

Agent-Driven DevSecOps

Rainer Poisel



Overview

- Introduction (3 Min.)
- Agentic AI + DevSecOps (10 Min.)
- Usage Scenarios + Demos (10 Min.)
- Outlook (2 Min.)

About the Speaker

- Rainer Poisel
 - Embedded Systems DevSecOps
 - Embedded Systems CI/CT/CD
 - Development Productivity



Process Automation, Security Testing (IEC 62443, CRA)

Introduction

- **Original Title:**

Agent-Driven DevSecOps: Transforming Embedded Software Development

- **Research Question:**

How can current AI technologies be applied to improve the effectiveness of DevSecOps workflows?

Motivational Example (I)

- Automatic Failure Analysis
 - Failed Tests in CI/CD
 - Keep state of DUT(s)
 - Analyze Logs
 - Perform automatic analysis on DUT

Motivational Example (II)

- Product Certification Process
 - IEC 62443
 - CRA / EUCC

Key Contributions

This talk presents:

1. **Containerized Multi-Agent Infrastructure**

ACP + MCP + Agent Skills with security isolation

2. **Labgrid MCP Server**

AI-assisted CI/CD failure triage with controlled device access

3. **Lessons Learned & Roadmap**

Practical experience + A2A for agent collaboration

Terminology

DevOps

DevOps combines development (Dev) and operations (Ops) to **unite** people, process, and technology in application planning, development, delivery, and operations.

– Source: Microsoft

DevSecOps

DevSecOps, which stands for development, security, and operations, is a framework that **integrates** security into all phases of the software development lifecycle.

– Source: Microsoft

Large Language Models (LLMs)

- Most used type of a probabilistic generative model
- Can understand & generate:
 - natural language,
 - code, and
 - structured data.
- Key enabler for automation in DevSecOps

AI Agent

- LLM-based system that can reason
- Uses tools, data, and memory
- Can execute multi-step tasks autonomously

Building Blocks Overview

Protocols

Communication standards between components

- **ACP:** Client ↔ Agent
- **MCP:** AI ↔ Tools/Data
- **A2A:** Agent ↔ Agent

Skills

Modular capabilities for agents

- Domain expertise
- Reusable workflows
- Progressive disclosure

AI Protocols

ACP: Agent Client Protocol

- Open standard
- Client ↔ agent interface
- Request/response and streaming patterns

Mission: Decouple clients from agent implementations

ACP: Integration

Interfaces:

- JSON-RPC
- SDKs in common languages
- IDE/CLI clients

Integrations:

- Emacs: agent-shell
- Neovim
- Zed

MCP: Model Context Protocol

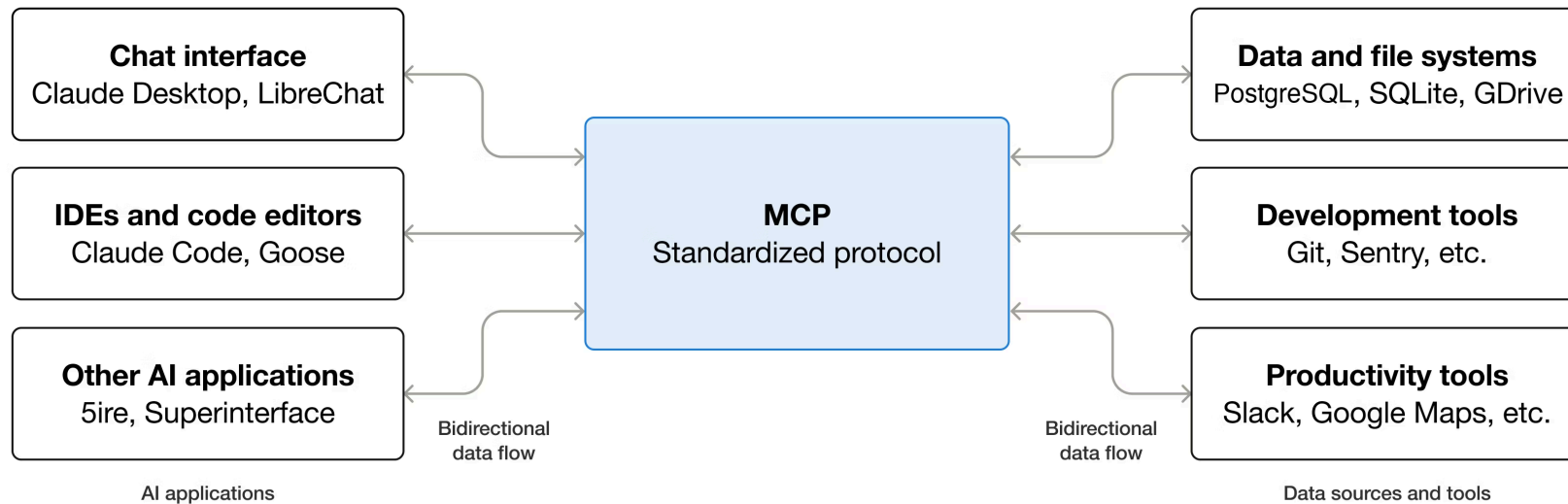


Figure 1: What is the Model Context Protocol (MCP)?

