

AI-Driven Embedded Testing

Rainer Poisel

Overview

- Introduction
- State of the Art
- Our Approach
- Demo / Results
- Outlook

About the Speaker

- Rainer Poisel
 - Embedded Systems DevSecOps
 - Embedded Systems CI/CT/CD
 - Process Automation
 - Security Testing
(IEC 62443, CRA)



Introduction

- **Actual Title:** Qualification of AI for Embedded Systems Testing: Accelerating DevSecOps Workflows

Research Hypotheses

1. Integrating LLMs into test case preparation reduces implementation time while maintaining human oversight and adaptability in embedded system testing.
2. Prompt Engineering alone can significantly improve test case preparation efficiency without fine-tuning the LLM.

Large Language Model (LLM)

Definition of LLMs:

Large Language Models (LLM for short) are powerful models designed to understand and generate human language.

Programming with AI:

- Conversational AI Assistants
- Integrated AI Assistants

Prompt Engineering

Prompt engineering is the process of structuring or crafting an instruction in order to produce the best possible output from a generative artificial intelligence (AI) model.

– Dina Genkina (IEEE Spectrum)

Motivation

- Product Certification Process
 - IEC 62443
 - CRA / EUCC
- Automate “Boring Stuff”
- Increase Efficiency
- Faster Adaption
- Developers vs. Test Engineers

The 70% Problem

Honest reflections from coding with AI so far as a non-engineer:

It can get you 70% of the way there, but that last 30% is frustrating. It keeps taking one step forward and two steps backward with new bugs, issues, etc.

– Peter Yang (quoted by Addy Osmani)

State of the Art

Literature Research (I)

IEC 62443 and DevOps/DevSecOps:

- Security Tools lack complete Coverage
- Automation remains difficult
- Shifting Security Left

Literature Research (II)

LLM-based Test Generation:

- Confirm existing functionality vs. uncover defects
- Models need project-specific fine-tuning
- Human intervention is required
- Large-scale test generation is expensive

