BMAD-METHOD Structure Analysis for Advertising Research Expansion

Overview

This document provides a comprehensive analysis of the existing BMAD-METHOD framework structure to guide the creation of new advertising research components.

Directory Structure

```
bmad-core/
├── agent-teams/      # Team configuration files
├── agents/           # Individual agent definitions
├── checklists/       # Quality and process checklists
├── core-config.yaml  # Core configuration
├── data/             # Knowledge base and reference data
├── tasks/            # Reusable task definitions
├── templates/        # Document templates (YAML format)
└── workflows/        # Workflow definitions
```

Agent File Structure

Sample Agent: analyst.md

```
---
name: analyst
role: Business Analyst
persona: |
  You are a meticulous Business Analyst with expertise in requirements elicitation,
  documentation, and stakeholder management. You excel at:
  - Conducting thorough requirements gathering sessions
  - Creating clear, comprehensive documentation
  - Identifying gaps and ambiguities in requirements
  - Facilitating communication between technical and business stakeholders
  - Applying various elicitation techniques (interviews, workshops, observation)

  Your approach is systematic and detail-oriented, ensuring nothing is overlooked.
  You ask probing questions to uncover hidden requirements and validate assumptions.

commands:
  - name: elicit
    description: Conduct requirements elicitation using various techniques
    usage: "@analyst elicit [technique] [context]"

  - name: document
    description: Create or update requirements documentation
    usage: "@analyst document [artifact-type]"

  - name: validate
    description: Validate requirements for completeness and clarity
    usage: "@analyst validate [requirements-doc]"

  - name: trace
    description: Create requirements traceability matrix
    usage: "@analyst trace"

dependencies:
  tasks:
    - advanced-elicitation
    - trace-requirements
    - document-project
  templates:
    - prd-tmpl
    - project-brief-tmpl
  checklists:
    - pm-checklist
  data:
    - elicitation-methods
    - brainstorming-techniques
---

# Business Analyst Agent

## Core Responsibilities

1. **Requirements Elicitation**
   - Conduct stakeholder interviews
   - Facilitate brainstorming sessions
   - Perform competitive analysis
   - Document user stories and use cases

2. **Documentation**
   - Create Product Requirements Documents (PRDs)
   - Maintain requirements traceability
   - Document business processes
   - Create project briefs
```

```
3. **Validation & Verification**
   - Review requirements for completeness
   - Validate against business objectives
   - Ensure technical feasibility
   - Identify conflicts and dependencies

## Workflow Integration

The analyst works closely with:
- **Product Owner**: To understand business vision
- **Architect**: To validate technical feasibility
- **PM**: To align with project constraints
- **Dev Team**: To clarify implementation details

## Command Details

### elicit
Conducts requirements gathering using specified techniques from the elicitation-
methods knowledge base.

### document
Creates structured documentation using appropriate templates (PRD, project brief,
etc.).

### validate
Reviews requirements against checklists and best practices to ensure quality.

### trace
Creates and maintains requirements traceability to ensure all requirements are ad-
dressed.
```

All Agent Files:

- analyst.md
- architect.md
- bmad-master.md
- bmad-orchestrator.md
- dev.md
- pm.md
- po.md
- qa.md
- sm.md
- ux-expert.md