MCP: Core Primitives

- **Resources:** Data sources exposed by servers
- **Tools:** Executable functions
- **Prompts:** Pre-configured workflows

MCP: Implementation Example

```
1 from fastmcp import FastMCP
2
3 mcp = FastMCP("Demo 🚀")
4
5 @mcp.tool
6 def add(a: int, b: int) -> int:
7     """Add two numbers"""
8     return a + b
9
10 if __name__ == "__main__":
11     mcp.run()
```

Minimal FastMCP Example

MCP: Overkill

- MCP can be heavyweight
- Consider letting your agents use CLIs

A2A: Agent2Agent

- Open standard from Google → Linux Foundation
- Universal language for agent interoperability
- JSON-RPC 2.0 over HTTP(S) + Server-Sent Events (SSE)
- Key principle:
 - Agents collaborate without exposing internal logic.

A2A: Architecture

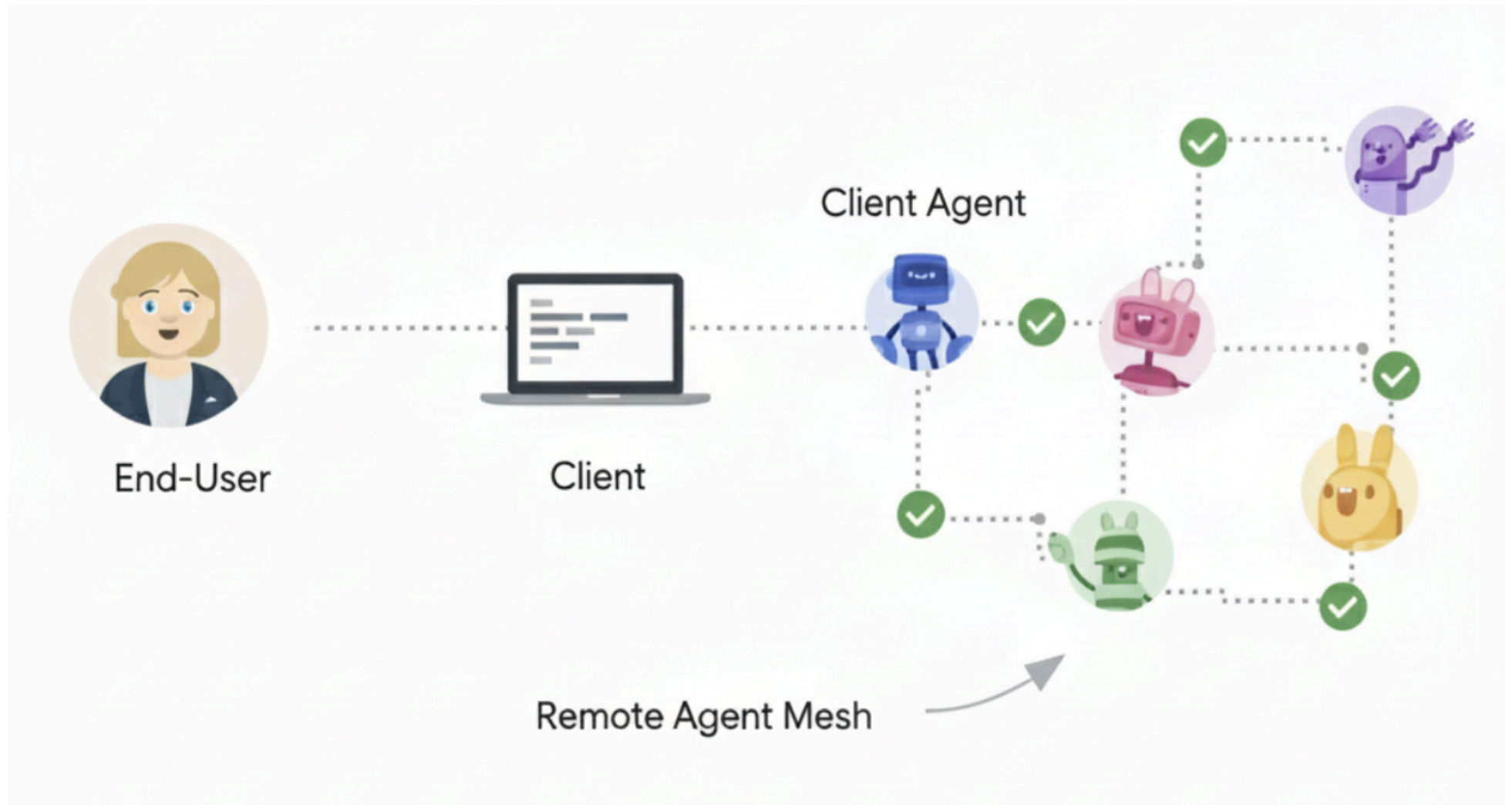


Figure 3: Core Actors in A2A Interactions

Protocol Frameworks

Incomplete list:

- LangChain
- Pydantic AI
- FastMCP

Agent Skills

Agent Skills: What Is It?