The Agile V-Model

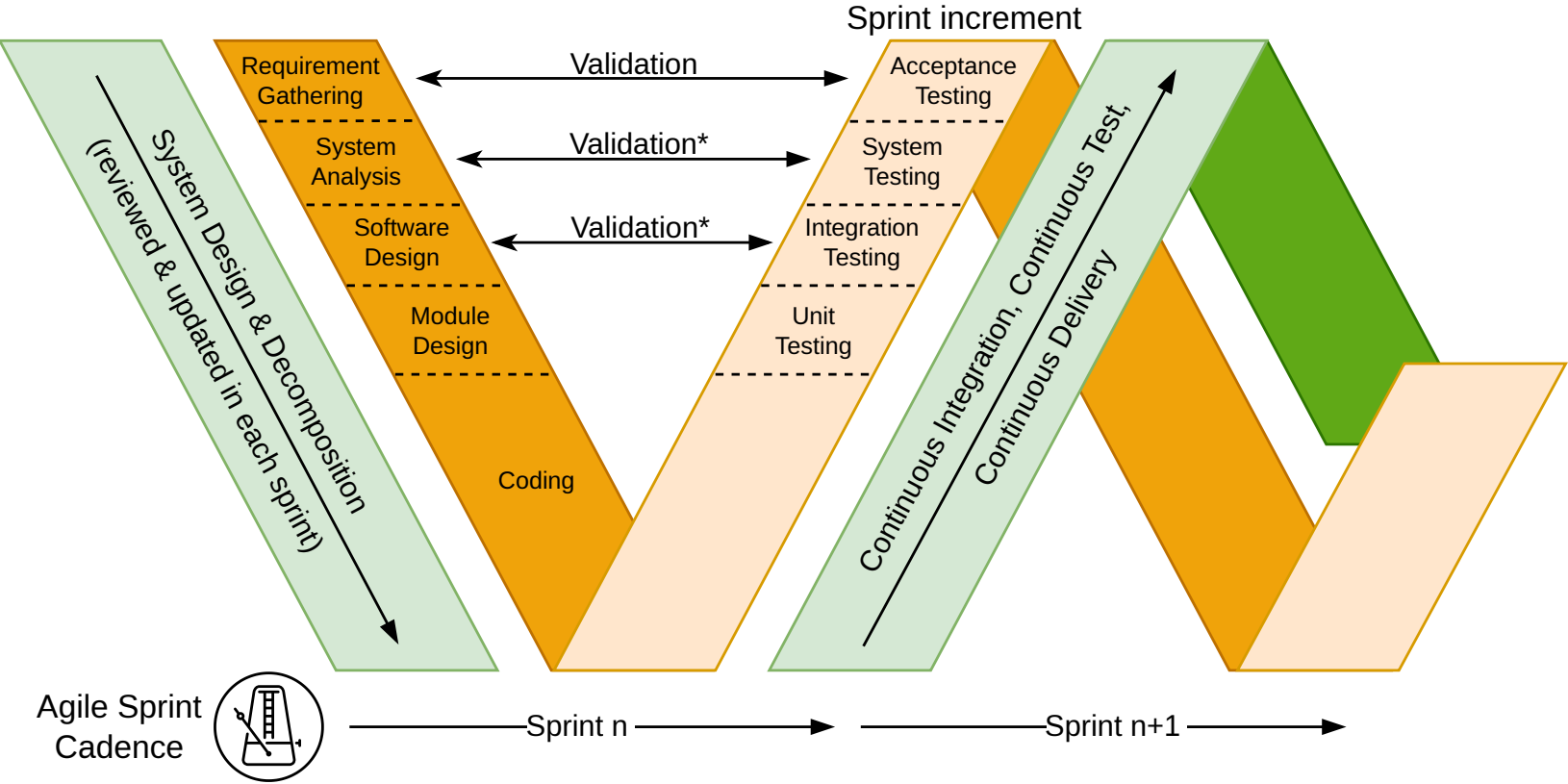


Figure 1: The Agile V-Model (AIoT Playbook)

Optimizing Tests Development (I)

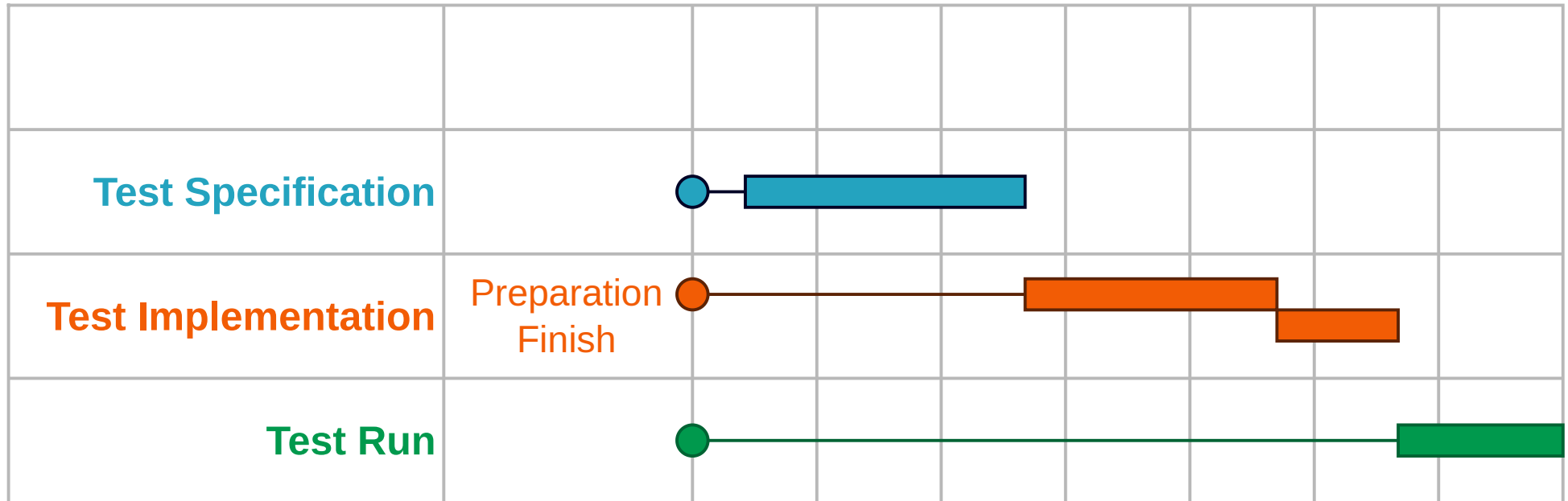


Figure 2: Test Development Phases (Before)