Sample Agent: architect.md

```
---
name: architect
role: Software Architect
persona: |
  You are an experienced Software Architect with deep expertise in system design,
  technology selection, and architectural patterns. You excel at:
  - Designing scalable, maintainable system architectures
  - Evaluating and selecting appropriate technologies
  - Creating technical specifications and diagrams
  - Identifying and mitigating technical risks
  - Balancing technical excellence with business constraints

  Your approach is pragmatic and forward-thinking, considering both immediate
  needs and long-term maintainability. You communicate complex technical concepts
  clearly to both technical and non-technical stakeholders.

commands:
  - name: design
    description: Create architectural design for a system or component
    usage: "@architect design [scope]"

  - name: review
    description: Review existing architecture or design decisions
    usage: "@architect review [artifact]"

  - name: assess
    description: Assess technical feasibility and risks
    usage: "@architect assess [requirements]"

  - name: specify
    description: Create detailed technical specifications
    usage: "@architect specify [component]"

dependencies:
  tasks:
    - nfr-assess
    - risk-profile
  templates:
    - architecture-tmpl
    - front-end-architecture-tmpl
    - fullstack-architecture-tmpl
    - brownfield-architecture-tmpl
  checklists:
    - architect-checklist
  data:
    - technical-preferences
    - bmad-kb
---

# Software Architect Agent

## Core Responsibilities

1. **System Design**
   - Define overall system architecture
   - Select appropriate architectural patterns
   - Design component interactions
   - Create architecture diagrams

2. **Technology Selection**
   - Evaluate technology options
   - Consider technical preferences
   - Assess team capabilities
   - Balance innovation with stability

3. **Technical Specification**
   - Document architectural decisions
   - Create API specifications
   - Define data models
   - Specify integration points

4. **Risk Management**
   - Identify technical risks
   - Assess non-functional requirements
   - Plan mitigation strategies
   - Monitor technical debt

## Workflow Integration

The architect collaborates with:
- **Product Owner**: To understand business requirements
- **Analyst**: To validate technical feasibility
- **Dev Team**: To guide implementation
- **QA**: To define testing strategies

## Command Details

### design
Creates comprehensive architectural designs using appropriate templates based on
project type (greenfield/brownfield, frontend/backend/fullstack).

### review
Evaluates existing architectures against best practices and project requirements using
the architect checklist.

### assess
Analyzes non-functional requirements and creates risk profiles for technical
decisions.

### specify
Produces detailed technical specifications for components, APIs, and integrations.
```

Task File Structure

Sample Task: create-next-story.md

```
# Create Next Story Task

## Purpose
Generate the next user story in the development backlog based on the current project
state, PRD, and architecture.

## Inputs
- Current PRD
- Architecture document
- Existing stories (if any)
- Story template
- Definition of Done checklist

## Process

### 1. Review Context
- Read the PRD to understand overall requirements
- Review architecture to understand technical constraints
- Examine existing stories to avoid duplication
- Identify next logical feature or component

### 2. Story Creation
- Use the story template (@template:story-tmpl)
- Write clear user story in format: "As a [user], I want [goal], so that [benefit]"
- Define acceptance criteria
- Estimate complexity/effort
- Identify dependencies

### 3. Technical Details
- Specify affected components
- List required APIs or services
- Note data model changes
- Identify integration points

### 4. Quality Criteria
- Apply Definition of Done checklist (@checklist:story-dod-checklist)
- Ensure testability
- Verify completeness
- Check for ambiguities

### 5. Validation
- Confirm alignment with PRD
- Verify architectural consistency
- Check for missing requirements
- Validate acceptance criteria

## Outputs
- Completed user story document
- Updated backlog
- Dependency map (if applicable)

## Success Criteria
- Story is clear and unambiguous
- Acceptance criteria are testable
- Technical details are sufficient for implementation
- Story aligns with PRD and architecture
- All DoD checklist items are addressed

## Related
- @template:story-tmpl
- @checklist:story-dod-checklist
- @task:validate-next-story
```

All Task Files:

- advanced-elicitation.md
- apply-qa-fixes.md
- brownfield-create-epic.md
- brownfield-create-story.md
- correct-course.md
- create-brownfield-story.md
- create-deep-research-prompt.md
- create-next-story.md
- document-project.md
- facilitate-brainstorming-session.md
- generate-ai-frontend-prompt.md
- index-docs.md
- kb-mode-interaction.md
- nfr-assess.md
- qa-gate.md
- review-story.md
- risk-profile.md
- shard-doc.md
- test-design.md
- trace-requirements.md
- validate-next-story.md

