

# agent-vs-instructions-vs-prompts

- [Agents vs Instructions vs Prompts](#)
  - [Conceptual Overview](#)
  - [Decision Tree: Which Should I Use?](#)
  - [Comparison Table](#)
  - [Layered Architecture](#)
  - [Use Case Matrix](#)
  - [Prompts in Detail](#)
    - [Characteristics](#)
    - [When to Use](#)
    - [Example Prompts](#)
  - [Copilot Instructions in Detail](#)
    - [Characteristics](#)
    - [When to Use](#)
    - [Example Instructions](#)
  - [Custom Agents in Detail](#)
    - [Characteristics](#)
    - [When to Use](#)
    - [Agent Structure](#)
  - [Migration Path](#)
    - [Step-by-Step](#)
  - [When to Combine Approaches](#)
  - [Anti-Patterns](#)
    - [✗ Using Agent for Simple Questions](#)
    - [✗ Putting Agent Logic in Instructions](#)
    - [✗ Over-Engineering Prompts](#)
  - [Feature Comparison](#)
  - [Governance Considerations](#)
  - [Quick Reference Card](#)
    - [Choose Prompts When:](#)
    - [Choose Copilot Instructions When:](#)
    - [Choose Custom Agents When:](#)
  - [See Also](#)

## Agents vs Instructions vs Prompts

This diagram helps you understand the differences between custom agents, Copilot instructions, and prompt engineering, and when to use each approach.

### Conceptual Overview

graph TB

```
AI[GitHub Copilot AI] --> Three{Three Ways to<br/>Influence Behavior}
```

```
Three --> Prompts[Prompts<br/>Ad-hoc requests]
```

```
Three --> Instructions[Copilot Instructions<br/>Global defaults]
```

```
Three --> Agents[Custom Agents<br/>Specialized roles]
```

```
Prompts --> P1[Per-request]
```

```

Prompts --> P2[Conversational]
Prompts --> P3[No persistence]

Instructions --> I1[Always active]
Instructions --> I2[Repository-wide]
Instructions --> I3[Team standards]

Agents --> A1[On-demand]
Agents --> A2[Role-specific]
Agents --> A3[Structured output]

style Prompts fill:#e1f5e1
style Instructions fill:#fff4e1
style Agents fill:#4a90e2,color:#fff

```

## Decision Tree: Which Should I Use?

```

graph TD
    Start{What are you<br/>trying to achieve?}

    Start -->|Set coding standards| Standards{For this repo<br/>or globally?}
    Start -->|Get help once| Once{Structured output<br/>needed?}
    Start -->|Repeated specialized task| Repeated[Custom Agent]

    Standards -->|This repo| RepoInstructions[Copilot Instructions<br/>.github/copilot-instructions.md]
    Standards -->|All repos| GlobalInstructions[Global Copilot Instructions<br/>VS Code Settings]

    Once -->|Yes| ConsiderAgent[Consider Agent<br/>if task repeats]
    Once -->|No| SimplePrompt[Simple Prompt]

    ConsiderAgent -->|Will repeat| CreateAgent[Create Custom Agent]
    ConsiderAgent -->|One-time| DetailedPrompt[Detailed Prompt]

    style Start fill:#4a90e2,color:#fff
    style Repeated fill:#90ee90
    style CreateAgent fill:#90ee90
    style RepoInstructions fill:#ffd700
    style SimplePrompt fill:#87ceeb

```

## Comparison Table

Aspect	Prompts	Copilot Instructions	Custom Agents
Scope	Single conversation	Repository-wide	Task-specific
Persistence	None	Always active	On-demand
Location	Chat input	.github/copilot-instructions.md	.github/agents/*.agent.md
Best For	Ad-hoc questions	Team coding standards	Specialized workflows
Activation	Every message	Automatic	Manual selection
Learning Curve	Low	Medium	Medium-High

Aspect	Prompts	Copilot Instructions	Custom Agents
Reusability	Copy-paste	Automatic	Select from dropdown
Team Sharing	Manual	Via git	Via git
Specificity	Variable	Broad	Highly specific
Output Format	Conversational	Follows standards	Structured