Optimizing Tests Development (II)

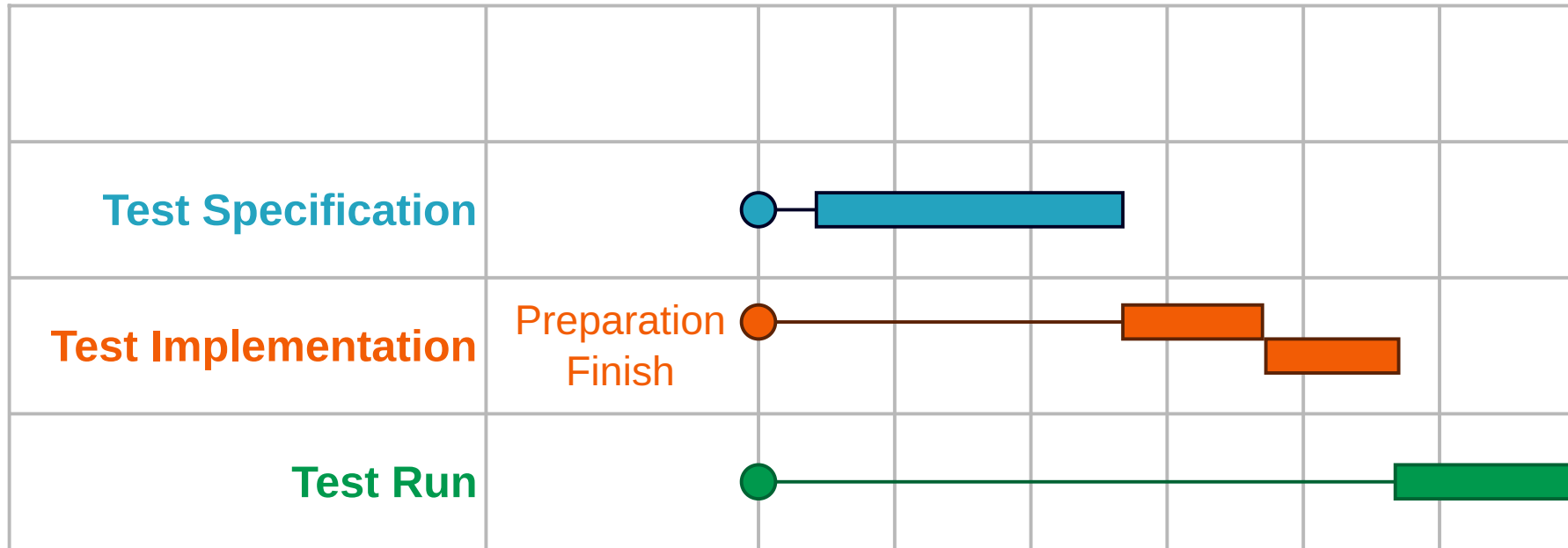


Figure 3: Test Development Phases (After)

Development Velocity Gain

- Manual Test Preparation: **30 Minutes**
 - Link: Test Spec ↔ Test Implementation
 - Implement Test Structure / Fixtures
 - Document Test
- Reviewing AI-generated Test Cases: **10 Minutes**

