lab-06-custom-agents-intro

- Lab 06: Introduction to Custom Copilot Agents
  - Objectives
  - Prerequisites
  - Background
    - What Are Custom Copilot Agents?
    - Mental Model: The Specialist Analogy
    - How Agents Differ From...
    - When to Use Custom Agents
  - Exercise 1: Using the Architecture Reviewer Agent (15 minutes)
    - Setup
    - Scenario
    - Instructions
    - Expected Outcome
    - Reflection Questions
  - Exercise 2: Compare Agent vs. Standard Chat (10 minutes)
    - Instructions
    - Comparison Table
  - Exercise 3: Exploring Other Agents (5 minutes)
    - 1. Backlog Generator
    - 2. Test Strategist
    - Questions
  - Key Insights
    - If Copilot Instructions Are Guardrails...
    - Agents Are Products, Not Prompts
  - Key Takeaways
  - Common Questions
  - Next Steps
  - Additional Resources

Lab 06: Introduction to Custom Copilot Agents

Module: 2
Duration: 30 minutes
Part: Advanced GitHub Copilot (Part 2)

Objectives

By the end of this lab, you will:

- Understand what Custom Copilot Agents are and how they work
- Differentiate between agents, prompts, and Copilot Instructions
- Use a pre-built custom agent in Agent Mode
- Recognize when custom agents improve consistency over ad-hoc prompting

Prerequisites

- Completion of Lab 05: Interaction Models
- VS Code with GitHub Copilot extension
- Access to the TaskManager workshop repository

Background

What Are Custom Copilot Agents?

Custom Copilot Agents are **specialized AI assistants** that provide consistent, role-based guidance for specific workflows. Think of them as expert consultants you can invoke when needed.

Key Characteristics:

- **Named entities** - Selectable from the agent dropdown
- **Role-based personas** - Architecture reviewer, test strategist, backlog generator, etc.
- **Defined scope** - Clear responsibilities and constraints
- **Structured outputs** - Consistent format for results
- **Team-aligned** - Encode team practices and standards

Mental Model: The Specialist Analogy

```
Standard Copilot Chat = General AI Assistant
Custom Agent = Domain Expert Consultant

You wouldn't ask a general assistant to:
- Review architecture (you'd ask an architect)
- Plan testing strategy (you'd ask a QA specialist)
- Generate backlog items (you'd ask a product analyst)

Custom agents ARE those specialists.
```

How Agents Differ From...

Feature	Copilot Instructions	Ad-hoc Prompts	Custom Agents
Scope	Always active	One-off	Invoked on demand
Purpose	Global guardrails	Specific task	Repeatable workflow
Reusability	Implicit	Manual copy/paste	Built-in
Consistency	Background rules	Variable	Structured
Best for	Coding standards	Exploration	Workflow automation

When to Use Custom Agents

✅ **Use agents when:**

- You have **repeated workflows** (reviews, planning, analysis)
- You need **consistent outputs** across team members
- You want to **encode expert knowledge** in a reusable form
- You're performing **validation or review** tasks

❌ **Don't use agents when:**

- A simple prompt suffices
- You're exploring or learning
- The task is one-off or unique

Exercise 1: Using the Architecture Reviewer Agent (15 minutes)

Setup

The repository includes a pre-built **Architecture Reviewer** agent at `.github/agents/architecture-reviewer.agent.md`.

Scenario

You suspect there might be architectural issues in the Task entity or the repository implementation. You want an expert review.

