🚀 AI Dev Flow

Universal Specification-Driven Development Framework

Version 2.2 | Production Ready

📋 Executive Overview

> **Mission:** Enable AI-assisted software development across any project domain through structured, traceable requirements and specifications—transforming business needs into production-ready code through a systematic, auditable workflow.

16

Artifact Layers

48×

Code Gen Speedup

100%

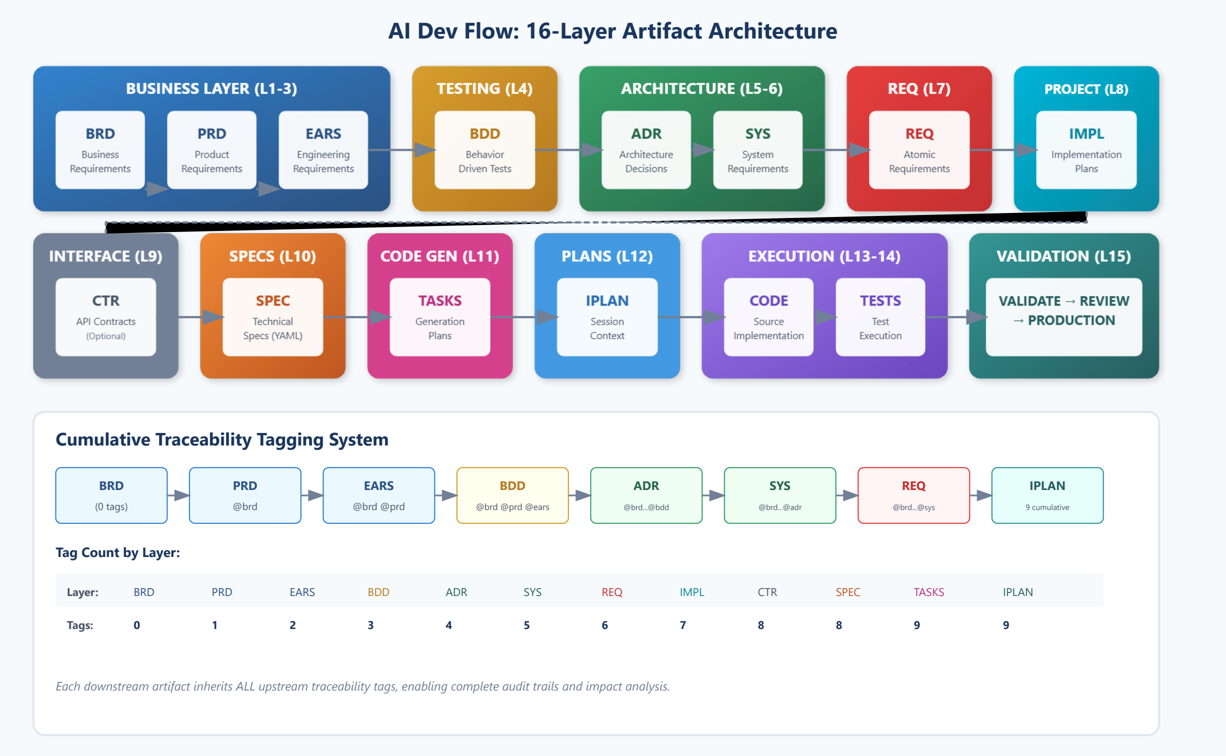
Traceability Coverage

2

Development Tracks

🏗 16-Layer Architecture

The framework organizes documentation into 16 hierarchical layers, each building upon the previous with cumulative traceability tagging.



AI Dev Flow: 16-Layer Artifact Architecture

BUSINESS LAYER (L1-3)
BRD Business Requirements
PRD Product Requirements
EARS Engineering Requirements

TESTING (L4)
BDD Behavior Driven Tests

ARCHITECTURE (L5-6)
ADR Architecture Decisions
SYS System Requirements

REQ (L7)
REQ Atomic Requirements

PROJECT (L8)
IMPL Implementation Plans

INTERFACE (L9)
CTR API Contracts (Optional)

SPECS (L10)
SPEC Technical Specs (YAML)

CODE GEN (L11)
TASKS Generation Plans

PLANS (L12)
IPLAN Session Context

EXECUTION (L13-14)
CODE Source Implementation
TESTS Test Execution

VALIDATION (L15)
VALIDATE → REVIEW → PRODUCTION

Cumulative Traceability Tagging System

BRD (0 tags) → PRD @brd → EARS @brd @prd → BDD @brd @prd @ears → ADR @brd...@bdd → SYS @brd...@adr → REQ @brd...@sys → IPLAN 9 cumulative

Tag Count by Layer:

Layer:	BRD	PRD	EARS	BDD	ADR	SYS	REQ	IMPL	CTR	SPEC	TASKS	IPLAN
Tags:	0	1	2	3	4	5	6	7	8	8	9	9

Each downstream artifact inherits ALL upstream traceability tags, enabling complete audit trails and impact analysis.