Generating Test Case Preparation with AI: ≈ 2 ct

System Overview

Components

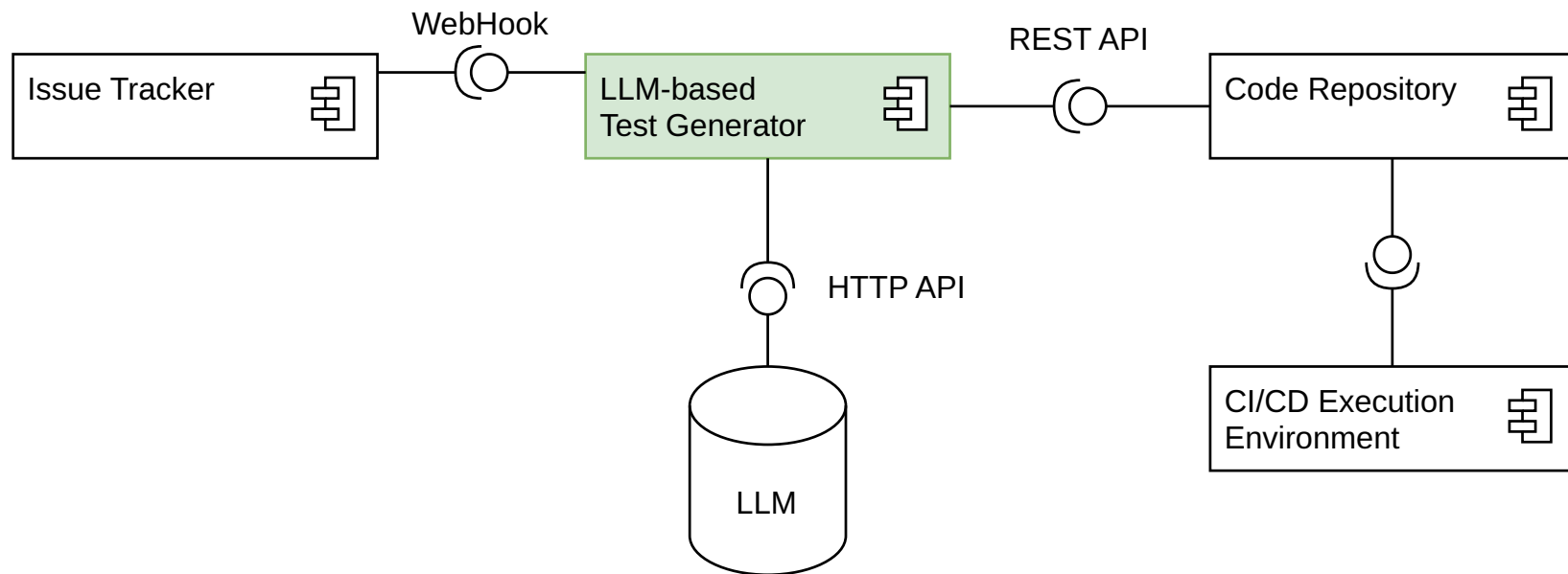


Figure 4: Components Diagram

Test Generation Sequence

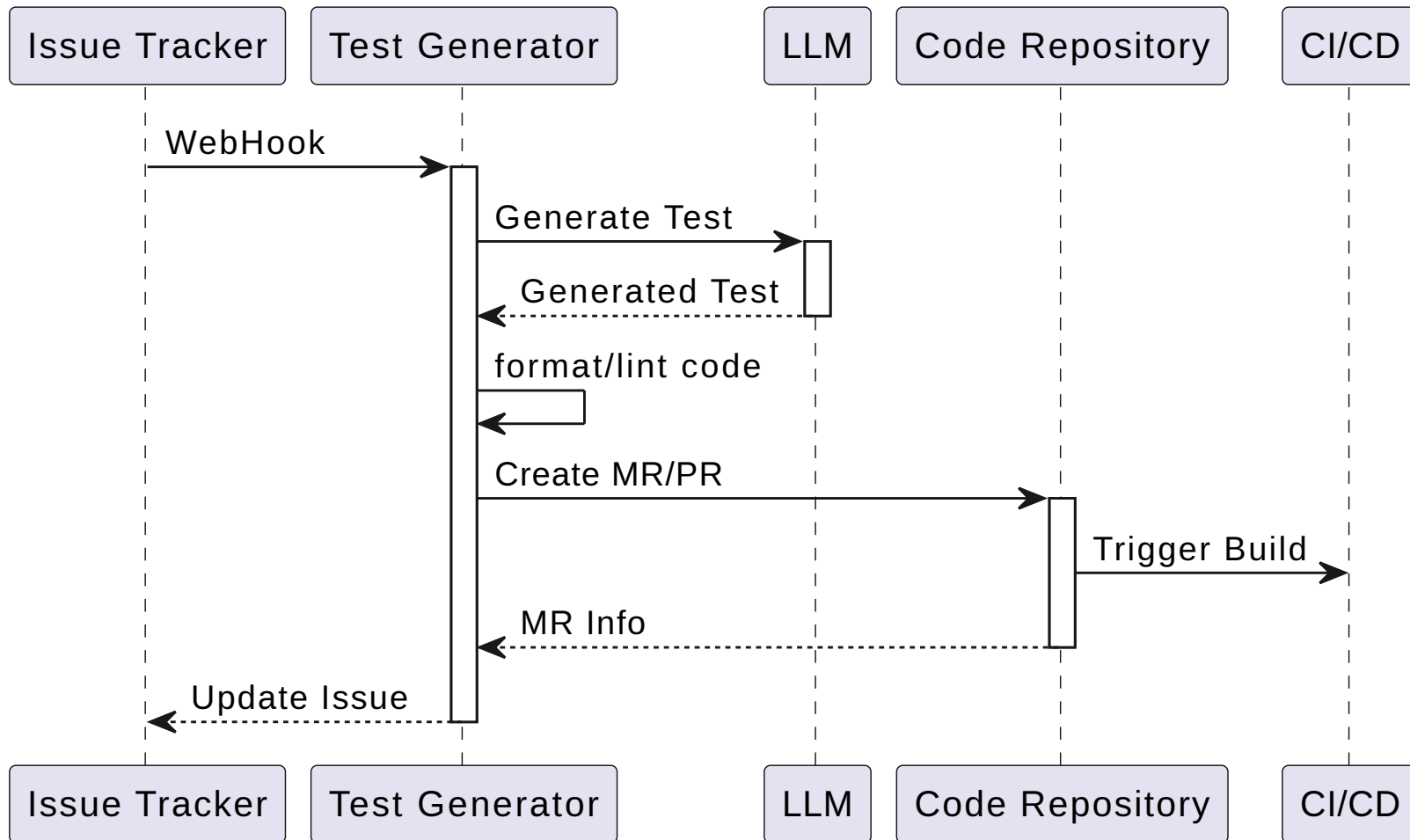


Figure 5: Sequence Diagram

Prepared Command Prompt (I)

```
1 Generate a pytest from the following description in markdown format:
2 ```markdown
3 {description}
4 ```
5
6 Terminology:
7
8 < ... >
9
10 Important:
11
12 < ... >
13
14 Code Guidelines:
15
16 < ... >
```

Figure 7: Prepared Command Prompt

Prepared Command Prompt (II)

- **Terminology:** Abbreviations
- **Important:**
 - High-Level Test Code Requirements
 - General Structure of generated Code
- **Code Guidelines:**
 - Description of the test infrastructure
 - Libraries/Techniques to follow/avoid

Large Language Model

- Open-Source Models
 - Self-Hosted: Privacy
 - Hosted: Low computing resources available
- Models evaluated:
 - `meta/meta-llama-3.1-405b-instruct`
 - `meta/meta-llama-3-70b-instruct`

Software Engineering and Architecture Roles

- System/Software Architect
- Security Architect
- Software Developer
- Quality Assurance:
 - Framework/Library Developer
 - Test Developer

Test Framework

pytest

- Python-based
- 12.5k Stars on GitHub
- Well-known to LLMs
- Easy to use

labgrid (I)

- Local/Distributed Infrastructure
- Pytest Integration
- CLI/Library Usage
- Drivers
- Strategies
- Virtualization Support



labgrid (II)

