FACILITATOR_GUIDE_PART2

Facilitator's Guide: Advanced GitHub Copilot (Part 2)

Duration: 3 Hours
Target Audience: Developers who have completed Part 1 or have equivalent GitHub Copilot experience
Prerequisites: VS Code with GitHub Copilot extension, familiarity with custom instructions

---

Workshop Schedule at a Glance

Time	Module	Duration	Activity	Lab
0:00-0:10	**Module 0: Kickoff**	10 min	Welcome, Part 1 recap, Part 2 intro	-
0:10-0:35	**Module 1: Interaction Models**	25 min	Ask/Edit/Agent overview, live demo	Lab 05
0:35-1:05	**Module 2: Custom Agents**	30 min	What are agents? Architecture Reviewer demo	Lab 06
1:05-1:50	**Module 3: Workflow Agents**	45 min	Backlog, Architecture, Test Strategy workflows	Lab 07
1:50-2:00	**Break**	10 min	Rest and questions	-
2:00-2:25	**Module 4: Agent Design**	25 min	Design principles, iteration, patterns	Lab 08
2:25-3:00	**Module 5: Capstone**	35 min	Build your own production-ready agent	Lab 09
3:00-3:10	**Module 6: Wrap-Up**	10 min	Key takeaways, governance, next steps	-

Lab Summary

- **Lab 05: Interaction Models** (20 min) - Compare Ask, Edit, and Agent modes
- **Lab 06: Custom Agents Intro** (15 min) - Explore Architecture Reviewer, Backlog Generator, Test Strategist
- **Lab 07: Workflow Agents** (35 min) - Apply agents to 3 real workflows
- **Lab 08: Agent Design** (15 min) - Analyze agent components and iterate on instructions
- **Lab 09: Capstone** (30 min) - Design, build, test, and document your own agent

---

Overview

This guide helps facilitators deliver Part 2 of the AI Code Workshop, focusing on **advanced GitHub Copilot features** including interaction models (Ask/Edit/Agent) and **custom agents**. The workshop is highly interactive with 5 hands-on labs.

Learning Objectives

By the end of Part 2, participants will:

- Understand Ask, Edit, and Agent interaction models
- Use custom Copilot agents for specialized workflows
- Design and build their own production-ready agents
- Apply governance principles to AI tooling

Preparation Checklist

For Facilitators

- [ ] Review all 5 labs (05-09) and practice the exercises
- [ ] Test the 3 custom agents (Architecture Reviewer, Backlog Generator, Test Strategist)
- [ ] Ensure presentation slides render correctly in Marp
- [ ] Prepare live demo scenarios for each interaction model
- [ ] Have backup examples ready for agent design patterns
- [ ] Set up screen sharing for agent mode demonstrations

For Participants

- [ ] VS Code with GitHub Copilot extension (latest version)
- [ ] Access to workshop repository (`.github/agents/` directory)
- [ ] Verify agents appear in Copilot Chat agent dropdown
- [ ] Clone repository and be on correct branch

Environment Setup Verification

Before starting, confirm:

```
# Repository has custom agents
ls .github/agents/*.agent.md

# Expected output:
# architecture-reviewer.agent.md
# backlog-generator.agent.md
# test-strategist.agent.md
```

Critical: Custom agents must be in `.agent.md` format with YAML front-matter. Verify agents appear in VS Code Copilot Chat dropdown selector.

Module 0: Kickoff & Context Reset (0:00 – 0:10, 10 min)

Facilitator Actions

Welcome and Part 1 Recap (5 min)

- Quick poll: Who completed Part 1? Who's joining directly for Part 2?
- Recap Part 1 key concepts:
  - Using Copilot for TDD and refactoring
  - Copilot Instructions as guardrails
  - Documentation and requirements workflows

Part 2 Introduction (5 min)

- Present the learning journey (show slides 1-5)
- Emphasize the shift from **code generation** to **workflow orchestration**
- Set expectations: Part 2 is more conceptual with hands-on validation

Common Questions

Q: Do I need Part 1 to succeed in Part 2?
A: No, but you should be comfortable using GitHub Copilot for code generation and chat interactions.

Q: Will we write a lot of code?
A: Less code than Part 1. Focus is on designing agents and evaluating workflows.