Template File Structure

Sample Template: prd-tmpl.yaml

```yaml
name: prd-tmpl
description: Product Requirements Document Template
version: 1.0

sections:
  - name: executive_summary
    title: Executive Summary
    description: High-level overview of the product and its objectives
    fields:
      - name: product_name
        type: text
        required: true
      - name: vision
        type: textarea
        required: true
      - name: objectives
        type: list
        required: true

  - name: problem_statement
    title: Problem Statement
    description: Clear articulation of the problem being solved
    fields:
      - name: current_situation
        type: textarea
        required: true
      - name: pain_points
        type: list
        required: true
      - name: impact
        type: textarea
        required: true

  - name: target_users
    title: Target Users
    description: Definition of primary and secondary user personas
    fields:
      - name: primary_personas
        type: list
        required: true
      - name: secondary_personas
        type: list
        required: false
      - name: user_needs
        type: list
        required: true

  - name: solution_overview
    title: Solution Overview
    description: High-level description of the proposed solution
    fields:
      - name: approach
        type: textarea
        required: true
      - name: key_features
        type: list
        required: true
      - name: differentiators
        type: list
        required: false

  - name: functional_requirements
```

```yaml
    title: Functional Requirements
    description: Detailed functional requirements organized by feature area
    fields:
      - name: feature_areas
        type: nested_list
        required: true
        structure:
          - feature_name
          - requirements
          - priority

  - name: non_functional_requirements
    title: Non-Functional Requirements
    description: Performance, security, scalability, and other quality attributes
    fields:
      - name: performance
        type: list
        required: true
      - name: security
        type: list
        required: true
      - name: scalability
        type: list
        required: true
      - name: usability
        type: list
        required: true
      - name: reliability
        type: list
        required: true

  - name: constraints
    title: Constraints
    description: Technical, business, and regulatory constraints
    fields:
      - name: technical_constraints
        type: list
        required: false
      - name: business_constraints
        type: list
        required: false
      - name: regulatory_constraints
        type: list
        required: false

  - name: assumptions
    title: Assumptions
    description: Key assumptions underlying the requirements
    fields:
      - name: assumptions_list
        type: list
        required: true

  - name: dependencies
    title: Dependencies
    description: External dependencies and integrations
    fields:
      - name: external_systems
        type: list
        required: false
      - name: third_party_services
        type: list
        required: false
      - name: internal_dependencies
        type: list
        required: false

  - name: success_metrics
    title: Success Metrics
    description: KPIs and metrics to measure product success
    fields:
      - name: kpis
        type: list
        required: true
      - name: measurement_approach
        type: textarea
        required: true

metadata:
  author: Product Owner
  reviewers:
    - Business Analyst
    - Software Architect
    - Project Manager
  approval_required: true
```

All Template Files:

- architecture-tmpl.yaml
- brainstorming-output-tmpl.yaml
- brownfield-architecture-tmpl.yaml
- brownfield-prd-tmpl.yaml
- competitor-analysis-tmpl.yaml
- front-end-architecture-tmpl.yaml
- front-end-spec-tmpl.yaml
- fullstack-architecture-tmpl.yaml
- market-research-tmpl.yaml
- prd-tmpl.yaml
- project-brief-tmpl.yaml
- qa-gate-tmpl.yaml
- story-tmpl.yaml

Checklist File Structure

Sample Checklist: story-dod-checklist.md

```
# Story Definition of Done Checklist

## Story Content Quality

- [ ] User story follows standard format: "As a [user], I want [goal], so that [bene-
fit]"
- [ ] Story title is clear and descriptive
- [ ] Story description provides sufficient context
- [ ] Business value is clearly articulated
- [ ] Story is sized appropriately (not too large or too small)

## Acceptance Criteria

- [ ] Acceptance criteria are clearly defined
- [ ] Criteria are testable and measurable
- [ ] All happy path scenarios are covered
- [ ] Edge cases and error conditions are addressed
- [ ] Criteria include both functional and non-functional aspects

## Technical Details

- [ ] Affected components are identified
- [ ] Required APIs or services are specified
- [ ] Data model changes are documented
- [ ] Integration points are defined
- [ ] Technical dependencies are listed

## Dependencies

- [ ] Dependencies on other stories are identified
- [ ] External dependencies are documented
- [ ] Blocking issues are noted
- [ ] Required resources are specified

## Testability

- [ ] Story is testable
- [ ] Test scenarios can be derived from acceptance criteria
- [ ] Test data requirements are identified
- [ ] Testing approach is feasible

## Documentation

- [ ] Story references relevant PRD sections
- [ ] Architecture alignment is verified
- [ ] Related stories are linked
- [ ] Supporting documentation is attached

## Estimation

- [ ] Story is estimated (story points or hours)
- [ ] Estimation includes all aspects (dev, test, review)
- [ ] Complexity is assessed
- [ ] Risk factors are considered

## Completeness

- [ ] No ambiguities or unclear requirements
- [ ] All questions are answered
- [ ] Story is ready for implementation
- [ ] Team has reviewed and accepted the story
```

```
## Compliance

- [ ] Security requirements are addressed
- [ ] Performance requirements are specified
- [ ] Accessibility requirements are included
- [ ] Regulatory requirements are met (if applicable)

## Review

- [ ] Product Owner has reviewed and approved
- [ ] Architect has validated technical approach
- [ ] QA has confirmed testability
- [ ] Development team has no blocking questions
```

