Faysal Aziz

Low-Code Harness Engineering with Visual AI Frameworks

FA-0294 · two-day

Software Development

Introduction

Building Controlled Agent Workflows with Langflow and Python Extensions

Design, connect and verify AI-agent harnesses without building every component from scratch.

Harness Engineering does not always require a fully custom software platform. Visual and low-code frameworks can provide many of the components needed to assemble an agent harness, including model connections, prompts, tools, memory, workflow routing, API integration and execution monitoring. This allows engineering teams to concentrate on task design, operational controls and verification rather than spending most of their time developing infrastructure.

This two-day course focuses primarily on **Langflow**, an open-source, Python-based framework for visually building AI applications, agents and Model Context Protocol integrations. Its component-based editor supports rapid workflow construction, while custom Python components allow developers to extend the platform when built-in capabilities are insufficient.

Participants will construct a practical agent harness using visual components and a limited amount of Python. The resulting workflow will accept structured tasks, assemble context, provide controlled tools, maintain state, route failures and collect verification evidence. Python will be used only where custom validation, data transformation or specialised tool behaviour is required.

The course also introduces the capabilities and positioning of Flowise, Dify and n8n. Flowise provides visual agent workflows, evaluation, tracing and human-in-the-loop controls; Dify combines workflow nodes with agent capabilities and Python code execution; and n8n connects AI agents with broader business and system automation.

The scope is deliberately limited to a controlled single-agent harness. Complex multi-agent orchestration, distributed execution and production-scale platform administration are reserved for the advanced phase. The instructor has more than 30 years of industry experience and will use practical, industry-demanded approaches rather than treating the subject as a purely academic exercise.

Learning outcomes

By the end of this course, participants should be able to:

- Build a visual agent harness using Langflow
- Connect models, prompts, tools and memory components
- Create controlled workflow branches and approval points
- Develop simple custom components using Python
- Integrate APIs, repository information and MCP tools
- Apply execution boundaries and input validation
- Add verification and failure-handling workflows
- Compare suitable uses of Langflow, Flowise, Dify and n8n
- Package and expose a completed harness as an API

Prerequisites

- Completion of Harness Engineering Foundations or equivalent knowledge
- Basic understanding of AI agents and language models
- Basic familiarity with Python syntax
- General knowledge of APIs and JSON
- Familiarity with software testing and Git repositories
- Access to an approved language-model API or local model
- Docker knowledge is helpful but not mandatory

Training outline

1. Low-Code Harness Engineering Foundations

1. Visual workflow concepts
2. Components, nodes and connections
3. Deterministic versus agent-controlled routing
4. Low-code benefits and limitations
5. Selecting an appropriate framework

2. Langflow Environment Setup

1. Local or container-based installation
2. Workspace and project configuration
3. Model-provider connections
4. Credential management
5. Visual editor navigation
6. Playground and execution inspection

3. Building the Core Agent Flow

1. Task input
2. System instructions
3. Model component
4. Agent component
5. Tool connections
6. Structured output
7. Completion response

4. Visual Context Management

1. Static instructions
2. Repository documents
3. File ingestion
4. Text processing
5. Context filtering
6. Retrieval components
7. Sensitive-information exclusions

5. Tools and External Integrations

1. Built-in tools
2. API request components
3. Database and service connectors
4. MCP servers and tools
5. Tool descriptions
6. Tool input validation

7. Tool-result handling

6. Low-Code Python Extension

1. Custom component structure
2. Component inputs and outputs
3. Python data transformation
4. Custom validation logic
5. Reusable utility components
6. Error and exception handling

7. Harness Control and Safety

1. Input restrictions
2. Tool allowlists
3. Conditional routing
4. Iteration limits
5. Approval checkpoints
6. Failure branches
7. Restricted code execution

8. State and Observability

1. Flow state
2. Conversation memory
3. Task progress
4. Execution traces
5. Component outputs
6. Error records
7. Activity review

9. Verification Workflow

1. Acceptance criteria
2. Rule-based validation
3. Python validation components
4. Test-command integration

5. Output-quality checks
6. Failure-driven correction
7. Human approval

10. Practical Harness Assembly

1. Component integration
2. Controlled task execution
3. Tool-call inspection
4. Validation and correction
5. Failure diagnosis
6. Workflow refinement
7. Final harness assessment

11. Deployment and Framework Direction

1. Flow export and versioning
2. API exposure
3. Environment configuration
4. Langflow deployment considerations
5. Flowise Agentflow overview
6. Dify workflow overview
7. n8n automation integration
8. Preparation for advanced Harness Engineering

Disclaimer

This outline is provided as a professional instructional guideline for a two-day practical programme. The trainer reserves the right to amend, reorder, reduce, expand or substitute topics at their professional discretion to reflect participant ability, available infrastructure, organisational requirements, framework changes or instructional priorities. Such adjustments may be made without prior notice where reasonably necessary to maintain the relevance, safety and practical value of the training.

© 2026 Faysal Aziz · Kuala Lumpur