- Modular packages that extend AI agent capabilities
- Task-specific expertise as reusable components
- Not a protocol - they're capabilities agents can load/use

Think of Skills as plugins for AI agents.

Agent Skills: Structure

Each skill contains:

```
skill-name/  
├── SKILL.md           # Instructions & metadata (required)  
├── scripts/          # Executable code (optional)  
├── references/       # Documentation (optional)  
└── assets/           # Templates, configs (optional)
```

SKILL.md frontmatter:

```
1 ---  
2 name: security-test-generator  
3 description: Generate security tests for embedded systems  
4 ---
```

Agent Skills: Context Window

- Context window kept as small as possible
- Three-level loading strategy to manage context efficiently:
 1. Metadata, ~100 tokens
 2. Instructions, <5k tokens
 3. Resources, effectively unlimited

DevSecOps

AI Use-Case:

ACP Project Agents

The Problem

- Locked into specific IDEs
- Broad, unrestricted system access
- Can't easily switch between agents/vendors
- No standard editor ↔ agent protocol

Our Need

We want to use multiple AI agents simultaneously:

- Different strengths
- Unified interface
- Security isolation

Our Solution: Agent Circus

- Zero configuration¹
- Editor independence
- Supports major AI models
- Usage of standardized protocols
- Container isolation

Easily choose the best agent for each task.

Related Work

- Coding Agent VMs on NixOS with microvm.nix
- A field guide to sandboxes for AI
- YOLOArena (EclipseSource)
- AgentPool

