last-redoc-date
2025-10-12

BMM Workflows - The Complete v6 Flow

The BMM (BMAD Method Module) orchestrates software development through four distinct phases, each with specialized workflows that adapt to project scale (Level 0-4) and context (greenfield vs brownfield). This document serves as the master guide for understanding how these workflows interconnect to deliver the revolutionary v6 methodology.

Core v6 Innovations

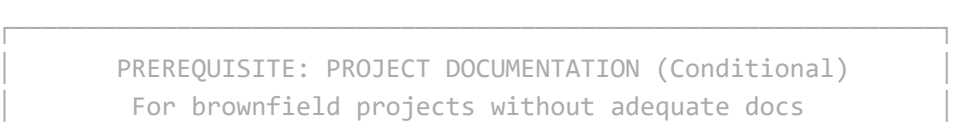
Scale-Adaptive Planning: Projects automatically route through different workflows based on complexity (Level 0-4), ensuring appropriate documentation and process overhead.

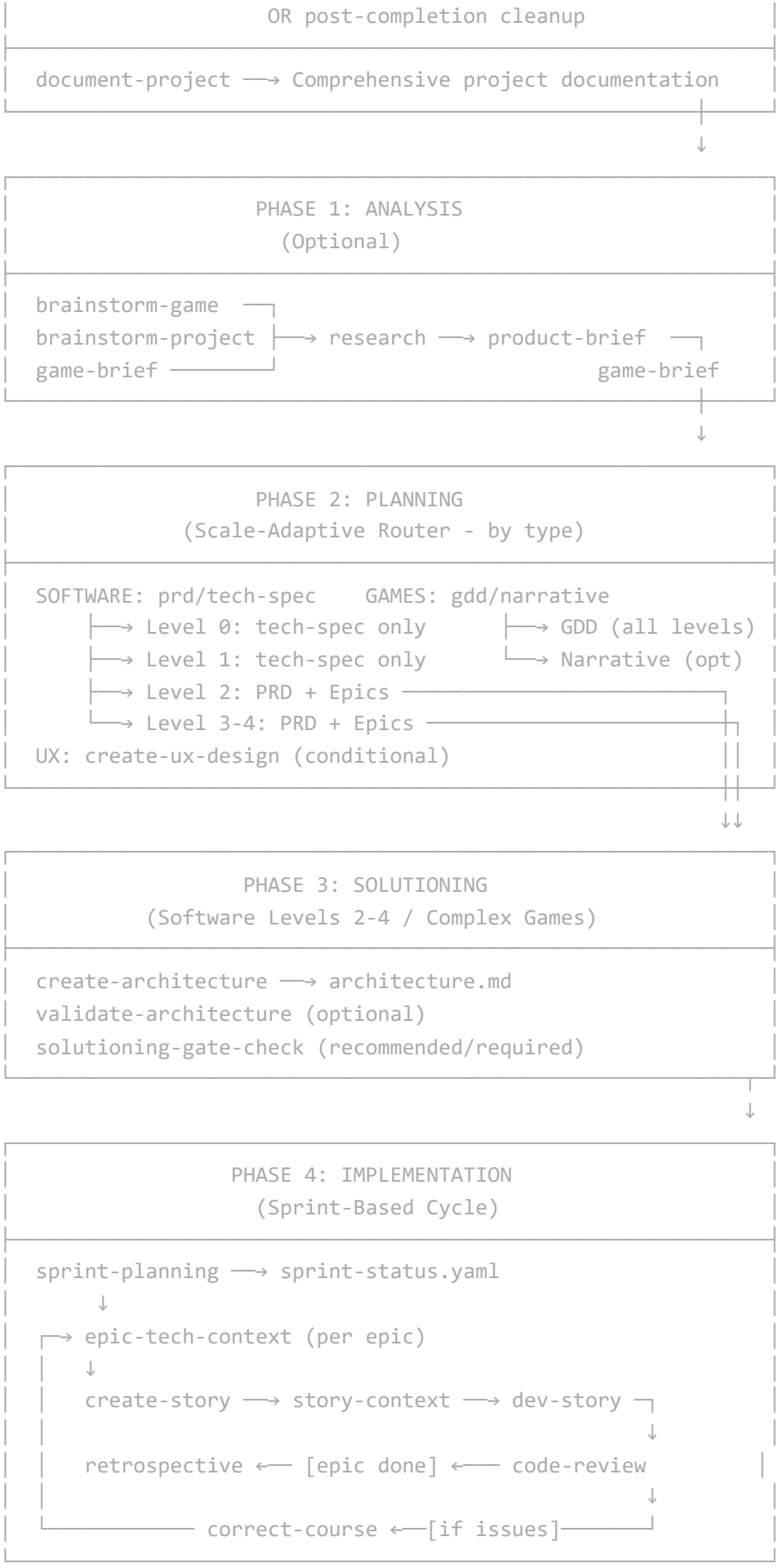
Just-In-Time Design: Technical specifications are created one epic at a time during implementation, not all upfront, incorporating learnings as the project evolves.