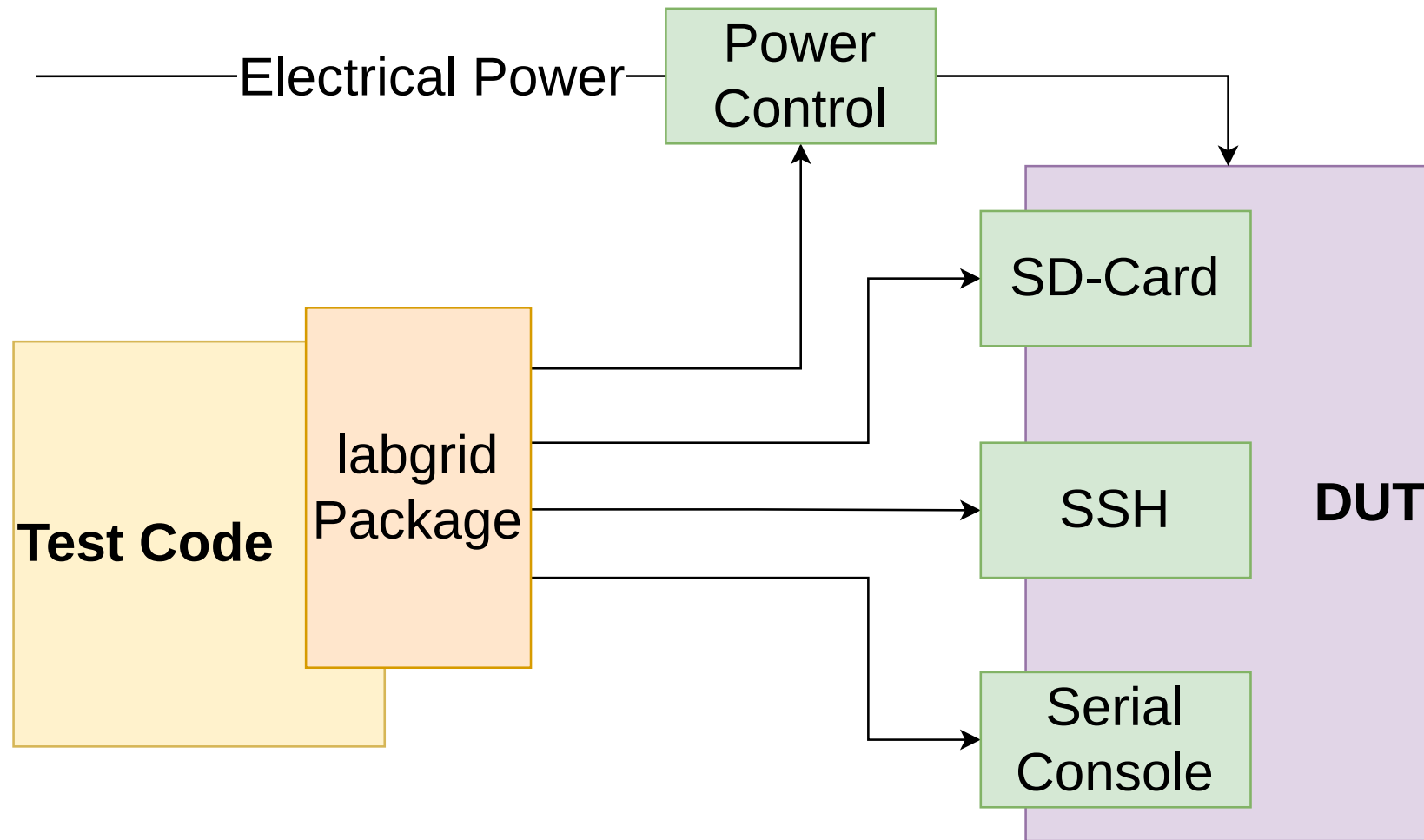


Figure 8: Labgrid Local Architecture

labgrid (III)

