

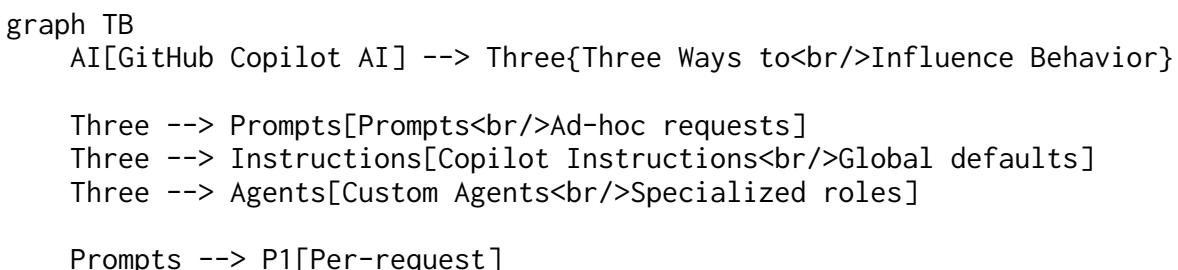
# agent-vs-instructions-vs-prompts

- [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



```

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

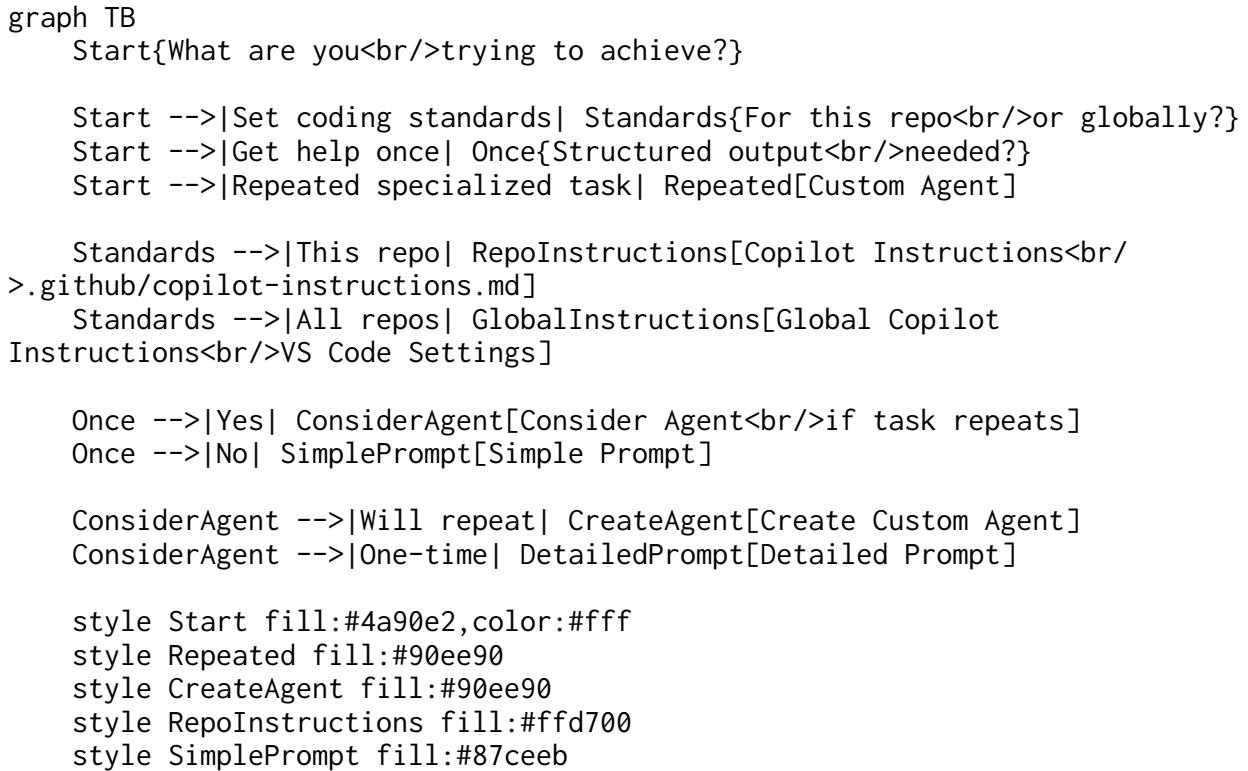
Instructions --> I1[Always active]
Instructions --> I2[Repository-wide]
Instructions --> I3[Team standards]

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?



