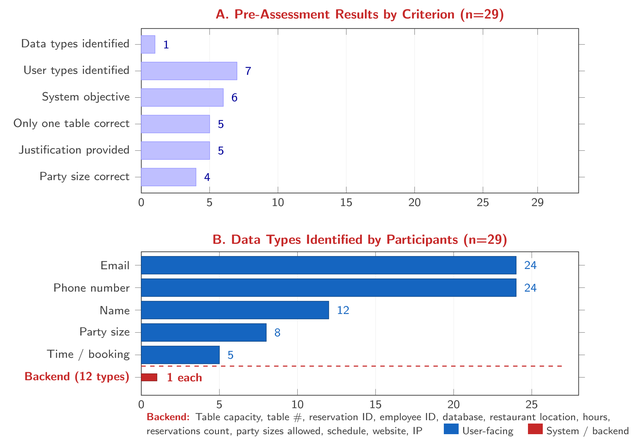
Rider attributes

Vehicle attributes

Specify Tests in Natural Language

Uses **tests** and **edge case** discovery to **elicit system requirements**

Quantitative Pre-Assessment Results



Iterative Pilot Studies

PILOT 1 — EXPLORATORY
n=1 • High school • Dec 2025
→ Teachable agent framework increased engagement
→ "What if / now what" framing improved RE understanding

PILOT 2 — CLASS DEPLOYMENT
n=29 • AP CSP • Jan 2026
→ Students conflated features with requirements
→ Omitted non-trivial data types; no cross-user testing

PILOT 3 — LARGE-SCALE
n=40 • AP CSA • Jan 2026
→ **Only 10%** demonstrated RE skills beyond domain knowledge
→ Focused on human behavior, not system behavior

PILOT 4 — EXPERT VS. NOVICE
n=2 • Expert vs. Novice • Feb 2026
→ Expert self-corrected omissions; novice missed system data
→ Requirements emerged from edge case identification

4 pilots • 72 participants • Nov 2025 – Feb 2026

Implications

Based on the **extensive literature review,** pilot studies with **72 participants,** and **four expert interviews** with CS teachers and Ph.D. students, I make a prediction about supporting **end-user programmers** in CS courses. LLMs enable us to use testing to scaffold requirement engineering education by emphasizing edge case identification, specification probing, system testing, and natural language articulation. This presents an opportunity for a new pedagogical requirement engineering paradigm that teaches the cognitive skills for prompt engineering based on existing **requirement engineering pedagogy frameworks.**

Future Work and Discussion

- RCT comparing experts and novices requirement engineering skills to make rubric more quantitative and empirical
- Compare learning outcomes on requirement engineering between multi-user testing paradigm and "business-as-usual" prompt engineering & requirement engineering pedagogy in a CS1 context.
- Open source materials for end-user and conversational programmers.
- Generalize to math, upper-division computer science, and other subjects.
- Exploratory study with a full class deployment at large institution.