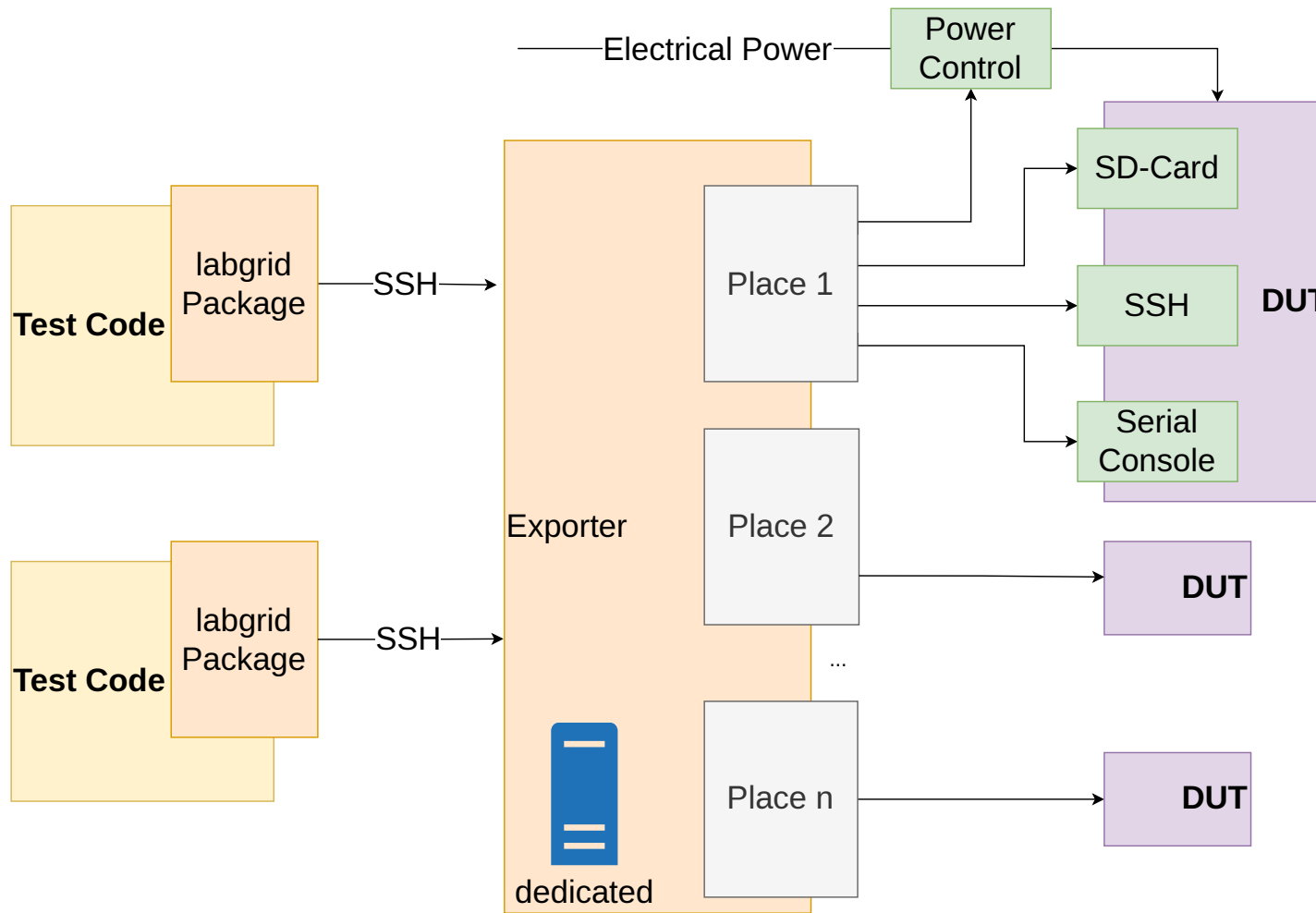


Figure 9: Labgrid Distributed Architecture

labgrid: Example (I)

Task: Check that `uname -s` returns the string `Linux`

```
1 from labgrid.util.ssh import SSHConnection
2
3 def test_uname(ssh_cmd: SSHConnection) -> None:
4     stdout = ssh_cmd.run_check("uname -s")
5     assert stdout == "Linux"
```

Figure 10: Simple pytest/labgrid sample

labgrid: Example (II)

Not in the code:

- Power on
- Pass bootloader
- Wait for login
- Wait for shell
- Configure DHCP client (if required)
- Determine IP