## Comparison Table

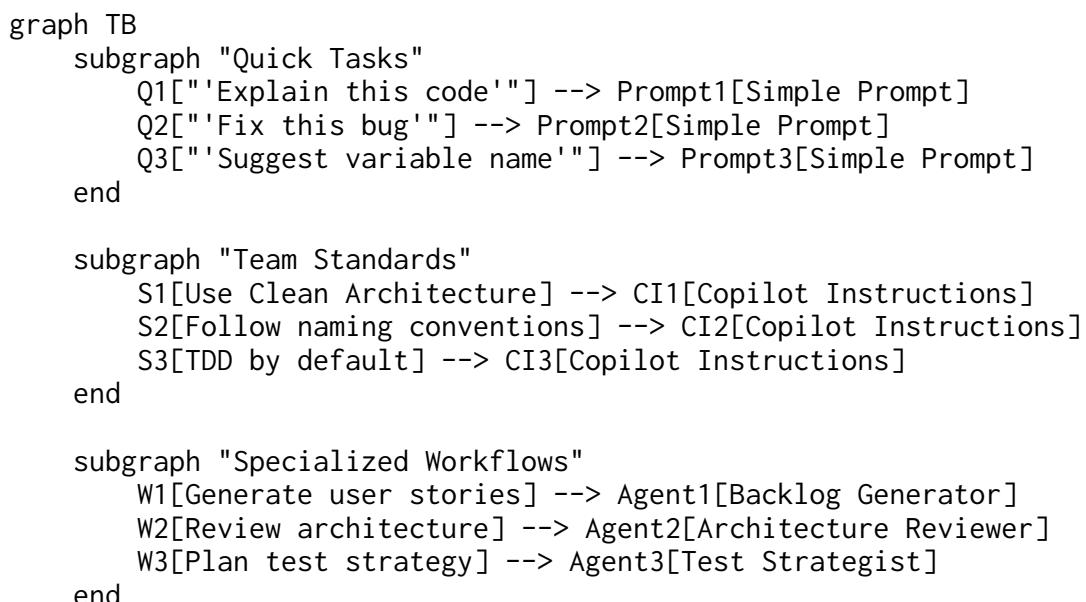
Aspect	Prompts	Copilot Instructions	Custom Agents
Scope	Single conversation	Repository-wide	Task-specific
Persistence	None	Always active	On-demand
Location	Chat input	.github/copilot-instructions.md	.github/agents/*.agent.md
Best For	Ad-hoc questions	Team coding standards	Specialized workflows
Activation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

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

## Layered Architecture



## Use Case Matrix



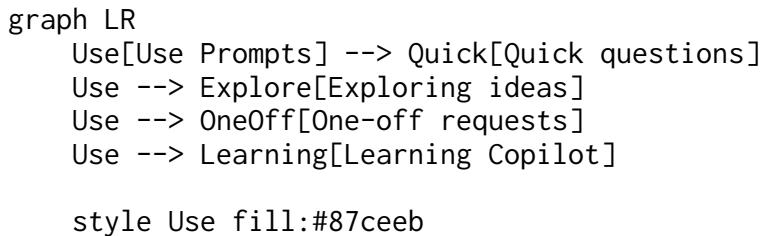
```
style Prompt1 fill:#87ceeb  
style Prompt2 fill:#87ceeb  
style Prompt3 fill:#87ceeb  
style CI1 fill:#ffd700  
style CI2 fill:#ffd700  
style CI3 fill:#ffd700  