Architecture Overview

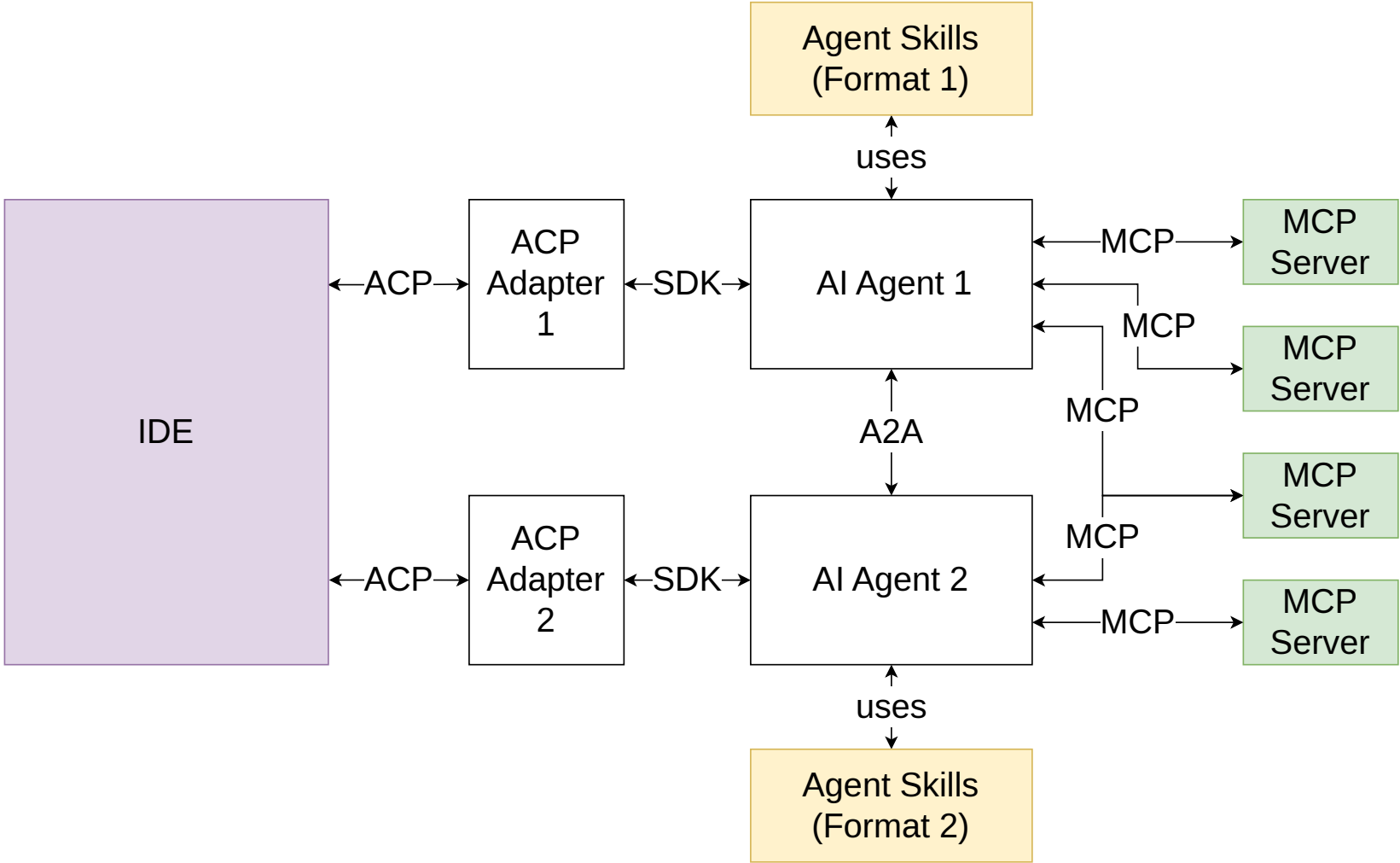


Figure 5: AI Architecture Using Open Standards

Agent Containerization

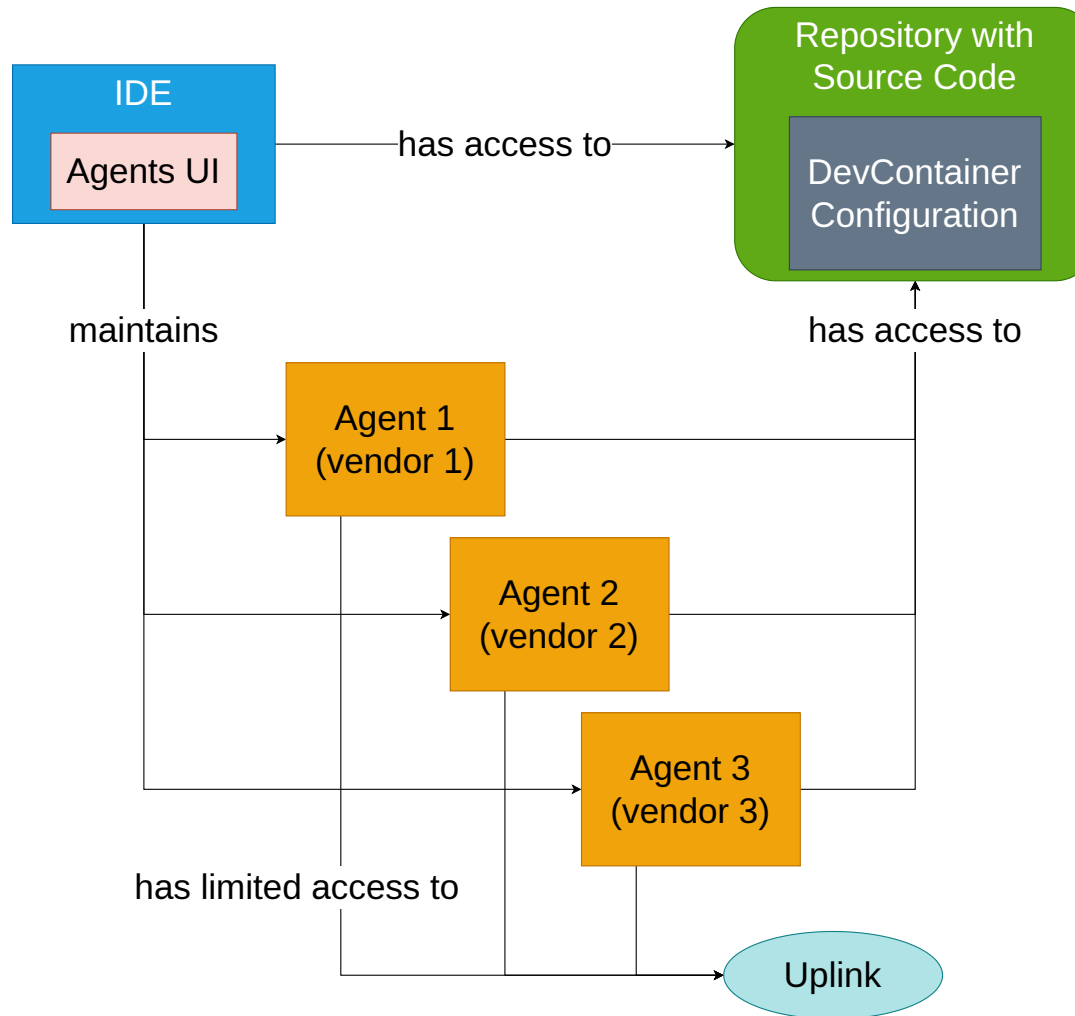


Figure 6: Containerized multi-agent architecture with IDE integration

Workflow Example (I)

Task 1: Architectural Refactoring

- Model: Claude Code
- Sub-Tasks:
 - Ingest Requirements
 - Refactor Implementation
 - Update Docs

Workflow Example (II)

Task 2: Test Generation and Review

- Model: OpenAI Codex
- Sub-Tasks:
 - Ingest Requirements
 - Implement Tests

Workflow Example (III)

Task 3: Review Changes

- Model: Mistral Devstral
- Sub-Tasks:
 - Ingest Requirements
 - Review Claude's Implementation
 - Review Codex's Tests