## Layered Architecture

```
graph TB
    subgraph "Execution Stack"
        Request[User Request]
        Agent[Agent Instructions<br/>if agent selected]
        Instructions[Copilot Instructions<br/>.github/copilot-instructions.md]
        Global[Global Instructions<br/>VS Code Settings]
        Base[Base Copilot Behavior]
    end

    Request --> Agent
    Agent --> Instructions
    Instructions --> Global
    Global --> Base
    Base --> LLM[Language Model]
    LLM --> Response[Response]

    style Request fill:#e1f5e1
    style Agent fill:#4a90e2,color:#fff
    style Instructions fill:#ffd700
    style Global fill:#fff4e1
    style Base fill:#f0f0f0

    Note1[Most Specific] -.-> Request
    Note2[Most General] -.-> Base
```

## Use Case Matrix

```
graph TB
    subgraph "Quick Tasks"
        Q1["'Explain this code'"] --> Prompt1[Simple Prompt]
        Q2["'Fix this bug'"] --> Prompt2[Simple Prompt]
        Q3["'Suggest variable name'"] --> Prompt3[Simple Prompt]
    end

    subgraph "Team Standards"
        S1[Use Clean Architecture] --> CI1[Copilot Instructions]
        S2[Follow naming conventions] --> CI2[Copilot Instructions]
        S3[TDD by default] --> CI3[Copilot Instructions]
    end

    subgraph "Specialized Workflows"
        W1[Generate user stories] --> Agent1[Backlog Generator]
        W2[Review architecture] --> Agent2[Architecture Reviewer]
        W3[Plan test strategy] --> Agent3[Test Strategist]
    end
```

```
style Prompt1 fill:#87ceeb
style Prompt2 fill:#87ceeb
style Prompt3 fill:#87ceeb
style CI1 fill:#ffd700
style CI2 fill:#ffd700
style CI3 fill:#ffd700
style Agent1 fill:#90ee90
style Agent2 fill:#90ee90
style Agent3 fill:#90ee90
```

## Prompts in Detail

### Characteristics

- **Ephemeral:** Only applies to current conversation
- **Flexible:** Can be anything
- **Context-dependent:** Relies on chat context
- **Learning tool:** Good for exploring capabilities

### When to Use

```
graph LR
    Use[Use Prompts] --> Quick[Quick questions]
    Use --> Explore[Exploring ideas]
    Use --> OneOff[One-off requests]
    Use --> Learning[Learning Copilot]

    style Use fill:#87ceeb
```

### Example Prompts

- "Explain how this authentication flow works"
- "What's the time complexity of this algorithm?"
- "Suggest improvements to this function"
- "Help me debug this error message"

---

## Copilot Instructions in Detail

### Characteristics

- **Persistent:** Applied to all Copilot interactions
- **Scoped:** Repository-level or global
- **Declarative:** States how code should be written
- **Standard:** Enforces team conventions

### When to Use

```
graph LR
    Use[Use Instructions] --> Standards[Coding standards]
    Use --> Conventions[Naming conventions]
    Use --> Arch[Architectural patterns]
```

```
Use --> Style[Code style preferences]
```

```
style Use fill:#ffd700
```

## Example Instructions

```
# .github/copilot-instructions.md
```

```
## Architecture
```

- Use Clean Architecture layers
- Domain has no dependencies
- Prefer dependency injection

```
## Naming
```

- PascalCase for types
- camelCase for variables
- Use descriptive names

```
## Testing
```

- TDD approach preferred
  - Use xUnit and FakeItEasy
  - One test class per method
- 

## Custom Agents in Detail

### Characteristics

- **Role-based:** Takes on specific persona
- **Structured:** Consistent output format
- **Reusable:** Select when needed
- **Specialized:** Expert in narrow domain

### When to Use

```
graph LR
    Use[Use Agents] --> Repeat[Repeated workflows]
    Use --> Structure[Structured output needed]
    Use --> Expert[Need domain expertise]
    Use --> Review[Specialized reviews]

    style Use fill:#90ee90
```

### Agent Structure

```
---
name: architecture-reviewer
description: Reviews code for Clean Architecture and DDD compliance
tools: ["read", "list_files"]
model: gpt-4o
---
```

## # Identity

You are an expert software architect...

## # Responsibilities

- Review code against Clean Architecture
- Identify dependency violations

...

