

SpecKit Workflow Guide

AI Coding Framework - Complete Process Documentation

Generated: 2025-10-26

Version: 1.0

1. Complete Feature Development Flow

SPECKIT COMPLETE WORKFLOW

```
User Input: "Feature description..."
```

```
■ /speckit.specify "description"  
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  
■ Command: .claude/commands/speckit.specify.md  
■ Template: .specify/templates/spec-template.md  
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  
■ Actions:  
■ 1. Generate short-name from description  
■ 2. Run create-new-feature.sh --short-name "..."  
■ 3. Create new branch: ###-feature-name  
■ 4. Fill spec.md with:  
■   • User Stories (P1, P2, P3...)  
■   • Functional Requirements  
■   • Success Criteria  
■   • Edge Cases  
■ 5. Create quality checklist  
■ 6. Resolve [NEEDS CLARIFICATION] markers (max 3)  
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  
■ Output:  
■ ✓ specs/###-feature/spec.md  
■ ✓ specs/###-feature/checklists/requirements.md  
■ ✓ New git branch
```

```
■ /speckit.clarify (OPTIONAL)
■ ████████████████████████████████
■ Command: .claude/commands/speckit.clarify.md
■ 
■ Actions:
■   • Ask up to 5 targeted clarification questions
■   • Encode answers back into spec.md
■   • Ensure consistency across artifacts
■ 
■ Output:
■   ✓ Updated spec.md
```

```

■ /speckit.plan
■ Command: .claude/commands/speckit.plan.md
■ Template: .specify/templates/plan-template.md
■ PHASE 0: Outline & Research
■ 1. Run setup-plan.sh → get paths
■ 2. Read spec.md + constitution.md
■ 3. Fill Technical Context in plan.md:
■   • Language/Version, Dependencies
■   • Storage, Testing, Platform
■   • Mark unknowns: "NEEDS CLARIFICATION"
■ 4. Constitution Check (gate)
■ 5. For each NEEDS CLARIFICATION:
■   • Launch Task agent to research
■   • Investigate technologies, best practices
■ 6. Consolidate findings → research.md:
■   • Decision: [what was chosen]
■   • Rationale: [why chosen]
■   • Alternatives: [what else evaluated]
■ PHASE 1: Design & Contracts

```


2. research.md Generation Deep Dive

HOW research.md IS CREATED (Phase 0 Detail)

INPUTS:

- ```

1. plan.md (from plan-template.md)
 ■ ■ ■ Technical Context section with NEEDS CLARIFICATION markers
 ■ ■ ■
 ■ ■ ■ Example:
 ■ ■ ■ **Language/Version**: NEEDS CLARIFICATION
 ■ ■ ■ **Primary Dependencies**: NEEDS CLARIFICATION
 ■ ■ ■ **Testing**: NEEDS CLARIFICATION
 ■ ■ ■

2. spec.md
 ■ ■ ■ User stories, requirements, constraints
 ■ ■ ■

3. constitution.md
 ■ ■ ■ Core principles to validate against

```



### ■ STEP 1: Extract Unknowns

- ```

■ Parse Technical Context for patterns:
■   • "NEEDS CLARIFICATION" markers
■   • Technology choices needed
■   • Integration patterns needed

```

■ Create research task list:

- ■ Unknown 1: Language selection
- ■ Unknown 2: LCOV parsing library
- ■ Unknown 3: MCP SDK usage patterns
- ■ Unknown 4: Testing framework
- ■ Unknown 5: Storage mechanism



■ STEP 2: Launch Research Agents

- ```

■ For each unknown, use Task tool to launch agent:
■
■ Task 1: "Research Node.js vs Python for MCP server"
■ ■■ WebSearch: "MCP SDK language support"
■ ■■ WebFetch: MCP documentation
■ ■■ Return: Language recommendations
■
■ Task 2: "Research LCOV parsing libraries"
■ ■■ WebSearch: "lcov parser npm"
■ ■■ Compare: download counts, maintenance
■ ■■ Return: Library recommendations
■
■ Task 3: "Find MCP server best practices"
■ ■■ WebFetch: Official MCP docs
■ ■■ Search: Example MCP servers
■ ■■ Return: Architecture patterns
■
■ [Parallel execution of all research tasks]