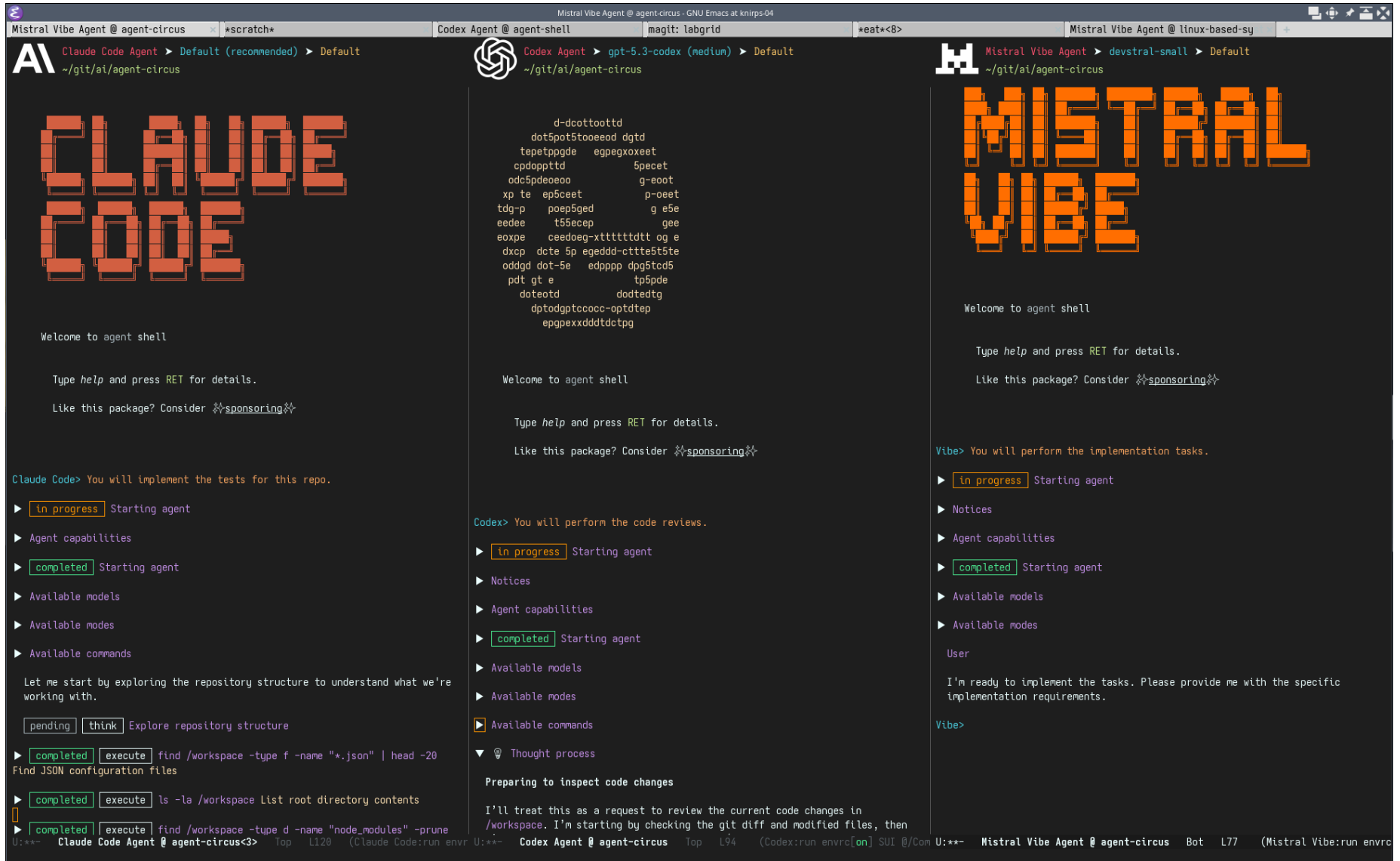
Current Challenges

- Not many editors support ACP yet
 - Emacs
 - Zed
 - Neovim
- Integration into existing projects
 - Agent environment
 - Human development environment

Project Implementation

- Code Repository:
 - <https://codeberg.org/Embedded-Focus/agent-circus/>
- MIT License
- Dependencies:
 - uv
 - docker compose

Screenshot



DevSecOps

AI Use-Case:

Labgrid MCP Server

Labgrid MCP Server: Intro (I)

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



Labgrid MCP Server: Intro (II)