Instructions

1. **Locate the agent:**
   - Navigate to `.github/agents/architecture-reviewer.agent.md`
   - Read the agent's responsibilities and constraints
   - Note the structured output format
2. **Open Copilot Chat** and switch to **Agent Mode**
3. **Select the Architecture Reviewer agent** from the dropdown
4. **Use this prompt:**

   ```
   Review the Task entity and TaskRepository for architectural compliance
   with Clean Architecture and DDD patterns.
   ```

5. **Observe the agent's behavior:**
   - Does it follow its defined structure?
   - Does it reference the layers (Domain, Application, Infrastructure, Api)?
   - Does it provide the expected sections (Strengths, Concerns, Violations, Recommendations)?
6. **Compare to standard Copilot Chat:**
   - Try the same prompt in regular Chat (without the agent)
   - Note the differences in depth, structure, and consistency

Expected Outcome

The Architecture Reviewer agent should:

- Analyze the code through a Clean Architecture lens
- Identify specific boundary violations (if any)
- Provide structured findings (✅ Strengths, ⚠️ Concerns, 🚫 Violations)
- Give actionable recommendations
- Reference project conventions (ADRs, DDD patterns)

Reflection Questions

1. **How did the agent's response differ from standard Chat?**
2. **Did the agent follow its defined output format?**
3. **Would this consistency be valuable for team code reviews?**
4. **What happens if you ask the agent something outside its scope?**

Exercise 2: Compare Agent vs. Standard Chat (10 minutes)

Instructions

Perform the same task twice:

Round 1: Standard Copilot Chat

1. Open Copilot Chat (no agent selected)
2. Prompt: `Review the Task domain model for DDD compliance`
3. Record the response structure and depth

Round 2: Architecture Reviewer Agent

1. Switch to Agent Mode
2. Select **Architecture Reviewer** from dropdown
3. Same prompt: `Review the Task domain model for DDD compliance`
4. Record the response structure and depth

Comparison Table

Fill in based on your observations:

Aspect	Standard Chat	Architecture Reviewer Agent
Response Format	[Your observation]	[Your observation]
Depth of Analysis	[Your observation]	[Your observation]
Consistency	[Your observation]	[Your observation]
Actionability	[Your observation]	[Your observation]
Repeatability	[Your observation]	[Your observation]

Exercise 3: Exploring Other Agents (5 minutes)

The repository includes three custom agents. Briefly explore each:

1. Backlog Generator

- **Location:** `.github/agents/backlog-generator.agent.md`
- **Try:** "Generate user stories for adding task comments feature"
- **Observe:** Structured user stories with acceptance criteria

2. Test Strategist

- **Location:** `.github/agents/test-strategist.agent.md`
- **Try:** "Propose test scenarios for Task creation"
- **Observe:** Categorized test scenarios (unit, integration, edge cases)

Questions

- Which agent would you use most frequently in your work?
- Can you think of other agents your team might need?

Key Insights

If Copilot Instructions Are Guardrails...

Copilot Instructions = Background rules always enforced
(e.g., "Use Clean Architecture, write tests first, follow DDD")

Custom Agents = Specialists you consult on demand
(e.g., "Architecture Reviewer, analyze this design")

Agents Are Products, Not Prompts

- Agents should be **versioned and reviewed** (like code)
- Agents **encode team knowledge** and standards
- Agents provide **repeatable, consistent outcomes**
- Agents improve **onboarding** (new team members use the same expert guidance)

Key Takeaways

✅ **Custom agents provide role-based expertise** on demand
✅ **Agents ensure consistency** across team members
✅ **Agents are reusable** - define once, use repeatedly
✅ **Agents complement Instructions** - not a replacement
⚠️ **Agents require maintenance** - treat them as team assets

---

Common Questions

Q: Can I use multiple agents in one session?
A: Yes! Switch agents as needed for different workflow steps.

Q: Do agents replace Copilot Instructions?
A: No. Instructions are always-on guardrails; agents are on-demand specialists.

Q: Can I create my own agent?
A: Absolutely! That's covered in Lab 08: Agent Design and Lab 09: Build Your Own.

Q: What if an agent gives incorrect advice?
A: Agents are assistants, not authorities. You're accountable for the final decision. Iterate on agent instructions to improve accuracy.

Next Steps

In Lab 07: Workflow Agents in Action, you'll apply these agents to real development workflows and compare their outputs to ad-hoc prompting.

Additional Resources

- Custom Agent Catalog
- Agent vs Instructions vs Prompts Diagram
- GitHub Documentation: Custom Agents