Module 1: Copilot Interaction Models (0:10 – 0:35, 25 min)

Section Breakdown

- **0:10-0:15** - Presentation: Ask/Edit/Agent overview (slides 6-10)
- **0:15-0:25** - Live Demo: Same task, three ways
- **0:25-0:30** - Guided Exercise: Participants try each mode
- **0:30-0:35** - Discussion and Q&A

Facilitator Actions

Presentation (5 min)

- Use slides 6-10 to explain the three modes
- Key point: **Agent Mode is not "better chat"** – it's a different execution model
- Show the decision tree diagram (copilot-interaction-models.md)

Live Demo: Same Task, Three Ways (10 min)

Task: Add a `Priority` property to the Task entity

1. **Ask Mode Demo:**
    - Open Copilot Chat
    - Prompt: "How should I add a Priority property to the Task entity?"
    - Show: You get guidance, but no changes
2. **Edit Mode Demo:**
    - Open `src/TaskManager.Domain/Tasks/Task.cs`
    - Use inline chat (`Ctrl+I`): "Add a Priority property of type TaskPriority enum"
    - Show: Direct file modification
3. **Agent Mode Demo:**
    - Open Copilot Chat in Agent Mode
    - Prompt: "Add a Priority property to Task entity across all layers"
    - Show: Multi-step plan with checkpoints

Key Observation Points:

- Scope: Ask (informational), Edit (file-level), Agent (repository-level)
- Control: Ask (total), Edit (review diff), Agent (approve steps)
- Speed: Ask (instant), Edit (fast), Agent (deliberate)

Guided Exercise (5 min)

- Participants try all three modes with a simple task
- Circulate to help with agent mode activation
- Troubleshoot: If agents don't appear, check `.agent.md` files exist

Discussion Questions (5 min)

- When would you use Ask vs Edit?
- What scenarios require Agent mode?
- What are the risks of Agent mode?

Common Issues

Issue: Agent mode doesn't appear
Fix: Ensure Copilot extension is updated, restart VS Code, verify `.agent.md` files

Issue: Agent mode produces unexpected results
Fix: Agents are non-deterministic; emphasize human review of plans

---

Module 2: Custom Agents Intro (0:35 – 1:05, 30 min)

Section Breakdown

- **0:35-0:40** - Presentation: What are custom agents? (slides 11-15)
- **0:40-0:50** - Live Demo: Using Architecture Reviewer agent
- **0:50-1:00** - Guided Exercise: Lab 06 (hands-on)
- **1:00-1:05** - Debrief and Q&A

Facilitator Actions

Presentation (5 min)

- Use slides 11-15 to introduce custom agents
- Mental model: **Agents are specialists, not task executors**
- Show agents vs instructions vs prompts comparison table

Live Demo: Architecture Reviewer (10 min)

1. **Setup:**
   - Open Copilot Chat in Agent Mode
   - Select "Architecture Reviewer" from dropdown
   - Open `src/TaskManager.Infrastructure/Legacy/LegacyTaskProcessor.cs`
2. **Demo Prompt:**

   ```
   Review the LegacyTaskProcessor class for Clean Architecture violations
   ```

3. **Show the structured output:**
   - Architecture Review Summary
   - Layer Analysis
   - Violations Found
   - Recommendations
4. **Compare to standard chat:**
   - Use same prompt without agent
   - Highlight differences: structure, depth, consistency

Key Teaching Points:

- Agents provide **consistent, structured outputs**
- Agents encode **team knowledge** (Clean Architecture, DDD)
- Agents are **reusable** across team members

Guided Exercise: Lab 06 (10 min)

- Direct participants to Lab 06: Custom Agents Intro
- Participants explore all 3 agents:
  - Architecture Reviewer
  - Backlog Generator

    - Test Strategist
- Circulate to help with agent invocation

Debrief (5 min)

- Ask: Which agent was most useful? Why?
- Discuss: When would you NOT use an agent?
- Introduce concept of agent catalog

Common Issues

Issue: Agent output is too verbose
Teaching moment: This is by design for transparency; can be refined in agent instructions

Issue: Agent misses obvious issues
Teaching moment: Agents are first-pass tools; humans still review

---

Module 3: Workflow Agents in Action (1:05 – 1:50, 45 min)

Section Breakdown

- **1:05-1:10** - Presentation: Agent workflows (slides 16-20)
- **1:10-1:45** - Lab 07: Three real workflows
- **1:45-1:50** - Group discussion

Facilitator Actions

Presentation (5 min)

- Use slides 16-20 to set up the lab
- Emphasize: We'll **compare standard chat vs agents** for each workflow
- Preview the 3 workflows: Backlog, Architecture, Testing

Lab 07: Hands-On (35 min)

Participants work through Lab 07: Workflow Agents

Workflow 1: Backlog Generation (10 min)

- Scenario: Add notification system
- Try standard chat first, then Backlog Generator agent
- Compare outputs

Workflow 2: Architecture Review (10 min)

- Scenario: Review NotificationService design
- Try standard chat first, then Architecture Reviewer agent
- Compare outputs

Workflow 3: Test Strategy (10 min)

- Scenario: Plan tests for task assignment
- Try standard chat first, then Test Strategist agent

- Compare outputs

Facilitator Circulation Tips:

- Watch for participants skipping standard chat comparison
- Help participants articulate **why** structured output is better
- Encourage note-taking for reflection questions

Group Discussion (5 min)

- Which agent provided the most value?
- Did agents catch issues standard chat missed?
- What are the limitations?
- When would standard chat be better?