Dynamic Expertise Injection: Story-context workflows provide targeted technical guidance per story, replacing static documentation with contextual expertise.

Continuous Learning Loop: Retrospectives feed improvements back into workflows, making each epic smoother than the last.

The Four Phases (Plus Documentation Prerequisite)





Universal Entry Point: workflow-status

Before starting any workflow, check your status!

The `workflow-status` workflow is the **universal entry point** for all BMM workflows, if you have not already set up your workflow, run `workflow-init`, but even if you just run the workflow-status and the file does not exist you should still be directed to run workflow-init.

What it does:

- ✅ Checks for existing workflow status file
- ✅ Displays current phase, progress, and next action
- ✅ Helps new users plan their workflow approach
- ✅ Guides brownfield projects to documentation first
- ✅ Routes to appropriate workflows based on context

No status file? It will:

1. Ask about project context (greenfield vs brownfield)
2. Generate your bmm-workflow-status.md file.

Status file exists? It will:

1. Display current phase and progress
2. Show Phase 4 implementation state (BACKLOG/TODO/IN PROGRESS/DONE)
3. Recommend exact next action
4. Offer to change workflow or display menu

All phase 1-3 workflows should check workflow-status on start of the workflow.

Documentation Prerequisite (Brownfield Projects)

NOT a numbered phase - this is a prerequisite workflow for brownfield projects without adequate documentation, OR a post-completion tool for creating clean source-of-truth documentation after Phases 1-4 are complete.

Purpose

The `document-project` workflow serves TWO critical purposes:

1. **Pre-Phase 1 Prerequisite (Brownfield)**: Run BEFORE planning to understand existing codebases
2. **Post-Phase 4 Documentation (Any Project)**: Run AFTER completion to create superior documentation that replaces scattered PRD/architecture/story artifacts

Workflows

Workflow	Agent	Purpose	Output	When to Use
document-project	Analyst	Analyze and document	Comprehensive project	Brownfield without docs OR post-

Workflow	Agent	Purpose	Output	When to Use
		entire project	documentation	completion cleanup (any scale)

Use Cases

Use Case 1: Brownfield Prerequisite

```
workflow-init detects undocumented brownfield
    ↓
document-project (generates index.md, architecture.md, etc.)
    ↓
Phase 1 (optional) → Phase 2 (planning with full context)
```

Use Case 2: Post-Completion Documentation

```
Phase 4 Implementation Complete
    ↓
document-project (scans final codebase)
    ↓
Produces clean, LLM-optimized docs > scattered phase artifacts
```

Why it's superior: Creates comprehensive, consistent documentation that both humans and LLMs can use to understand projects of any size or complexity - often better than manually-maintained PRDs, architecture docs, and story files.

Phase 1: Analysis (Optional)

Optional workflows for project discovery and requirements gathering. Output feeds into Phase 2 planning.