Demo: Workflow

The Test Case

- Determine Listening TCP Ports
 - Expected Listening Ports: [22, 80, 443]
- Determine Listening UDP Ports
 - Expected Listening Ports: []

Future Vision

(Still) Limited Practical Suitability

- Simple Complexity
- Limits of Prompt Engineering
- Hosted LLMs Used

LLM fine-tuning

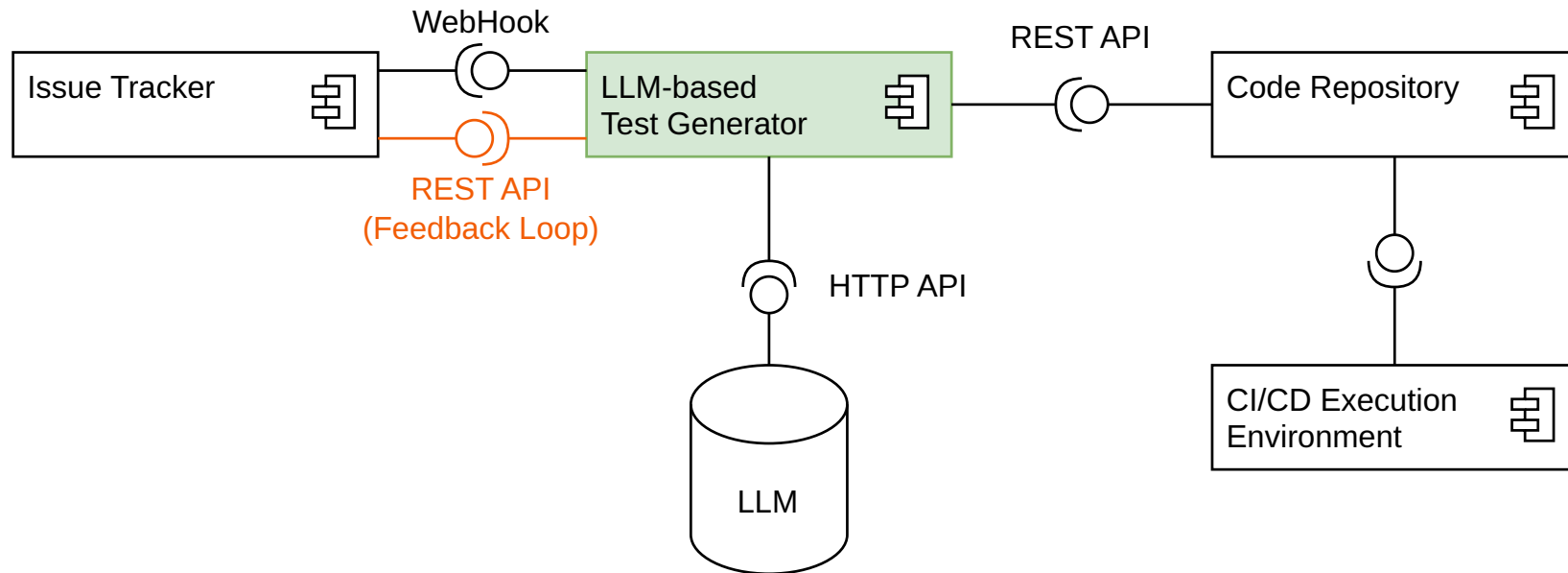


Figure 11: Components with Feedback Loop

No-Code Security

- Inspired by “No-Code Development”
- Security Audits and System Analyses w/o Coding
- Empowering non-technical Stakeholders
- Proactive Security Culture

Call to Action

- Say Hi 🙌
- Connect on LinkedIn: [in/rainer-poisel](https://www.linkedin.com/in/rainer-poisel)

Addendum

References

pytest

<https://pytest.org>

labgrid

<https://pengutronix.de/en/software/labgrid.html>

labgrid QEMU Sample

<https://github.com/Embedded-Focus/labgrid-qemu-sample>

Our Services

- Embedded CI/CD
- Software Development
- DevSecOps Trainings
- Automation
- Modernization
- Supply Chain Security