Teaching Moments

When agents shine:

- Repeatable workflows
- Need for consistency
- Encoding team standards

When agents struggle:

- Unique, one-off scenarios
- Highly creative exploration
- Context outside their expertise

Break (1:50 – 2:00, 10 min)

Facilitator Notes:

- Use this time to address 1-on-1 questions
- Check in with anyone struggling
- Prepare for Module 4 (agent design)

Module 4: Designing Effective Agents (2:00 – 2:25, 25 min)

Section Breakdown

- **2:00-2:05** - Presentation: Agent design principles (slides 21-26)
- **2:05-2:20** - Lab 08: Design patterns and iteration
- **2:20-2:25** - Key takeaways discussion

Facilitator Actions

Presentation (5 min)

- Use slides 21-26 to introduce design principles
- Core message: **Agents are products, not prompts**

- Show the 7 components of agent instructions

Critical Teaching Points:

1. **Role-based vs Task-based:**
   - ❌ Bad: "Generate unit tests"
   - ✅ Good: "Test Strategist who proposes test strategies"
2. **Explicit Constraints:**
   - Show ALWAYS/NEVER examples
   - Emphasize: What's unstated is undefined
3. **Structured Outputs:**
   - Consistency requires format
   - Show before/after examples

Lab 08: Hands-On (15 min)

Participants work through Lab 08: Agent Design

Exercise 1: Analyze Existing Agents (10 min)

- Participants dissect the 3 workshop agents
- Map components: Identity, Responsibilities, Constraints, etc.
- Fill in the analysis template

Exercise 2: Iteration Exercise (5 min)

- Modify Test Strategist to avoid over-testing
- Add constraint: "Focus on high-value tests only"
- Re-test and observe behavior change

Facilitator Tips:

- Emphasize: Iteration is expected and normal
- Show your own iteration examples
- Normalize "failing fast" with agents

Discussion (5 min)

- What makes a good agent?
- How many iterations before "production-ready"?
- Who should own agent governance?

Common Pitfalls to Address

1. **Task-based agents** → Guide toward role-based
2. **Vague instructions** → Demand specificity
3. **Over-scoping** → Encourage focus
4. **No testing** → Make testing a requirement
5. **Set-and-forget** → Normalize continuous iteration

Module 5: Capstone Lab (2:25 – 3:00, 35 min)

Section Breakdown

- **2:25-2:30** - Presentation: Capstone overview (slides 27-30)
- **2:30-2:55** - Lab 09: Build your own agent
- **2:55-3:00** - Group share (optional)

Facilitator Actions

Presentation (5 min)

- Use slides 27-30 to introduce the capstone
- Participants will **build a production-ready agent from scratch**
- Walk through the 6-step process

Lab 09: Build Your Own Agent (25 min)

Participants work through Lab 09: Capstone

Step 1: Select Role (5 min)

- Participants choose from 5 options or propose their own
- Encourage: Pick something you'll actually use

Step 2: Define Success Criteria (5 min)

- Write 5 success criteria
- Define 3 test scenarios
- Facilitator circulates to validate criteria

Step 3: Create Agent Definition (10 min)

- Use the template to build `.agent.md` file
- Save to `.github/agents/[name].agent.md`
- Facilitator helps with YAML front-matter syntax

Step 4: Test Agent (5 min)

- Run test scenarios
- Record results
- Identify issues

Step 5: Iterate (optional, if time)

- Refine based on test results
- Re-test

Step 6: Document (optional, if time)

- Create usage guide

Group Share (5 min, optional)

- If time allows, ask 2-3 participants to demo their agents

- Focus: What did you build? What surprised you?