style Agent1 fill:#90ee90  
style Agent2 fill:#90ee90  
style Agent3 fill:#90ee90
```

## 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



### 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"

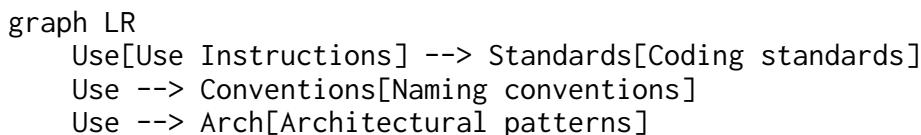
---

## 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



```
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
```

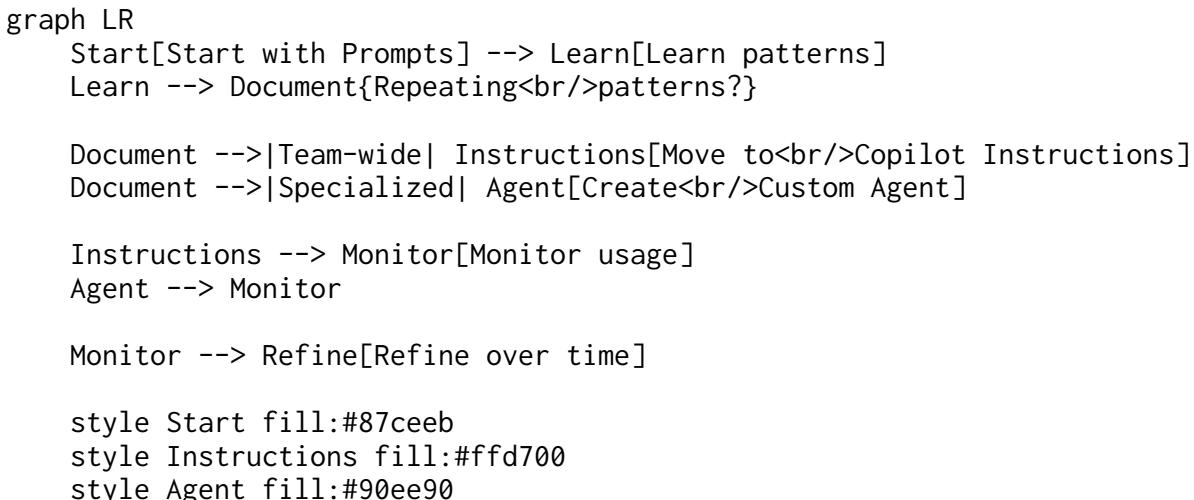
```
---
```

```
# Identity  
You are an expert software architect...
```

```
# Responsibilities  
- Review code against Clean Architecture  
- Identify dependency violations  
...
```

---

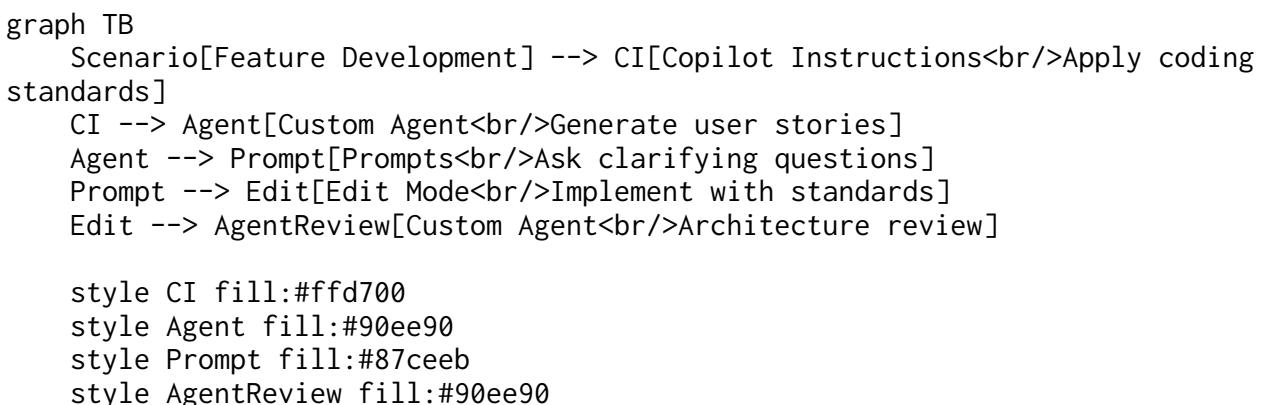
## Migration Path



### 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



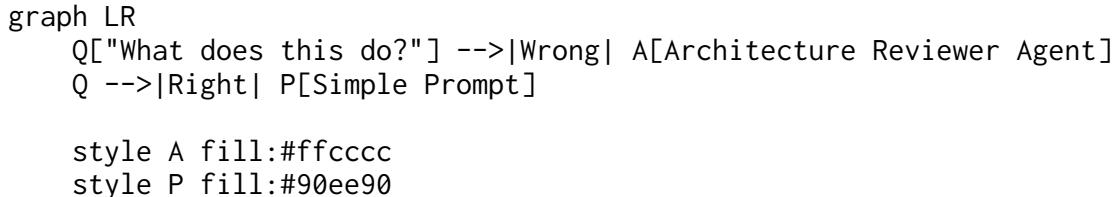