All Checklist Files:

- architect-checklist.md
- change-checklist.md
- pm-checklist.md
- po-master-checklist.md
- story-dod-checklist.md
- story-draft-checklist.md

Agent Team Configuration Structure

Sample Team: team-fullstack.yaml

```yaml
team_name: fullstack-development
description: Complete team for full-stack application development from concept to
deployment

agents:
  - name: po
    role: Product Owner
    primary: true
    responsibilities:
      - Define product vision
      - Prioritize backlog
      - Accept completed work

  - name: analyst
    role: Business Analyst
    responsibilities:
      - Elicit requirements
      - Document specifications
      - Validate completeness

  - name: architect
    role: Software Architect
    responsibilities:
      - Design system architecture
      - Select technologies
      - Create technical specifications

  - name: ux-expert
    role: UX Expert
    responsibilities:
      - Design user experience
      - Create wireframes
      - Validate usability

  - name: dev
    role: Developer
    responsibilities:
      - Implement features
      - Write tests
      - Code reviews

  - name: qa
    role: QA Engineer
    responsibilities:
      - Design test cases
      - Execute tests
      - Report defects

  - name: pm
    role: Project Manager
    responsibilities:
      - Coordinate activities
      - Track progress
      - Manage risks

workflows:
  - greenfield-fullstack
  - brownfield-fullstack

collaboration_patterns:
  - pattern: requirements_gathering
    participants: [po, analyst, ux-expert]
  - pattern: architecture_design
    participants: [architect, dev, qa]

  - pattern: story_refinement
    participants: [po, analyst, dev, qa]

  - pattern: implementation
    participants: [dev, qa]

  - pattern: review_and_acceptance
    participants: [po, qa, pm]

communication:
  daily_standup: true
  sprint_planning: true
  sprint_review: true
  retrospective: true
```

All Team Files:

- team-all.yaml
- team-fullstack.yaml
- team-ide-minimal.yaml
- team-no-ui.yaml

Workflow Structure

Sample Workflow: greenfield-fullstack.yaml

```yaml
workflow_name: greenfield-fullstack
description: Complete workflow for building a new full-stack application from scratch
version: 1.0

phases:
  - phase: 1
    name: discovery
    description: Initial discovery and requirements gathering
    agents:
      - po
      - analyst
    tasks:
      - advanced-elicitation
      - facilitate-brainstorming-session
    artifacts:
      - project-brief
      - initial-requirements
    duration: 1-2 weeks

  - phase: 2
    name: requirements
    description: Detailed requirements documentation
    agents:
      - analyst
      - po
    tasks:
      - document-project
      - trace-requirements
    templates:
      - prd-tmpl
      - project-brief-tmpl
    artifacts:
      - prd
      - requirements-traceability-matrix
    duration: 1-2 weeks

  - phase: 3
    name: architecture
    description: System architecture and design
    agents:
      - architect
      - ux-expert
    tasks:
      - nfr-assess
      - risk-profile
    templates:
      - fullstack-architecture-tmpl
      - front-end-spec-tmpl
    checklists:
      - architect-checklist
    artifacts:
      - architecture-document
      - technical-specifications
      - ux-designs
    duration: 1-2 weeks

  - phase: 4
    name: planning
    description: Sprint planning and story creation
    agents:
      - po
      - pm
      - analyst
    tasks:
      - create-next-story
      - validate-next-story
    templates:
      - story-tmpl
    checklists:
      - story-dod-checklist
      - pm-checklist
    artifacts:
      - product-backlog
      - sprint-plan
    duration: 1 week

  - phase: 5
    name: development
    description: Iterative development sprints
    agents:
      - dev
      - qa
      - pm
    tasks:
      - apply-qa-fixes
      - qa-gate
      - test-design
    templates:
      - qa-gate-tmpl
    checklists:
      - story-dod-checklist
    artifacts:
      - working-software
      - test-results
      - sprint-reports
    duration: 2-4 weeks per sprint
    iterative: true

  - phase: 6
    name: qa_validation
    description: Comprehensive quality assurance
    agents:
      - qa
      - po
    tasks:
      - qa-gate
      - test-design
    templates:
      - qa-gate-tmpl
    artifacts:
      - qa-report
      - defect-log
    duration: 1 week

  - phase: 7
    name: deployment
    description: Production deployment and handoff
    agents:
      - dev
      - pm
      - po
    artifacts:
      - deployment-guide
      - user-documentation
      - handoff-package
    duration: 1 week

gates:
  - gate: requirements_approval
    after_phase: 2
    approvers: [po]
    criteria:
      - PRD is complete
      - Stakeholders have reviewed
      - Requirements are validated

  - gate: architecture_approval
    after_phase: 3
    approvers: [architect, po]
    criteria:
      - Architecture is documented
      - Technical risks are assessed
      - Design is reviewed

  - gate: sprint_acceptance
    after_phase: 5
    approvers: [po, qa]
    criteria:
      - All stories meet DoD
      - QA gate passed
      - No critical defects

  - gate: production_readiness
    after_phase: 6
    approvers: [po, pm, qa]
    criteria:
      - All acceptance criteria met
      - Performance validated
      - Documentation complete

success_criteria:
  - All requirements implemented
  - Quality gates passed
  - Stakeholder acceptance achieved
  - Documentation delivered
```

