agent-workflow-patterns

Agent Workflow Patterns

This diagram illustrates common patterns for using custom agents in development workflows.

Pattern 1: Sequential Agent Chain

```
graph LR
    Start[Feature Request] --> BG[Backlog Generator]
    BG --> Stories[User Stories]
    Stories --> AR[Architecture Reviewer]
    AR --> Design[Design Feedback]
    Design --> TS[Test Strategist]
    TS --> TestPlan[Test Plan]
    TestPlan --> Impl[Implementation]

    style BG fill:#e1f5e1
    style AR fill:#fff4e1
    style TS fill:#e1e8ff
    style Impl fill:#d4edda
```

When to use:

- Each agent output informs the next step
- Building comprehensive documentation
- Progressive refinement of a concept

Example scenario:

1. Generate user stories from requirements
2. Review stories for architectural implications

3. Create test strategy based on design decisions

Pattern 2: Parallel Review

```
graph TB
    Code[Code Changes] --> Review{Multi-Agent<br/>Review}

    Review --> AR[Architecture Reviewer]
    Review --> SR[Security Reviewer]
    Review --> PR[Performance Reviewer]

    AR --> AF[Architecture<br/>Feedback]
    SR --> SF[Security<br/>Feedback]
    PR --> PF[Performance<br/>Feedback]

    AF --> Consolidate[Consolidate<br/>Feedback]
    SF --> Consolidate
    PF --> Consolidate

    Consolidate --> Action[Action Items]

    style Review fill:#4a90e2,color:#fff
    style Consolidate fill:#f0f0f0
```

When to use:

- Comprehensive code reviews
- Multiple perspectives needed simultaneously
- Different quality dimensions to evaluate

Example scenario:

- Before merging a PR, run architecture, security, and performance reviews in parallel
- Gather all feedback before making changes

Pattern 3: Iterative Refinement

```
graph TB
    Initial[Initial Draft] --> Agent[Custom Agent]
    Agent --> Output1[First Output]
    Output1 --> Eval{Good Enough?}

    Eval -->|No| Refine[Refine Prompt]
    Refine --> Agent

    Eval -->|Yes| Final[Final Output]

    style Agent fill:#4a90e2,color:#fff
    style Eval fill:#ffd700
```

When to use:

- Complex outputs requiring multiple iterations

- Learning agent behavior and improving prompts
- Fine-tuning agent instructions

Example scenario:

- Generate user stories, review quality, refine acceptance criteria
- Iterate until stories meet INVEST criteria

Pattern 4: Agent-Assisted Development Cycle

```
sequenceDiagram
    participant Dev as Developer
    participant BG as Backlog Generator
    participant AR as Architecture Reviewer
    participant Code as Copilot Edit
    participant TS as Test Strategist
    participant Test as Test Implementation

    Dev->>BG: Describe feature
    BG->>Dev: User stories

    Dev->>AR: Review design approach
    AR->>Dev: Architecture guidance

    Dev->>Code: Implement with Edit mode
    Code->>Dev: Code changes

    Dev->>TS: Request test strategy
    TS->>Dev: Test scenarios

    Dev->>Test: Generate tests
    Test->>Dev: Test code

    Dev->>Dev: Run tests & commit
```

When to use:

- Full feature development workflow
- Combining agents with other Copilot modes
- End-to-end guided development

Example scenario:

- Use agents for planning and strategy
- Use Edit mode for implementation
- Use Ask mode for quick questions

Pattern 5: Feedback Loop with Validation

```
graph TB
    Input[User Input] --> Agent[Custom Agent]
    Agent --> Draft[Draft Output]
    Draft --> Validate{Meets<br/>Criteria?}
```

```
    Validate -->|No - Format| Format[Adjust Output Format<br/>in Agent
Definition]
    Validate -->|No - Content| Content[Refine Instructions<br/>or Context]

    Format --> Agent
    Content --> Agent

    Validate -->|Yes| Accept[Accept Output]
    Accept --> Document[Document Pattern]

    style Agent fill:#4a90e2,color:#fff
    style Validate fill:#ffd700
    style Document fill:#d4edda
```

When to use:

- Establishing new agents
- Quality assurance for agent outputs
- Building agent definition library

Example scenario:

- Create new agent, test with real scenarios
- Refine instructions until output is consistent
- Document successful patterns

Pattern 6: Context-Rich Agent Invocation

```
graph LR
    subgraph "Preparation"
        Files[Gather Files]
        Docs[Review Docs]
        History[Check History]
    end

    subgraph "Invocation"
        Context[Build Context]
        Agent[Invoke Agent]
        Prompt[Specific Prompt]
    end

    subgraph "Execution"
        Process[Agent Processes]
        Generate[Generate Output]
        Validate[Self-Check]
    end

    Files --> Context
    Docs --> Context
    History --> Context

    Context --> Agent
    Prompt --> Agent
```

```
    Agent --> Process
    Process --> Generate
    Generate --> Validate
    Validate --> Output[Structured Output]

    style Agent fill:#4a90e2,color:#fff
```

When to use:

- Agent needs extensive project context
- Specialized domain knowledge required
- Complex analysis or decisions

Example scenario:

- Before architecture review, open relevant design docs
- Include ADRs and architecture diagrams in conversation
- Agent has full context for informed feedback

Pattern 7: Progressive Disclosure

```
graph TB
    Start[Start Simple] --> Agent1[Agent: Basic Query]
    Agent1 --> Review1{Sufficient?}

    Review1 -->|Yes| Done[Complete]
    Review1 -->|No| Details[Add Details]

    Details --> Agent2[Agent: Detailed Query]
    Agent2 --> Review2{Sufficient?}

    Review2 -->|Yes| Done
    Review2 -->|No| Deep[Deep Dive]

    Deep --> Agent3[Agent: Specific Focus]
    Agent3 --> Done

    style Agent1 fill:#e1f5e1
    style Agent2 fill:#fff4e1
    style Agent3 fill:#e1e8ff
```

