Project Proposal

Project Name:

Build-Bench Challenge: Autonomous LLM Agents for Cross-Architecture Package Repair

Project Proposer:

Minghua Ma, minghuama@microsoft.com

Open Source:

Yes

Mentors:

Minghua Ma, minghuama@microsoft.com

Preferred Experience

Technical skill	Level
Python programming and software-engineering fundamentals	Required (for most or all team members)
Linux command line, Git, and Docker or container workflows	Required (for most or all team members)
LLM APIs, prompting, or agent frameworks	Required (at least one team member)
Build systems and package management, such as CMake, Make, Autotools, Debian, or RPM	Required (at least one team member)
Debugging compiler, linker, dependency, and test failures	Valuable
C/C++ or systems programming	Valuable
Software testing, CI/CD, and reproducible experiments	Valuable
Experience with x86_64, ARM/aarch64, or RISC-V environments	Nice to have
Program repair or automated software engineering	Nice to have

Project Background

Modern production systems must run reliably across cloud, edge, and emerging hardware platforms. When organizations migrate software packages between x86_64, ARM/aarch64, and RISC-V systems, builds often fail for reasons buried across source code, compiler toolchains, dependencies, configuration, and packaging logic. Resolving these failures is a real engineering need: software that cannot be built and validated on its deployment platform cannot be shipped, maintained, or operated reliably.

This task is challenging because no single error message explains the repair. Students must connect incomplete build evidence with a large, unfamiliar software package; distinguish the root cause from downstream failures; decide which files may safely change; and verify that a proposed fix is reproducible in a clean target environment. A useful solution must go beyond generating plausible code: it must produce a patch that genuinely builds and yields the expected artifacts under realistic constraints.

The Build-Bench Challenge | ICSE 2027 turns this production problem into a rigorous, hands-on project. By building an LLM repair agent, students will learn how AI systems can support systems engineering while confronting the limits of automated reasoning, tool use, validation, cost, and reliability. The work connects foundational skills in software engineering and systems with a timely applied AI problem, and gives the team a concrete opportunity to participate in an ICSE 2027 competition. Details, rules, data, and participation information are available on the challenge website.

Project Description

The team will design, implement, and evaluate an autonomous LLM-based software-repair agent for the Build-Bench Challenge.

The project will begin by reproducing and understanding the provided starter kit and baseline agent. The team will then develop an improved agent that can:

- Analyze package metadata, build scripts, source code, build logs, and target-architecture failure evidence.
- Form and test hypotheses about the causes of architecture-migration build failures.
- Select and execute permitted diagnostic and repair actions.
- Iteratively modify package files while preserving a clean, reproducible repair history.
- Validate candidate fixes through target-architecture builds and artifact checks.
- Manage runtime and LLM cost through effective tool selection, context management, and stopping criteria.

The final deliverable will be a reproducible, runnable agent submission, technical documentation, an experimental report on public development cases, and a short demo or presentation. The team should submit a qualified agent to the official ICSE 2027 competition.

Learning Outcomes

Students will gain practical experience with:

- Building and evaluating autonomous LLM agents for real software-engineering tasks.
- Diagnosing cross-platform and cross-architecture software build failures.
- Working with Linux package builds, compilers, dependencies, and build systems.
- Using reproducible environments, containers, version control, and automated testing.
- Designing agent tool interfaces, repair loops, validation strategies, and resource-aware workflows.
- Interpreting benchmark metrics, including verified success rate, runtime, and token consumption.
- Collaborating in an open-source engineering project and preparing a competition-quality software artifact.