Facilitator Tips

Time Management:

- Steps 5-6 are optional if time is tight
- Minimum deliverable: Steps 1-4 complete

Helping Struggling Participants:

- Guide toward simpler roles (Code Reviewer, Documentation Writer)
- Provide pre-built examples as scaffolds
- Pair participants if needed

Advanced Participants:

- Challenge: Build two complementary agents (Bonus Challenge)
- Encourage: Write full test scenarios (from `docs/requirements/agent-scenarios/`)

Module 6: Wrap-Up & Governance (3:00 – 3:10, 10 min)

Section Breakdown

- **3:00-3:03** - Key takeaways recap (slides 31-35)
- **3:03-3:07** - Governance discussion
- **3:07-3:10** - Next steps and closing

Facilitator Actions

Key Takeaways (3 min)

- Use slides 31-35 to recap the workshop
- Hit these points:
  - ✅ Ask/Edit/Agent - Use the right mode
  - ✅ Custom agents - Specialists, not prompts
  - ✅ Role-based design - Focus on WHO
  - ✅ Iterate continuously - Agents improve over time
  - ✅ Humans accountable - Agents assist, you decide

Governance Discussion (4 min)

- Ask: How should teams govern agents?
- Discuss:
  - Version control (git)
  - PR review process
  - Catalog maintenance
  - Deprecation strategy
- Reference: Agent Governance Guide

Next Steps (3 min)

- Share the adoption roadmap (slide 36)
  - Week 1: Use existing agents
  - Week 2: Draft your own agent
  - Week 3: Test and refine
  - Week 4: Share with team
- Point to resources: Catalog, Design Guide, Governance Guide
- Encourage: Open issues for feedback

Closing

- Thank participants
- Share feedback survey (if applicable)
- Open floor for final questions

Post-Workshop Follow-Up

For Facilitators

Immediately After:

- [ ] Collect feedback (survey or direct)
- [ ] Note what went well / what struggled
- [ ] Update this guide with lessons learned

Within 1 Week:

- [ ] Share workshop materials with participants
- [ ] Create a Slack/Teams channel for ongoing questions
- [ ] Schedule optional office hours for agent design help

Within 1 Month:

- [ ] Review agent adoption metrics (if tracking)
- [ ] Gather usage feedback on workshop agents
- [ ] Plan iteration on workshop content

For Participants

Immediately After:

- [ ] Review your capstone agent
- [ ] Identify 1-2 workflows to try agents with
- [ ] Share workshop experience with team

Within 1 Week:

- [ ] Use workshop agents in daily work
- [ ] Document one real scenario where agents helped
- [ ] Identify one workflow to automate with custom agent

Within 1 Month:

- [ ] Build and test your own production agent
- [ ] Share with team and gather feedback
- [ ] Contribute to team agent catalog

Troubleshooting Guide

Custom Agents Not Appearing

Symptoms: Agents don't show in Copilot Chat dropdown

Fixes:

1. Verify `.agent.md` files exist in `.github/agents/`
2. Check YAML front-matter syntax (name, description, tools, model)
3. Restart VS Code
4. Update GitHub Copilot extension to latest version
5. Verify workspace is a git repository

Agent Behavior Inconsistent

Symptoms: Same prompt yields different results

Explanation: This is expected! LLMs are non-deterministic.

Guidance:

- Emphasize structured outputs reduce variance
- Teach participants to refine prompts iteratively
- Normalize re-running agents for better results

Agent Goes Off-Track

Symptoms: Agent produces output outside its scope

Fixes:

- Add stronger constraints (NEVER clauses)
- Narrow the role definition
- Add examples of out-of-scope scenarios
- Refine output format to be more prescriptive

Participants Struggle with Agent Design

Symptoms: Blank stares during capstone lab

Facilitator Actions:

- Pair participants for collaborative design
- Provide scaffolded templates with partial completion
- Walk through one example agent design as a group
- Simplify: Start with Code Reviewer agent (most familiar)

Time Management Issues

If running behind:

- Skip Exercise 2 in Lab 08 (iteration exercise)
- Reduce capstone to Steps 1-3 only (no testing)
- Skip group share in Module 5

If running ahead:

- Add Bonus Challenge in Lab 09
- Facilitate deeper governance discussion
- Do live agent design as a group exercise

Appendix A: Quick Reference

Workshop Structure

Module	Time	Duration	Focus
0	0:00-0:10	10 min	Kickoff & Context
1	0:10-0:35	25 min	Interaction Models
2	0:35-1:05	30 min	Custom Agents Intro
3	1:05-1:50	45 min	Workflow Agents Lab

Module	Time	Duration	Focus
Break	1:50-2:00	10 min	Rest
4	2:00-2:25	25 min	Agent Design
5	2:25-3:00	35 min	Capstone Lab
6	3:00-3:10	10 min	Wrap-Up