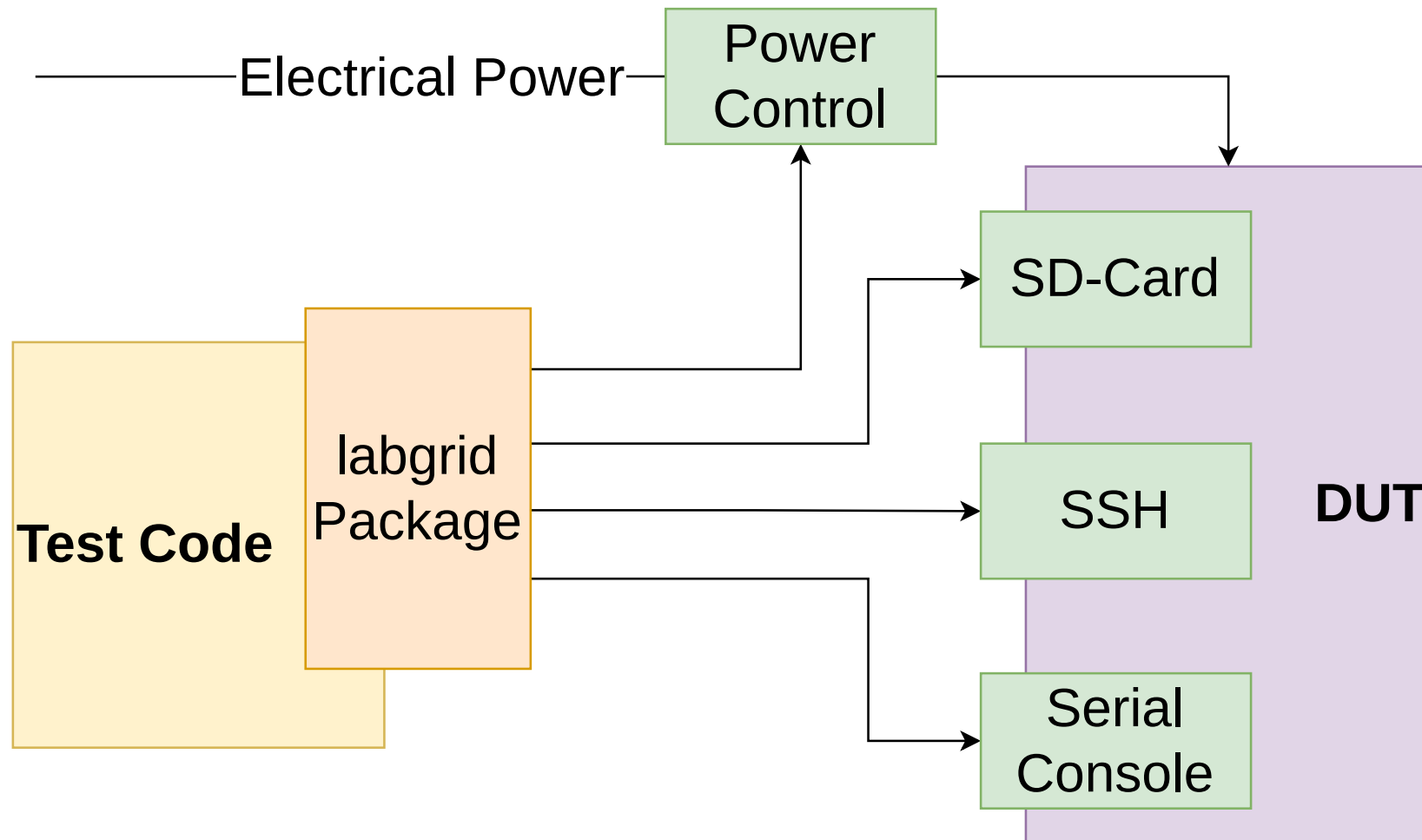


Figure 7: Labgrid Local Architecture

Labgrid MCP Server: Intro (III)