Workflows

Workflow	Agent	Purpose	Output	When to Use
workflow-status	Analyst	Universal entry point and status checker	Status display + guidance	**Start here!**
workflow-init	Analyst	Generate an initial workflow status file	Status display + guidance	**OR start here!**
brainstorm-game	Game Designer	Game concept ideation using 5 methodologies	Concept proposals	New game projects

Workflow	Agent	Purpose	Output	When to Use
brainstorm-project	Analyst	Software solution exploration	Architecture proposals	New software projects
game-brief	Game Designer	Structured game design foundation	Game brief document	Before GDD creation
product-brief	Analyst	Strategic product planning culmination	Product brief	End of analysis phase
research	Analyst	Multi-mode research (market/technical/deep)	Research artifacts	When evidence needed

Flow

```
workflow-status (check) → Brainstorming → Research → Brief → Planning (Phase 2)
```

Phase 2: Planning (Required)

Scale Levels

Level	Scope	Outputs	Next Phase
0	Single atomic change	tech-spec + 1 story	→ Implementation
1	1-10 stories, 1 epic	tech-spec + epic + 2-3 stories	→ Implementation
2	5-15 stories, 1-2 epics	PRD + epics	→ Solutioning → Implementation
3	12-40 stories, 2-5 epics	PRD + epics	→ Solutioning → Implementation
4	40+ stories, 5+ epics	PRD + epics	→ Solutioning → Implementation

Available Workflows

Workflow	Agent	Purpose	Output	Levels
prd	PM	Product Requirements Document	PRD.md + epics	2-4
tech-spec	PM	Technical specification	tech-spec.md	0-1
gdd	PM	Game Design Document	GDD.md	Games (all)
narrative	PM	Game narrative design	narrative.md	Games (opt)
create-ux-design	UX Designer	User experience and interface design	ux-design.md	Conditional

Key Outputs

- **PRD.md**: Product Requirements Document (Levels 2-4)
- **Epics.md**: Epic breakdown with stories (Levels 2-4)
- **tech-spec.md**: Technical specification (Levels 0-1)
- **story-{slug}.md**: Single user story (Level 0)
- **story-{slug}-1.md, story-{slug}-2.md, story-{slug}-3.md**: User stories (Level 1)
- **GDD.md**: Game Design Document (game projects)
- **narrative.md**: Narrative design (game projects, optional)
- **ux-design.md**: UX specification (conditional, UI-heavy projects)
- **bmm-workflow-status.md**: Versioned workflow state tracking

Phase 3: Solutioning (Levels 2-4)

Architecture and technical design phase for medium to complex projects.

Workflows

Workflow	Agent	Purpose	Output	When
create-architecture	Architect	Create system-wide architecture	architecture.md with ADRs	Levels 2-4
validate-architecture	Architect	Validate architecture design	Validation report	Optional
solutioning-gate-check	Architect	Validate PRD + UX + architecture	Gate check report	Recommended

Architecture Scope by Level

- **Level 2**: Lightweight architecture document focusing on key technical decisions

- **Level 3-4**: Comprehensive architecture with detailed ADRs, system diagrams, integration patterns

Phase 4: Implementation (Iterative)

The core development cycle that transforms requirements into working software through sprint-based iteration.

Sprint Planning - The Phase 4 Entry Point

Phase 4 begins with the **sprint-planning** workflow, which generates a `sprint-status.yaml` file that serves as the single source of truth for all implementation tracking.