### Example workflow:

1. Copilot Instructions ensure Clean Architecture

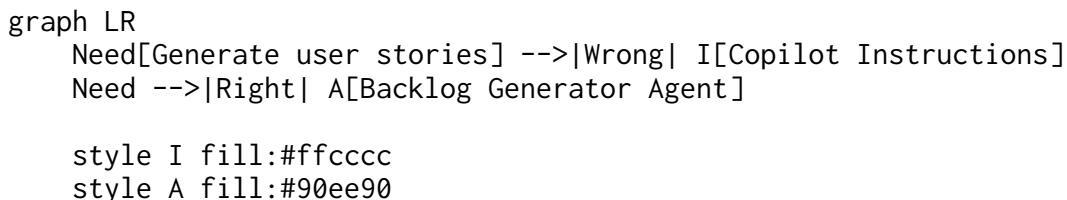
2. Backlog Generator creates user stories
  3. Prompts clarify acceptance criteria
  4. Edit mode implements with standards applied
  5. Architecture Reviewer validates approach
- 

## Anti-Patterns

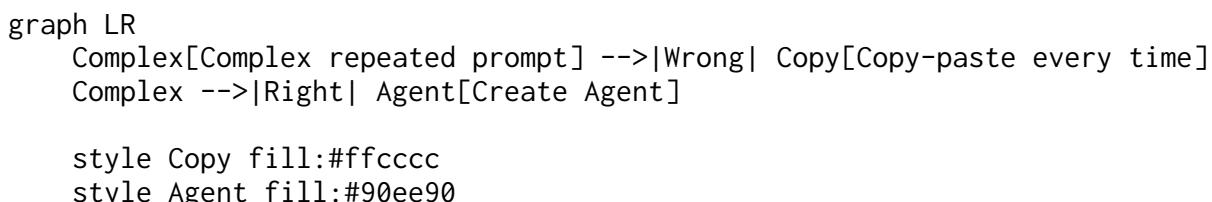
### ✖ Using Agent for Simple Questions



### ✖ Putting Agent Logic in Instructions



### ✖ Over-Engineering Prompts

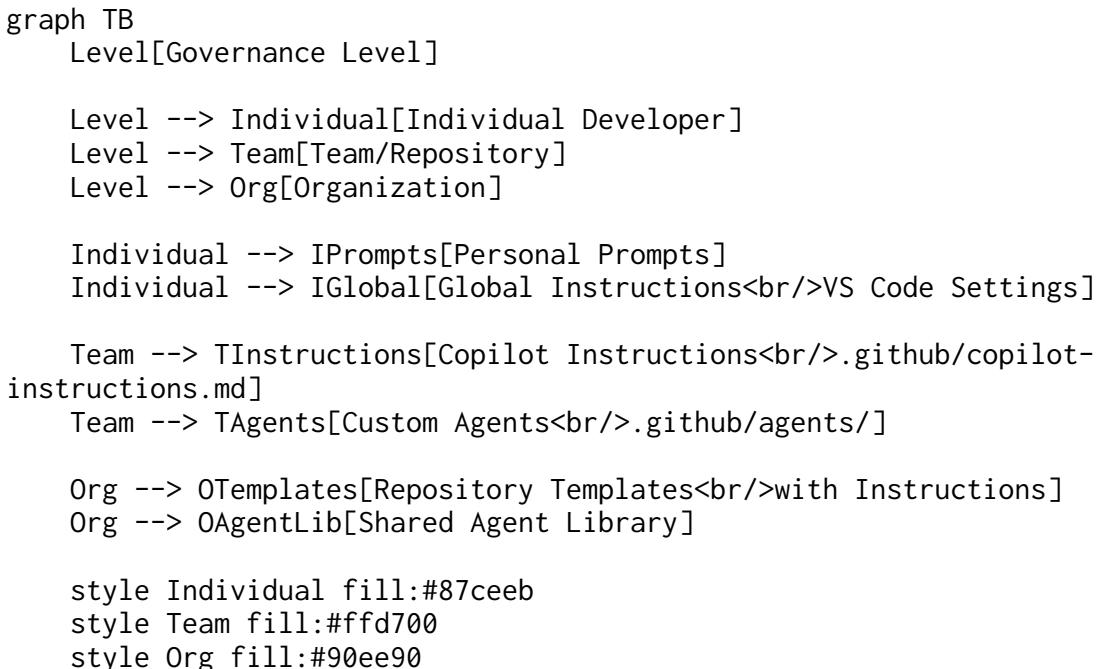


## Feature Comparison

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

---

# Governance Considerations



## 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](#)