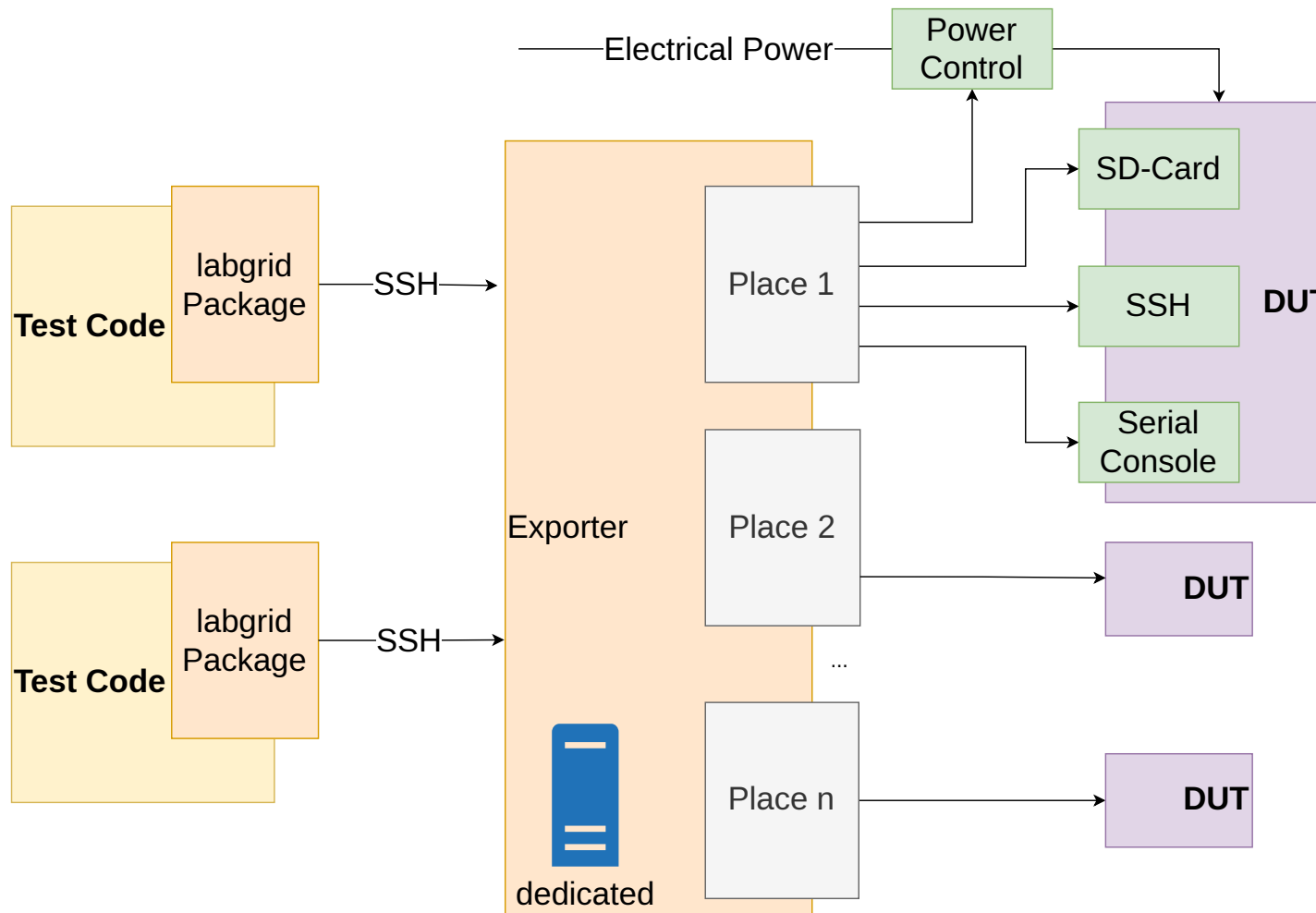


Figure 8: Labgrid Distributed Architecture

Labgrid MCP Server: Components

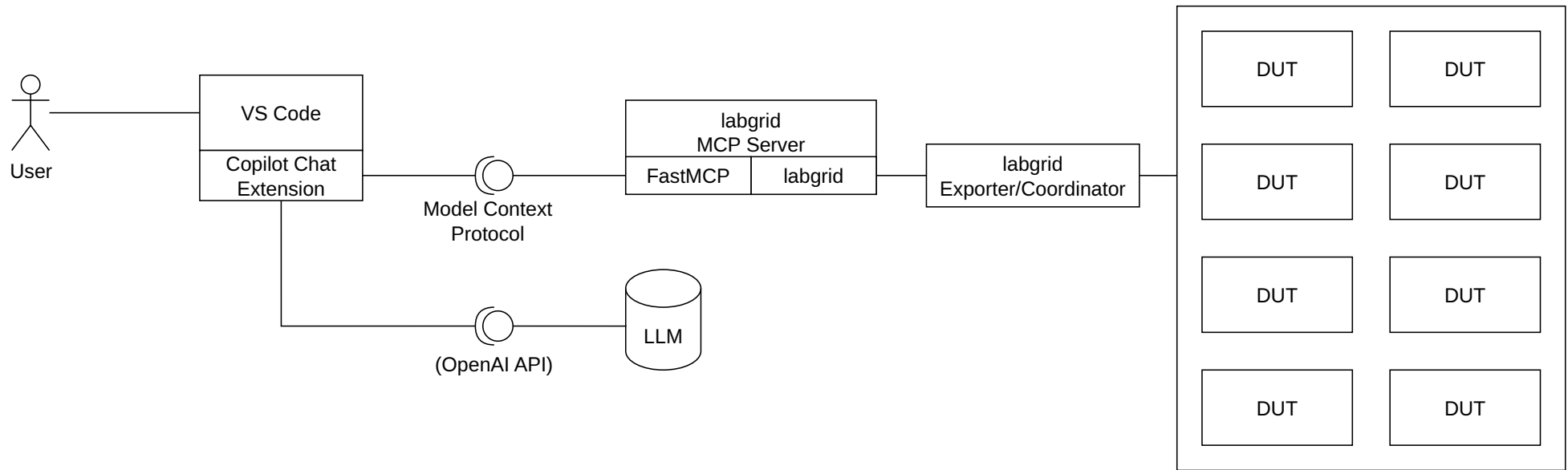


Figure 9: Components Diagram

Labgrid MCP Server: Sequence

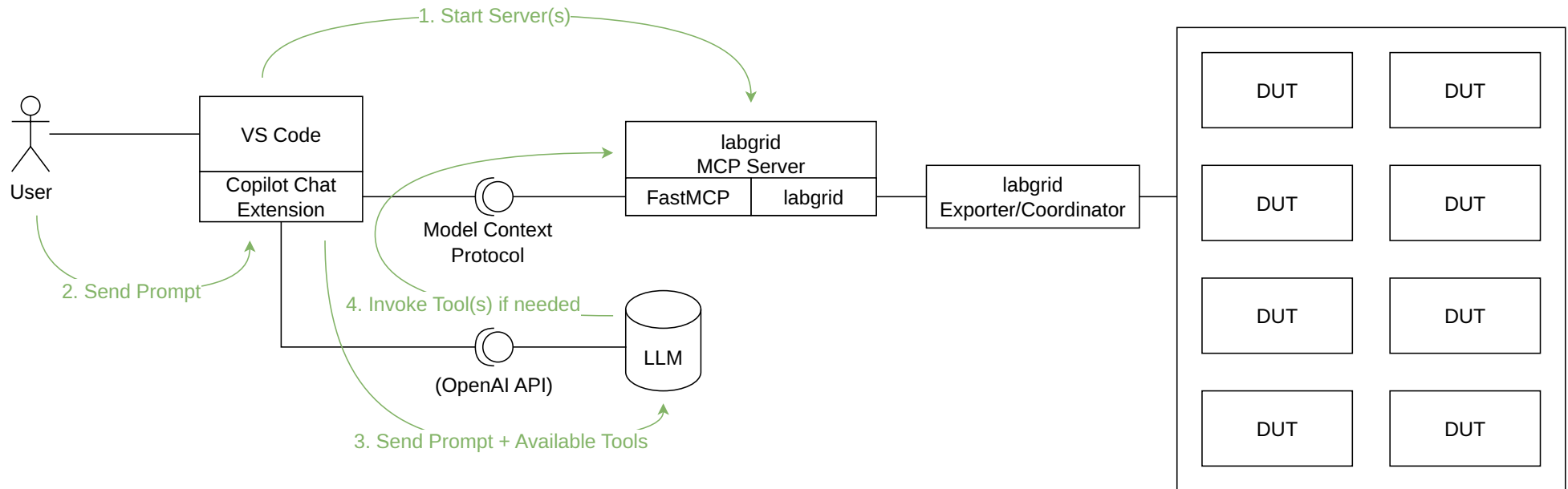


Figure 10: Components Diagram with Sequence

Labgrid MCP Server: Demo

Labgrid MCP Server: Results

Anecdotal Time Savings:

Metric	Before	After
CI Failure Triage	~15 min	~2 min
Manual Steps	Log inspection, research, device commands	Copy error, ask agent
Outcome	Found missing physical uplink	Same, with audit trail

DevSecOps

AI Use-Case:

A2A Future Vision

A2A: The Vision

Current Limitation:

Developers must manually select which agent to invoke for each task

A2A Solution:

Agents discover each other and delegate subtasks autonomously

A2A: Implementation

How It Works:

- Each agent publishes an AgentCard
- Advertises capabilities & skills
- Orchestrator queries available agents
- Delegates work without human intervention

A2A: Divide and Conquer

Potential Specialists:

- Security auditor agent
- Test generator agent
- Feature implementer agent
- Infrastructure agent

A2A: Potential Applications (I)

Agent-Orchestrated TDD for Compliance:

1. Test developer agent generates (failing) security tests
2. Tests are based on IEC 62443 or CRA requirements
3. Verifies they fail on hardware via labgrid MCP
4. Delegates implementation to feature agent