What sprint-planning does:

1. Extracts all epics and stories from epic files
2. Creates ordered status tracking for every work item
3. Auto-detects existing story files and contexts
4. Maintains status through the development lifecycle

The Sprint Status System

Phase 4 uses a 6-state lifecycle tracked in `sprint-status.yaml`:

Epic Status Flow:

```
backlog → contexted
```

Story Status Flow:

```
backlog → drafted → ready-for-dev → in-progress → review → done
```

Retrospective Status:

```
optional ↔ completed
```

Status Definitions

Epic Statuses:

- **backlog**: Epic exists in epic file but not yet contexted
- **contexted**: Epic technical context created (prerequisite for drafting stories)

Story Statuses:

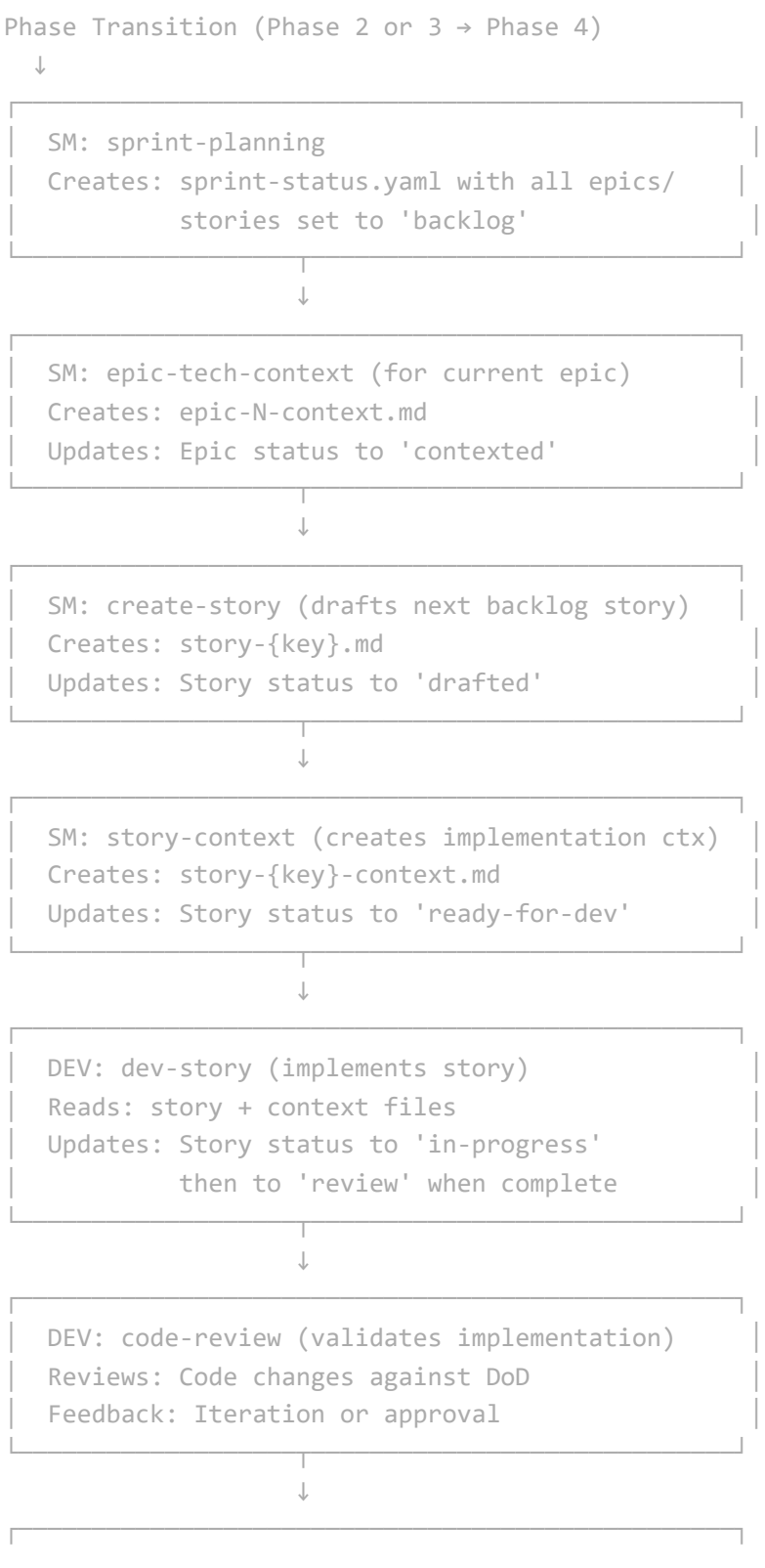
- **backlog**: Story only exists in epic file, not yet drafted