Key Resources

- **Labs:** `docs/labs/lab-05-*.md` through `lab-09-*.md`
- **Presentation:** `docs/presentations/advanced-github-copilot.md`
- **Agents:** `.github/agents/*.agent.md`
- **Diagrams:** `docs/design/diagrams/`
- **Guides:** `docs/guides/custom-agent-catalog.md`, `agent-design-guide.md`, `agent-governance.md`
- **Test Scenarios:** `docs/requirements/agent-scenarios/`

Agent Template (Quick Reference)

```
---
name: "agent-name"
description: 'Brief description'
tools: [changes]
model: Claude Sonnet 4
---

# Agent Name

You are an expert [role].

## Responsibilities
## Context
## Constraints
## Analysis Process
## Output Format
## Tone
```

Appendix B: Facilitator Self-Assessment

After delivering the workshop, reflect on these questions:

Content Delivery:

- [ ] Did I balance presentation vs hands-on time well?
- [ ] Were participants engaged throughout?
- [ ] Did I provide enough real-world examples?

Lab Facilitation:

- [ ] Did participants complete all labs?
- [ ] Were instructions clear?
- [ ] Did I circulate enough to help struggling participants?

Agent Demos:

- [ ] Were my agent demos clear and compelling?
- [ ] Did I show agent iteration effectively?
- [ ] Did I emphasize human accountability?

Governance Discussion:

- [ ] Did participants understand governance importance?
- [ ] Did we discuss team vs org-level standardization?
- [ ] Did I set clear next steps?

Overall:

- [ ] What went better than expected?
- [ ] What would I change for next time?
- [ ] What feedback did I receive?

Appendix C: Frequently Asked Questions

Before the Workshop

Q: Do participants need to complete Part 1 first?
A: No, but they should be comfortable with GitHub Copilot for code generation and chat.

Q: What if participants don't have Copilot access?
A: They can't fully participate. Ensure Copilot licenses are provisioned in advance.

Q: How technical is Part 2 compared to Part 1?
A: Less coding, more conceptual. Focus is on design, workflows, and governance.

During the Workshop

Q: Why use custom agents instead of just better prompts?
A: Agents provide consistency, encode team knowledge, and are reusable across the team.

Q: Can agents execute code changes automatically?
A: Agent mode requires human approval at checkpoints. Never blindly trust agent output.

Q: How do we prevent "prompt sprawl" with many agents?
A: Maintain a catalog, enforce review process, deprecate unused agents.

Q: Should agents be org-wide or team-specific?
A: Both. Architecture/Security agents → org-wide. Workflow agents → team-specific.

After the Workshop

Q: How do I measure agent adoption success?
A: Track: agent usage frequency, time saved, consistency improvements, team feedback.

Q: What if my custom agent doesn't work well?
A: Normal! Iterate on instructions, test with more scenarios, gather team feedback.

Q: Can I share my custom agents publicly?
A: Yes, but ensure they don't contain proprietary knowledge or security-sensitive rules.

Appendix D: Customization Notes

This workshop is designed to be customizable. Consider these adaptations:

For Different Durations

2-Hour Version (Condensed):

- Skip Module 0 (assume Part 1 completion)
- Combine Modules 1 & 2 (20 min)
- Shorten Module 3 (30 min)
- Skip Module 4 (agent design theory)
- Focus on Module 5 (capstone only)

4-Hour Version (Extended):

- Add deeper governance discussion (20 min)
- Include group agent design exercise (30 min)
- Add agent testing workshop (20 min)
- Include agent scenario walk-throughs (20 min)

For Different Audiences

Senior Engineers:

- Emphasize governance and architecture
- Challenge: Build complex, multi-agent workflows

- Focus: Encoding expert knowledge in agents

Junior Engineers:

- Emphasize learning from agent outputs
- Provide more scaffolding in capstone
- Focus: Using agents to accelerate learning

Engineering Managers:

- Emphasize team adoption and metrics
- Focus: Governance, standardization, ROI
- Challenge: Design org-level agent strategy

For Different Tech Stacks

Python:

- Replace .NET examples with Python equivalents
- Use pytest for testing scenarios
- Adapt architecture constraints (FastAPI, Flask, etc.)

Java:

- Replace .NET examples with Spring Boot
- Use JUnit for testing scenarios
- Adapt architecture constraints (Spring layers)

JavaScript/TypeScript:

- Replace .NET examples with Node.js/Express
- Use Jest for testing scenarios
- Adapt architecture constraints (NestJS, Next.js, etc.)

End of Facilitator's Guide - Part 2

For questions or suggestions, open an issue in the workshop repository.