All Workflow Files:

- brownfield-fullstack.yaml
- brownfield-service.yaml
- brownfield-ui.yaml
- greenfield-fullstack.yaml
- greenfield-service.yaml
- greenfield-ui.yaml

Core Configuration

core-config.yaml

```yaml
bmad_version: "2.0"
framework: BMAD-METHOD

directories:
  agents: bmad-core/agents
  tasks: bmad-core/tasks
  templates: bmad-core/templates
  checklists: bmad-core/checklists
  agent_teams: bmad-core/agent-teams
  workflows: bmad-core/workflows
  data: bmad-core/data

conventions:
  agent_file_format: markdown
  task_file_format: markdown
  template_file_format: yaml
  checklist_file_format: markdown
  team_file_format: yaml
  workflow_file_format: yaml

  naming:
    agents: "{role-name}.md"
    tasks: "{action-description}.md"
    templates: "{type}-tmpl.yaml"
    checklists: "{purpose}-checklist.md"
    teams: "team-{name}.yaml"
    workflows: "{context}-{type}.yaml"

reference_syntax:
  task: "@task:{task-name}"
  template: "@template:{template-name}"
  checklist: "@checklist:{checklist-name}"
  data: "@data:{data-file-name}"
  agent: "@{agent-name}"

supported_project_types:
  - greenfield-fullstack
  - greenfield-service
  - greenfield-ui
  - brownfield-fullstack
  - brownfield-service
  - brownfield-ui

default_teams:
  - team-fullstack
  - team-no-ui
  - team-ide-minimal
  - team-all

quality_gates:
  - requirements_approval
  - architecture_approval
  - sprint_acceptance
  - production_readiness
```

Key Patterns and Conventions Identified

1. Agent File Patterns

YAML Header Structure:
- `name`: Agent identifier (lowercase with hyphens)
- `role`: Human-readable role name
- `persona`: Detailed character description
- `commands`: List of available commands with descriptions
- `dependencies`: References to tasks, templates, checklists, and data files

File Naming:
- Format: `{role-name}.md`
- Examples: `analyst.md`, `architect.md`, `pm.md`
- All lowercase with hyphens for multi-word names

Content Structure:
1. YAML frontmatter (between `---` markers)
2. Detailed persona description
3. Command definitions with usage examples
4. Workflow integration notes

2. Task File Patterns

Structure:
- Markdown format with clear sections
- Step-by-step instructions
- Input/output specifications
- References to templates and checklists

Naming Convention:
- Format: `{action-description}.md`
- Examples: `create-next-story.md`, `qa-gate.md`, `nfr-assess.md`
- Descriptive, action-oriented names

3. Template File Patterns

YAML Structure:
- `name`: Template identifier
- `description`: Purpose and usage
- `sections`: Structured content areas
- `metadata`: Version, author, date information

Naming Convention:
- Format: `{document-type}-tmpl.yaml`
- Examples: `prd-tmpl.yaml`, `architecture-tmpl.yaml`, `story-tmpl.yaml`
- Always ends with `-tmpl.yaml`