When to use:

- Exploring unknown territory
- Learning agent capabilities
- Avoiding information overload

Example scenario:

1. Start with high-level architecture review
2. Drill into specific concerns identified
3. Deep dive on critical issues

Pattern 8: Agent Handoff

```
graph LR
    Dev1[Developer A] --> Agent1[Backlog Generator]
    Agent1 --> Stories[User Stories]
    Stories --> Commit1[Commit Stories]

    Commit1 --> Dev2[Developer B]
    Dev2 --> Agent2[Architecture Reviewer]
    Agent2 --> Design[Design Feedback]
    Design --> Commit2[Commit Design]

    Commit2 --> Dev3[Developer C]
    Dev3 --> Agent3[Test Strategist]
    Agent3 --> Tests[Test Plan]

    style Stories fill:#e1f5e1
    style Design fill:#fff4e1
    style Tests fill:#e1e8ff
```

When to use:

- Team collaboration on feature
- Asynchronous workflows
- Different expertise areas

Example scenario:

- Product owner generates stories with agent
- Architect reviews with agent, commits feedback
- QA engineer creates test plan with agent

Pattern 9: Multi-Turn Conversation

```
sequenceDiagram
    participant U as User
    participant A as Agent

    U->>A: Initial request
    A->>U: Clarifying questions
    U->>A: Provide details
    A->>U: Partial output + questions
    U->>A: Confirm approach
    A->>U: Refined output
    U->>A: Request specific section
    A->>U: Detailed section
    U->>A: Looks good, finalize
    A->>U: Complete final output
```

When to use:

- Complex requirements needing clarification
- Incremental understanding building
- Collaborative refinement

Example scenario:

- Start with vague feature idea
- Agent asks clarifying questions
- Iteratively build complete user stories

Pattern 10: Agent as Second Reviewer

```
graph TB
    Code[Code Written] --> Human[Human Review]
    Human --> Feedback1[Initial Feedback]

    Feedback1 --> Agent[Agent Review]
    Agent --> Feedback2[Additional Feedback]

    Feedback1 --> Consolidate[Consolidate Feedback]
    Feedback2 --> Consolidate

    Consolidate --> Revise[Revise Code]
    Revise --> Final[Final Review]

    style Human fill:#fff4e1
    style Agent fill:#4a90e2,color:#fff
    style Final fill:#d4edda
```

When to use:

- Catching issues humans might miss
- Ensuring consistency with standards
- Educational feedback for developers

Example scenario:

- Human reviewer focuses on business logic
- Agent checks architectural patterns, naming, structure
- Combined feedback improves code quality

Anti-Patterns to Avoid

❌ Agent Overload

```
graph LR
    Task[Simple Task] --> TooManyAgents[5+ Agents]
    TooManyAgents --> Confusion[Confusion]

    style TooManyAgents fill:#ffcccc
    style Confusion fill:#ff6666,color:#fff
```

Problem: Using too many agents for simple tasks
Solution: Use single agent or standard Copilot for simple requests

---

❌ Wrong Mode for Task

```
graph LR
    QuickQ[Quick Question] --> Agent[Custom Agent]
    Agent --> Overhead[Unnecessary Overhead]

    style Agent fill:#ffcccc
    style Overhead fill:#ff6666,color:#fff
```

Problem: Using agent when Ask mode would suffice
Solution: Use Ask for quick questions, agents for structured analysis

---

❌ Insufficient Context

```
graph LR
    NoContext[No Context] --> Agent[Custom Agent]
    Agent --> Generic[Generic Output]
    Generic --> Useless[Not Actionable]

    style NoContext fill:#ffcccc
    style Useless fill:#ff6666,color:#fff
```

Problem: Invoking agent without necessary context
Solution: Provide relevant files, docs, or descriptions

Best Practices Summary

Pattern	Best For	Avoid For
Sequential Chain	Progressive refinement	Simple tasks
Parallel Review	Comprehensive analysis	Single concern
Iterative Refinement	Complex outputs	Straightforward requests
Agent-Assisted Cycle	Full feature development	Bug fixes
Feedback Loop	Agent creation	Production use
Context-Rich	Complex decisions	Quick questions
Progressive Disclosure	Exploration	Known requirements
Agent Handoff	Team collaboration	Solo work
Multi-Turn	Unclear requirements	Well-defined specs
Second Reviewer	Quality assurance	First pass review

Workflow Decision Tree

```
graph TB
    Start{What are you<br/>trying to do?}

    Start -->|Generate artifacts| Generate{How many<br/>related outputs?}
    Start -->|Review code| Review{How many<br/>dimensions?}
    Start -->|Plan work| Plan{Clear requirements?}
```

```
Generate -->|One| SingleAgent[Single Agent]
Generate -->|Multiple| Sequential[Sequential Chain]

Review -->|One| SingleReview[Single Agent]
Review -->|Multiple| Parallel[Parallel Review]

Plan -->|Yes| DirectAgent[Direct Agent Use]
Plan -->|No| MultiTurn[Multi-Turn Conversation]

style Start fill:#4a90e2,color:#fff
style Generate fill:#e1f5e1
style Review fill:#fff4e1
style Plan fill:#e1e8ff
```

See Also

- Lab 07: Workflow Agents in Action
- Agent Architecture
- Copilot Interaction Models
- Custom Agent Catalog