```

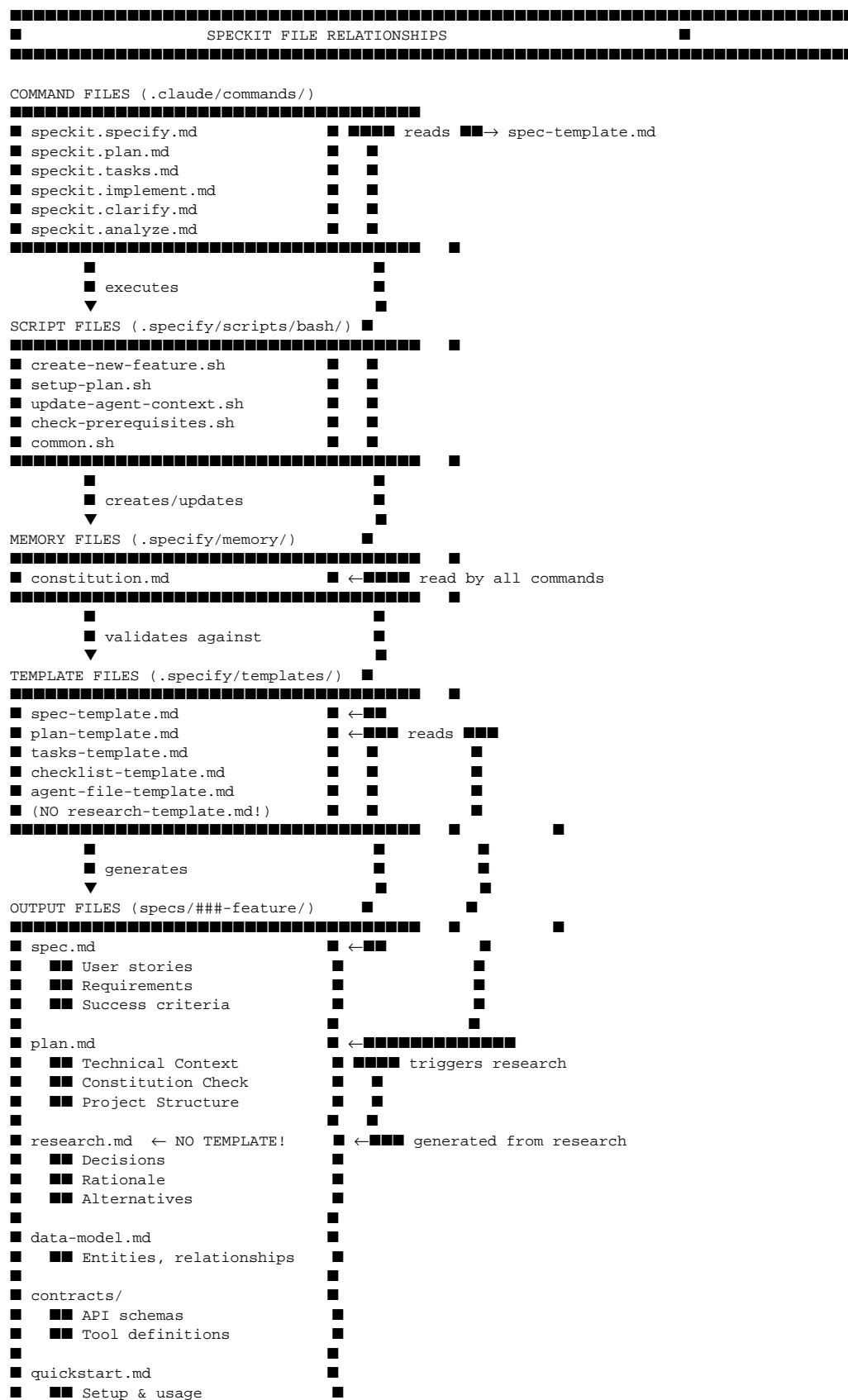


### ■ STEP 3: Consolidate Findings

- Gather agent results and create research.md with structure:

[illegible]

### 3. File Relationships & Dependencies





## 4. Data Flow: From User Input to research.md

User Types:

```
■ "I need an MCP tool to analyze LCOV coverage files..."
```

■

▼

```
■ /speckit.specify
```

```
■ Extracts: User stories, functional requirements
```

```
■ Creates: spec.md
```

```
■ spec.md contains:
```

- • What features are needed (not HOW)
- • Success criteria
- • Edge cases

■

▼

```
■ /speckit.plan - Phase 0 Start
```

```
■ 1. Load plan-template.md
```

```
■ ■■ Has Technical Context section with fields
```

```
■ 2. AI attempts to fill fields from spec.md:
```

```
■ ■ Can I determine from "analyze LCOV files" that:
```