4. Checklist Patterns

Structure:
- Markdown format with checkbox items
- Grouped by categories/phases
- Clear acceptance criteria
- References to related documents

Naming Convention:
- Format: `{purpose}-checklist.md`
- Examples: `story-dod-checklist.md`, `pm-checklist.md`
- Always ends with `-checklist.md`

5. Agent Team Configuration

YAML Structure:
- `team_name`: Team identifier
- `description`: Team purpose
- `agents`: List of agent references
- `workflows`: Associated workflow definitions

Naming Convention:
- Format: `team-{name}.yaml`
- Examples: `team-fullstack.yaml`, `team-no-ui.yaml`

6. Workflow Patterns

YAML Structure:
- `workflow_name`: Workflow identifier
- `description`: Workflow purpose
- `phases`: Sequential execution phases
- `agents`: Agent assignments per phase
- `artifacts`: Expected outputs

Naming Convention:
- Format: `{context}-{type}.yaml`
- Examples: `greenfield-fullstack.yaml`, `brownfield-service.yaml`

7. Dependency References

Pattern:
- Tasks reference templates: `@template:prd-tmpl`
- Agents reference tasks: `@task:create-next-story`
- Checklists referenced: `@checklist:story-dod-checklist`
- Data files: `@data:bmad-kb`

8. File Organization

Directory Purpose:
- `agents/`: Individual agent definitions (MD files)
- `tasks/`: Reusable task procedures (MD files)
- `templates/`: Document templates (YAML files)
- `checklists/`: Quality gates (MD files)
- `agent-teams/`: Team configurations (YAML files)
- `workflows/`: Process definitions (YAML files)
- `data/`: Knowledge base and reference materials (MD files)

9. Naming Conventions Summary

Component	Format	Extension	Example
Agent	`{role-name}`	`.md`	`analyst.md`
Task	`{action-description}`	`.md`	`create-next-story.md`
Template	`{type}-tmpl`	`.yaml`	`prd-tmpl.yaml`
Checklist	`{purpose}-checklist`	`.md`	`story-dod-checklist.md`
Team	`team-{name}`	`.yaml`	`team-fullstack.yaml`
Workflow	`{context}-{type}`	`.yaml`	`greenfield-fullstack.yaml`
Data	`{topic}`	`.md`	`bmad-kb.md`

Recommendations for Advertising Research Expansion

New Components to Create

1. **Agents:**
   - `media-planner.md` - Media planning and buying strategy
   - `creative-strategist.md` - Creative concept development
   - `brand-researcher.md` - Brand positioning and perception
   - `consumer-insights.md` - Audience research and segmentation
   - `campaign-analyst.md` - Campaign performance analysis

2. **Tasks:**
   - `conduct-brand-audit.md`
   - `develop-media-plan.md`
   - `analyze-campaign-performance.md`
   - `create-creative-brief.md`
   - `segment-audience.md`

3. **Templates:**
   - `brand-audit-tmpl.yaml`
   - `media-plan-tmpl.yaml`
   - `creative-brief-tmpl.yaml`
   - `campaign-report-tmpl.yaml`
   - `audience-persona-tmpl.yaml`

4. **Checklists:**
   - `brand-audit-checklist.md`
   - `media-plan-checklist.md`
   - `creative-brief-checklist.md`
   - `campaign-launch-checklist.md`

5. **Agent Teams:**
   - `team-advertising-research.yaml`
   - `team-brand-strategy.yaml`
   - `team-campaign-planning.yaml`

6. **Workflows:**
   - `brand-audit-workflow.yaml`
   - `campaign-planning-workflow.yaml`
   - `creative-development-workflow.yaml`

7. **Data/Knowledge Base:**
   - `advertising-research-kb.md`
   - `media-planning-frameworks.md`
   - `creative-strategy-models.md`
   - `audience-segmentation-methods.md`

Integration Points

- Existing `analyst.md` can collaborate with new advertising research agents
- `pm.md` can coordinate advertising campaign projects
- Templates should follow existing YAML structure patterns
- Tasks should reference new templates using `@template:` notation
- Checklists should align with advertising industry standards

Next Steps

1. Review user specification document for detailed requirements
2. Create agent definitions following identified patterns
3. Develop supporting tasks, templates, and checklists
4. Configure agent teams for advertising research workflows
5. Define workflows for common advertising research scenarios
6. Add knowledge base content for advertising domain