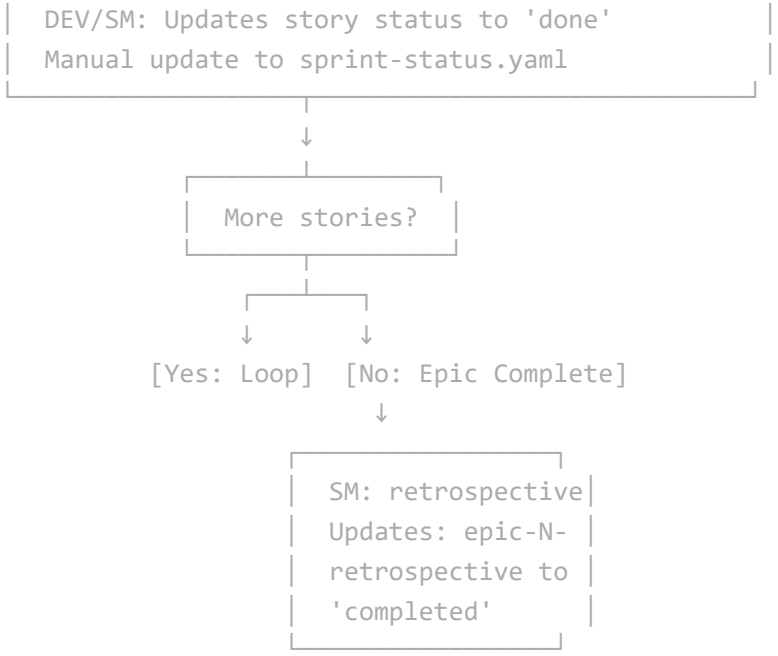
- **drafted**: Story file created (e.g., `stories/1-3-plant-naming.md`)
- **ready-for-dev**: Draft approved + story context created
- **in-progress**: Developer actively working on implementation
- **review**: Under SM review (via code-review workflow)
- **done**: Story completed and deployed

Retrospective Statuses:

- **optional**: Can be done but not required
- **completed**: Retrospective has been completed

The Implementation Loop





Workflow Responsibilities

Workflow	Agent	Purpose	Status Updates
sprint-planning	SM	Initialize sprint status tracking	Creates sprint-status.yaml
epic-tech-context	SM	Create epic-specific technical context	Epic: backlog → contexted
create-story	SM	Draft individual story files	Story: backlog → drafted
story-context	SM	Generate implementation context/XML	Story: drafted → ready-for-dev
dev-story	DEV	Implement story	Story: ready-for-dev → in-progress → review
code-review	SM/SR	Quality validation and feedback	(No automatic state change)
retrospective	SM	Capture epic learnings	Retrospective: optional → completed
correct-course	SM	Handle issues/scope changes	(Adaptive based on situation)

Key Guidelines

1. **Epic Context First**: Epics should be `contexted` before their stories can be `drafted`
2. **Sequential by Default**: Stories are typically worked in order within an epic
3. **Parallel Work Supported**: Multiple stories can be `in-progress` if team capacity allows
4. **Learning Transfer**: SM drafts next story after previous is `done` to incorporate learnings
5. **Flexible Status Updates**: Agents and users can manually update sprint-status.yaml as needed

Greenfield vs Brownfield Paths

Greenfield Projects (New Code)

Path: Phase 1 (optional) → Phase 2 → Phase 3 (Levels 2-4) → Phase 4

- **Level 0-1**: Skip Phase 3, go straight to implementation with tech-spec
- **Level 2-4**: Full solutioning with architecture before implementation
- Clean slate for architectural decisions
- No existing patterns to constrain design