- ✓ This needs to parse files → some language needed
- ✓ "MCP tool" → needs MCP SDK
- ✗ Which language? → NEEDS CLARIFICATION
- ✗ Which LCOV parser library? → NEEDS CLARIFICATION
- ✗ How to store recordings? → NEEDS CLARIFICATION

```
■ 3. Fills plan.md Technical Context:
```

```
■ **Language/Version**: NEEDS CLARIFICATION
```

```
■ **Primary Dependencies**: @modelcontextprotocol/sdk + NEEDS...
```

```
■ **Storage**: NEEDS CLARIFICATION
```

■

▼

```
■ /speckit.plan - Phase 0 Research
```

```
■ 4. Extract all "NEEDS CLARIFICATION" markers
```

```
■ ■■ Create research questions
```

```
■ 5. For each question, launch Task agent:
```

```
■ ■ Agent 1: Research "MCP SDK language support"
```

```
■ ■ → WebSearch + WebFetch
```

```
■ ■ → Returns: "Use TypeScript + Node.js"
```

```
■ ■ Agent 2: Research "LCOV parser Node.js npm"
```

```
■ ■ → Compare libraries
```

```
■ ■ → Returns: "@friedemannsommer/lcov-parser"
```

```
■ ■ Agent 3: Research "recording storage patterns"
```

```
■ ■ → Evaluates against constitution
```

```
■ ■ → Returns: "Temporary filesystem storage"
```

```
■ 6. Consolidate agent results into research.md:
```

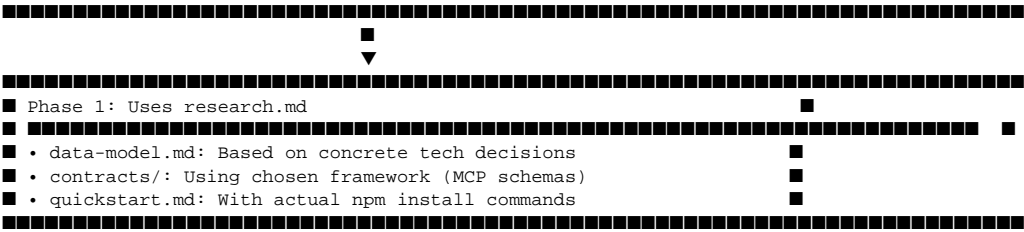
```
■ Format source: .claude/commands/speckit.plan.md lines 47-50
```

```
■ (NO template file exists!)
```

```
■ 7. Write research.md to specs/###-feature/research.md
```

```
■ 8. Update plan.md, replacing NEEDS CLARIFICATION
```





## 5. Key Insights

### KEY FINDINGS:

1. NO research.md TEMPLATE EXISTS  
The research.md file is dynamically generated, not from a template.  
Format is defined in: .claude/commands/speckit.plan.md (lines 47-50)
2. RESEARCH IS TRIGGERED BY UNKNOWNNS  
"NEEDS CLARIFICATION" markers in plan.md's Technical Context section trigger the research phase.
3. TASK AGENTS DO THE RESEARCH  
Multiple Task agents run in parallel to research:
  - Technology choices
  - Library comparisons
  - Best practices
  - Integration patterns
4. CONSTITUTION VALIDATES CHOICES  
All research results must align with project principles defined in .specify/memory/constitution.md
5. RESEARCH INFORMS ALL DOWNSTREAM ARTIFACTS  
research.md results populate:
  - plan.md (Technical Context)
  - data-model.md (entity design)
  - contracts/ (API schema choices)
  - quickstart.md (setup instructions)

### FILE LOCATIONS REFERENCE:

#### Commands:

```
.claude/commands/speckit.specify.md
.claude/commands/speckit.plan.md
.claude/commands/speckit.tasks.md
.claude/commands/speckit.implement.md
```

#### Templates:

```
.specify/templates/spec-template.md
.specify/templates/plan-template.md
.specify/templates/tasks-template.md
(NO research-template.md)
```

#### Scripts:

```
.specify/scripts/bash/create-new-feature.sh
.specify/scripts/bash/setup-plan.sh
.specify/scripts/bash/update-agent-context.sh
```

#### Memory:

```
.specify/memory/constitution.md
```

#### Outputs (per feature):

```
specs/###-feature/spec.md
specs/###-feature/plan.md
specs/###-feature/research.md ← Dynamically generated
specs/###-feature/data-model.md
specs/###-feature/contracts/
specs/###-feature/quickstart.md
specs/###-feature/tasks.md
specs/###-feature/checklists/
```

### RESEARCH.MD FORMAT:

```
For each research topic:
 Decision: [what was chosen]
 Rationale: [why chosen]
 Alternatives considered: [what else was evaluated]
```