Responsible AI in agentic software development

With demonstrations

CYDRespAg | 1 day | On-site or online | Demonstrated labs

Generative AI is reshaping the software industry, moving beyond code suggestions into autonomous, agent-driven development. Tools like GitHub Copilot, Claude Code and other MCP-enabled agents can now participate in requirements gathering, design, testing, and deployment - not just code generation. This evolution sparks both excitement and caution: while productivity gains are clear, new risks around reliability, security, and bias demand responsible use.

This course first introduces participants to the foundations of Generative AI and Responsible AI and then explores how agentic GenAI is changing the software development lifecycle. Participants will study prompting techniques, context engineering, and the integration of AI into requirements specification, design, and testing. A major emphasis is placed on agentic workflows, including automated scaffolding, code-to-spec and spec-to-code transformations, and Dev(Sec)Ops integration via the Model Context Protocol.

Through numerous demonstrations and hands-on labs, participants will gain practical experience with opportunities and pitfalls: from improved productivity and testing support, to challenges such as hallucinations, dangers of "vibe coding", and the expanded attack surfaces in agentic systems.

By the end of the course, software engineers and managers will understand both the capabilities and the limitations of Generative AI and especially agentic GenAI, and will be equipped with skills to integrate these tools responsibly into modern software engineering practices.

A must-have primer for those looking to understand and responsibly adopt agentic GenAI in their software development projects. Building on these foundations, and depending on the technology stack, we suggest continuing with one of the Generative AI courses – see Agentic software development with generative AI in C++/Java/C#/Python. For those working on machine learning solutions, the comprehensive 4-day Machine Learning Security course offers a natural next step.

Skills and drills

12 LABS

2 CASE STUDIES

Audience

All people involved in using agentic AI tools in software development

Group size

12 participants

Preparedness

General development

Outline

- A brief history of Artificial Intelligence
- Responsible AI
- Using GenAI responsibly in software development
- Summary and takeaways

Standards and references

NIST

What you'll have learned

- Understand various aspects of responsible AI
- How to use generative AI responsibly in software development
- Prompt engineering for optimal outcomes
- How to apply generative AI throughout the SDLC
- The challenges in using agentic GenAI

Table of contents

Day 1

› A brief history of Artificial Intelligence

The origins of AI
Neural networks and "probability engines"

› Responsible AI

What is responsible AI?
Mitigation of harmful bias
Validity and reliability
🔬 *Demonstration – Experimenting with validity and reliability*
Explainability and interpretability
Security and responsible AI in software development

› Using GenAI responsibly in software development

LLM code generation basics
Prompt templating
Can AI... boost your productivity?
Can AI... produce better quality code?
Reviewing generated code – the black box blues
Some further long-term effects of using GenAI

Prompt engineering

- Why is a good prompt so important?
- Why can't we just leave prompt engineering to the agents?
- Establishing the context for generative AI

🔬 *Demonstration – Experimenting with prompts*

- Zero-shot, one-shot, and few-shot prompting
- Reasoning-based prompt engineering, chain-of-thought
- Prompt patterns
  - Prompt patterns and prompt priming

- Least-to-Most and Self-Planning: decomposition of complex tasks

Demonstration – Task decomposition

- Unit tests, TDD and GenAI

Demonstration – Test-based code generation

Integrating generative AI into the SDLC

- Using GenAI beyond code generation
- Using AI during requirements specification
- Prompt patterns for requirements capturing
- Prompt patterns for software design

Demonstration – Requirements capturing and API design

- Using AI during implementation
- Prompt patterns for implementation

Demonstration – Finding hidden assumptions

- Using AI during testing and QA
- Prompt patterns for refactoring

Demonstration – Experimenting with code refactoring

Agentic software development

- How is agentic coding different?
- The Model Context Protocol (MCP)
- Changes in MCP 2.0
- Capabilities of MCP agents
- Agentic integration in IDEs and non-MCP alternatives
- Agentic development workflow
  - Code-to-spec and spec-to-code with GenAI
  - Automated scaffolding

Demonstration – Agentic scaffolding

- Setting up the runtime environment

Demonstration – Environment setup

- Incremental development

Demonstration – Incremental development

- The role of MCP in Dev(Sec)Ops

Demonstration – Using MCP for DevOps

- Pitfalls and best practices
  - "Vibe coding" and its implications

Case study – Some cautionary tales of vibe coding and beyond

- Engineering concerns with MCP
- Security concerns of agentic development
- OWASP Top 10 for Agentic Applications (2026)

- MCP's effect on the attack surface
- MCP-specific attack vectors
- Agent skills and the attack surface
- *Demonstration – Attacking agentic GenAI*
- *Case study – Database leakage via Supabase MCP*
- Managing agentic workflows
- When is agentic coding NOT the right answer?
- Hallucinations and 'agentic death spirals'
- Token limits and context
- Context degradation with very large token counts
- Prompt engineering vs context engineering
- Context engineering from a developer's perspective
- Context document examples

- Agentic context patterns
  - What are agentic patterns?
  - Agentic patterns vs. prompt patterns – differences and parallels
  - The 8 categories of agentic patterns
  - A less formal approach: Willison's Agentic Engineering Patterns
  - Agentic pattern: Inversion of Control
  - Agentic pattern: Plan-Then-Execute
  - Agentic pattern: Tree-of-Thought Reasoning
  - Agentic pattern: Tool Use Steering via Prompting
  - Agentic pattern: Curated Code Context Window
  - Agentic pattern: Reflection Loop
  - Agentic pattern: Spec-As-Test Feedback Loop
  - Agentic pattern: Agent-Assisted Scaffolding
  - Agentic pattern: Structured Output Specification
  - Agentic pattern: Compounding Engineering
  - Agentic pattern: Deterministic Security Scanning Build Loop
  - Agentic pattern: Lethal Trifecta Threat Model

› Summary and takeaways

Responsible AI principles in software development

Generative AI – Resources and additional guidance