Brownfield Projects (Existing Code)

Path: Documentation Prerequisite (if undocumented) → Phase 1 (optional) → Phase 2 → Phase 3 (Levels 2-4) → Phase 4

Documentation Prerequisite (Conditional):

```
workflow-status/workflow-init
    ├─→ Check: Is existing codebase documented?
    │   ├─→ YES: Proceed to Phase 1 or 2
    │   └─→ NO: REQUIRED - Run document-project workflow first
    │       ├─→ Analyzes existing code
    │       ├─→ Documents current architecture
    │       ├─→ Identifies technical debt
    │       ├─→ Creates comprehensive baseline documentation
    │       └─→ Produces superior docs for LLM + human understanding
    └─→ Continue with scale-adaptive planning (Phases 1-4)
```

Critical for Brownfield:

- Must understand existing patterns before planning
- Integration points need documentation
- Technical debt must be visible in planning
- Constraints from existing system affect scale decisions

Post-Completion Option: After Phase 4 completes, run `document-project` again to create clean source-of-truth documentation that supersedes scattered phase artifacts.

Agent Participation by Phase

Phase/Step	Primary Agents	Supporting Agents	Key Workflows
Prerequisite: Documentation	Analyst	-	document-project (conditional)

Phase/Step	Primary Agents	Supporting Agents	Key Workflows
Phase 1: Analysis	Analyst, Game Designer	PM, Researcher	brainstorm-_, research, _-brief
Phase 2: Planning	PM	UX Designer, Analyst	prd, tech-spec, gdd, narrative
Phase 3: Solutioning	Architect	PM, Tech Lead	create-architecture, solutioning-gate-check
Phase 4: Implementation	SM, DEV	SR (code-review)	sprint-planning, create-story, dev-story
Post-Completion: Documentation	Analyst	-	document-project (optional cleanup)

Key Files and Artifacts

Tracking Documents

- **bmm-workflow-status.md**: Phase and workflow tracking (updated by workflow-status)
  - Current phase and progress
  - Workflow history
  - Next recommended actions
  - Project metadata and configuration
- **sprint-status.yaml**: Implementation tracking (Phase 4 only)
  - All epics, stories, and retrospectives
  - Current status for each item (backlog → done)
  - Single source of truth for Phase 4 progression
  - Updated by agents as work progresses
- **Epics.md**: Master epic/story definitions (source of truth for planning, Level 2-4)

Phase Outputs

- **Documentation Prerequisite (if run)**:
  - Comprehensive project documentation (index.md, architecture.md, source-tree-analysis.md, component-inventory.md, etc.)
  - Superior to manually-maintained docs for LLM understanding
- **Phase 1**:
  - Product briefs, game briefs, research documents

- **Phase 2**:
  - Level 0: tech-spec.md + story-{slug}.md
  - Level 1: tech-spec.md + epic breakdown + story-{slug}-N.md files
  - Level 2-4: PRD.md + epics.md (+ optional ux-design.md, narrative.md)
- **Phase 3**:
  - architecture.md (with ADRs)
  - Validation reports
  - Gate check documentation
- **Phase 4**:
  - sprint-status.yaml (tracking file)
  - epic-N-context.md files (per epic)
  - story-{key}.md files (per story)
  - story-{key}-context.md files (per story)
  - Implemented code and tests

Best Practices

1. Respect the Scale

- Don't create PRDs for Level 0-1 changes (use tech-spec only)
- Don't skip architecture for Level 2-4 projects
- Let the workflow paths determine appropriate artifacts
- Level 2 still requires Phase 3 solutioning (lighter than 3-4)

2. Use Sprint Planning Effectively

- Run sprint-planning at the start of Phase 4
- Context epics before drafting their stories (epic-tech-context)
- Update sprint-status.yaml as work progresses
- Re-run sprint-planning to auto-detect new files/contexts