5. Iterates until tests pass
6. Security auditor agent verifies compliance constraints
7. Human oversight via git-based code review

A2A: Potential Applications (II)

Infrastructure Self-Healing:

- Detect CI failure patterns (offline devices, timeouts)
- Diagnose root cause automatically
- Resolve or escalate with detailed diagnostics

Continuous Compliance Verification:

- Monitor code changes against IEC 62443 / CRA rules
- Flag violations before CI runs complete
- Audit trail for certification evidence

Future Directions

Summary

Near-term:

- Golang Port
- A2A integration for multi-agent workflows
- Agent-orchestrated TDD for security features
- Cross-project Agent Skill libraries

Long-term:

- Self-healing CI/CD pipelines
- Compliance verification as continuous process

Recommendations

Step-by-Step Approach

Based on implementation experience:

- Start with MCP/CLIs for tool integration
- Containerize agents early (Security)
- Log all agent actions
- Human oversight essential

Call to Action

- Say Hi 🖐️
- Connect on LinkedIn:
 - in/rainer-poisel

Addendum

References

Agent Circus: Agent Containers Generator

<https://codeberg.org/Embedded-Focus/agent-circus>

Model Context Protocol (MCP)

<https://modelcontextprotocol.io/>

Agent Client Protocol (ACP)

<https://agentclientprotocol.com/>

Agent Skills

<https://agentskills.io/>

Our Services

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