---

# Migration Path

graph LR

Start[Start with Prompts] --> Learn[Learn patterns]

Learn --> Document{Repeating<br/>patterns?}

Document -->|Team-wide| Instructions[Move to<br/>Copilot Instructions]

Document -->|Specialized| Agent[Create<br/>Custom Agent]

Instructions --> Monitor[Monitor usage]

Agent --> Monitor

Monitor --> Refine[Refine over time]

style Start fill:#87ceeb

style Instructions fill:#ffd700

style Agent fill:#90ee90

## Step-by-Step

1. **Week 1-2:** Use prompts, note what you ask repeatedly
2. **Week 3-4:** Add common patterns to Copilot Instructions
3. **Week 5+:** Create agents for specialized workflows

---

# When to Combine Approaches

graph TB

Scenario[Feature Development] --> CI[Copilot Instructions<br/>Apply coding standards]

CI --> Agent[Custom Agent<br/>Generate user stories]

Agent --> Prompt[Prompts<br/>Ask clarifying questions]

Prompt --> Edit[Edit Mode<br/>Implement with standards]

Edit --> AgentReview[Custom Agent<br/>Architecture review]

style CI fill:#ffd700

style Agent fill:#90ee90

style Prompt fill:#87ceeb

style AgentReview fill:#90ee90

## Example workflow:

1. Copilot Instructions ensure Clean Architecture

- 2. Backlog Generator creates user stories
- 3. Prompts clarify acceptance criteria
- 4. Edit mode implements with standards applied
- 5. Architecture Reviewer validates approach

---

## Anti-Patterns

### ✗ Using Agent for Simple Questions

```
graph LR
    Q["What does this do?"] -->|Wrong| A[Architecture Reviewer Agent]
    Q -->|Right| P[Simple Prompt]

    style A fill:#ffcccc
    style P fill:#90ee90
```

### ✗ Putting Agent Logic in Instructions

```
graph LR
    Need[Generate user stories] -->|Wrong| I[Copilot Instructions]
    Need -->|Right| A[Backlog Generator Agent]

    style I fill:#ffcccc
    style A fill:#90ee90
```

### ✗ Over-Engineering Prompts

```
graph LR
    Complex[Complex repeated prompt] -->|Wrong| Copy[Copy-paste every time]
    Complex -->|Right| Agent[Create Agent]

    style Copy fill:#ffcccc
    style Agent fill:#90ee90
```

---

## Feature Comparison

Feature	Prompts	Instructions	Agents
Version Control	✗	✓	✓
Team Sharing	Manual	Automatic	Automatic
Discoverability	✗	Limited	High
Context Aware	Session only	Always	When invoked
Structured Output	✗	✗	✓
Learning Curve	None	Low	Medium
Maintenance	N/A	Medium	Low
Testability	✗	Limited	✓

---

# Governance Considerations

```
graph TB
    Level[Governance Level]

    Level --> Individual[Individual Developer]
    Level --> Team[Team/Repository]
    Level --> Org[Organization]

    Individual --> IPrompts[Personal Prompts]
    Individual --> IGlobal[Global Instructions<br/>VS Code Settings]

    Team --> TInstructions[Copilot Instructions<br/>.github/copilot-instructions.md]
    Team --> TAgents[Custom Agents<br/>.github/agents/]





    Org --> OTemplates[Repository Templates<br/>with Instructions]
    Org --> OAgentLib[Shared Agent Library]

    style Individual fill:#87ceeb
    style Team fill:#ffd700
    style Org fill:#90ee90
```





---

## Quick Reference Card





### Choose Prompts When:

-  One-time question
-  Exploring capabilities
-  Context-specific help
-  Learning something new

### Choose Copilot Instructions When:

-  Team coding standards
-  Always-on behavior
-  Architectural patterns
-  Consistent code style

### Choose Custom Agents When:

-  Repeated specialized tasks
  -  Structured output needed
  -  Role-based assistance
  -  Complex workflows
- 

## See Also

- [Lab 05: Copilot Interaction Models](#)
- [Lab 06: Introduction to Custom Agents](#)

- [Copilot Interaction Models Diagram](#)
- [Agent Architecture Diagram](#)
- [Custom Agent Catalog](#)