3. Maintain Flow Integrity

- Stories must be defined in Epics.md before sprint-planning
- Complete epic context before story drafting
- Create story context before implementation
- Each phase completes before the next begins

4. Document Brownfield First (Prerequisite)

- Never plan without understanding existing code

- Run document-project if codebase is undocumented (PREREQUISITE, not Phase 0)
- Technical debt must be visible in planning
- Integration points need documentation
- Can also run post-Phase 4 for superior final documentation

5. Learn Continuously

- Run retrospectives after each epic
- Incorporate learnings into next story drafts
- Update workflows based on team feedback
- Share patterns across teams

Common Pitfalls and Solutions

Pitfall	Solution
Skipping sprint-planning	Always run at Phase 4 start - it creates status file
Creating stories without epic context	Run epic-tech-context before create-story
Skipping story-context generation	Always run after create-story for better dev guidance
Not updating sprint-status.yaml	Update statuses as work progresses
Thinking Level 2 skips Phase 3	Level 2 DOES require architecture (just lighter)
Planning brownfield without docs	Run document-project first if undocumented
Not running retrospectives	Complete after every epic for learning transfer
Manually tracking stories elsewhere	Use sprint-status.yaml as single source of truth

Quick Reference Commands

```
# Universal Entry Point (Start Here!)
bmad analyst workflow-status  # Check status and get recommendations

# Documentation Prerequisite (Brownfield without docs OR post-completion cleanup)
bmad analyst document-project

# Phase 1: Analysis (Optional)
bmad analyst brainstorm-project    # Software ideation
bmad game-designer brainstorm-game # Game ideation
bmad analyst research              # Market/technical research
bmad analyst product-brief         # Software brief
bmad game-designer game-brief      # Game brief

# Phase 2: Planning (Required)
bmad pm prd                  # Level 2-4 software projects
bmad pm tech-spec            # Level 0-1 software projects
bmad pm gdd                  # Game projects (all levels)
```

```
bmad pm narrative            # Game narrative (optional)
bmad ux-designer create-ux-design  # UI-heavy projects

# Phase 3: Solutioning (Levels 2-4)
bmad architect create-architecture    # System architecture
bmad architect validate-architecture  # Validation (optional)
bmad architect solutioning-gate-check # Gate check

# Phase 4: Implementation (Sprint-Based)
bmad sm sprint-planning      # FIRST: Initialize sprint tracking
bmad sm epic-tech-context    # Create epic context (per epic)
bmad sm create-story         # Draft story file
bmad sm story-context        # Create story context
bmad dev dev-story           # Implement story
bmad sm code-review         # Quality validation
# (Update sprint-status.yaml to 'done' manually or via workflow)
bmad sm retrospective        # After epic complete
bmad sm correct-course       # If issues arise
```

Future Enhancements

Coming Soon

- **Automated status updates**: Workflows automatically update sprint-status.yaml
- **Workflow orchestration**: Automatic phase transitions and validation
- **Progress dashboards**: Real-time workflow status visualization
- **Team synchronization**: Multi-developer story coordination

Under Consideration

- AI-assisted retrospectives with pattern detection
- Automated story sizing based on historical data
- Predictive epic planning with risk assessment
- Cross-project learning transfer
- Enhanced brownfield analysis with architectural debt scoring

Version: v6-alpha **Last Updated**: 2025-10-26

This document serves as the authoritative guide to BMM v6a workflow execution. For detailed information about individual workflows, see their respective README files in the workflow folders.

Related Documentation

- **Workflow Paths**: See `workflow-status/paths/` for detailed greenfield/brownfield routing by level
- **Phase 2 Planning**: See `2-plan-workflows/README.md` for scale-adaptive planning details

- **Phase 4 Sprint Planning**: See `4-implementation/sprint-planning/README.md` for sprint status system
- **Individual Workflows**: Each workflow directory contains its own README with specific instructions