Layer	Artifact	Purpose	Optional
0	Strategy	External business strategy, vision documents	Pre-artifact
1-3	BRD → PRD → EARS	Business → Product → Engineering requirements	Required
4	BDD	Behavior-driven test scenarios (Gherkin)	Required
5-6	ADR → SYS	Architecture decisions → System requirements	Required

Layer	Artifact	Purpose	Optional
7	REQ	Atomic, testable requirements	Required
8	IMPL	Implementation plans (WHO/WHEN)	Optional
9	CTR	API Contracts (dual-file: .md + .yaml)	Optional
10	SPEC	Technical specifications (YAML)	Required
11	TASKS / ICON	Task breakdown / Implementation contracts	ICON optional
12	IPLAN	Session-specific execution plans	Required
13-14	Code → Tests	Source implementation → Test execution	Required
15	Validation	Quality gates → Review → Production	Required

⚡ Development Tracks

Choose the right track based on your project needs:

Development Track Comparison

🚀 MVP TRACK

Time to Code:	**1-2 Days**
Templates:	Single-file MVP templates
File Structure:	Flat files (no splitting)
Validation:	Relaxed MVP profile

Best For:

- Startups & prototypes
- Small teams (2-10 people)
- Rapid iteration cycles
- Early-stage validation

Key Command:

```
mvp_autopilot.py --auto-fix --mvp-validators
```

VS

🏢 FULL FRAMEWORK

Time to Code:	**1-2 Weeks**
Templates:	Full multi-section templates
File Structure:	Nested (document splitting)
Validation:	Strict compliance checks

Best For:

- Enterprise projects
- Regulated industries
- Large teams (10+ people)
- Audit requirements

Key Command:

```
validate_all.py --all --strict
```

✅ Validation System

Automated validation ensures consistency and compliance across all artifacts.

Validation Error Code Categories

SEC	Section File Validation
XREF	Cross-Reference Accuracy
IDPAT	ID Pattern Consistency
DIAG	Diagram Consistency
TERM	Terminology Consistency
TZ / DATE	Timezone & Date Validation
ELEM	Element Code Validation
COUNT	Count Validation
FWDREF	Forward Reference Prevention
XDOC	Cross-Document Traceability

Exit Code Conventions

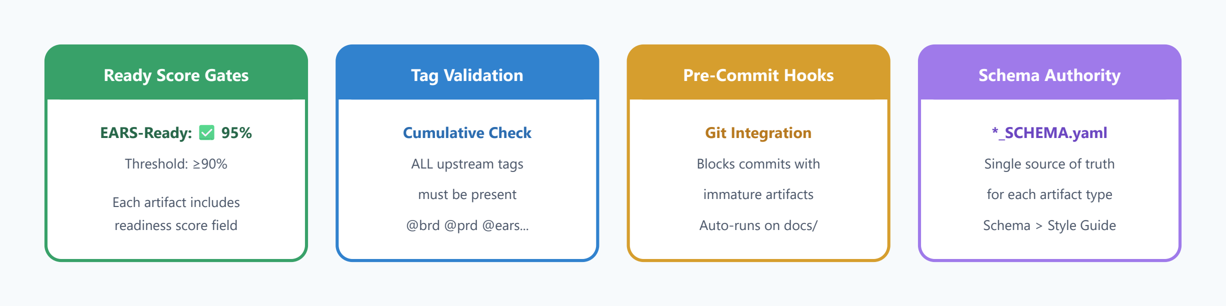
0	**PASS**	No errors or warnings
1	**WARNING**	Continue with notification
2	**ERROR**	Fail pipeline

Key Validators

Validator	Purpose	Command
validate_all.py	Master orchestrator for all validators	`--all --report markdown`
validate_links.py	Traceability link integrity	`--docs-dir ai_dev_flow`
validate_cross_document.py	Cross-document references	`--all --auto-fix`
mvp_autopilot.py	MVP workflow automation	`--auto-fix --mvp-validators`

🚦 Quality Gates

Automated gates prevent progression to downstream layers until artifacts meet maturity thresholds.



Ready Score Gates

EARS-Ready: ✅ 95%

Threshold: ≥90%

Each artifact includes readiness score field

Tag Validation

Cumulative Check

ALL upstream tags must be present

@brd @prd @ears...

Pre-Commit Hooks

Git Integration

Blocks commits with immature artifacts

Auto-runs on docs/

Schema Authority

***_SCHEMA.yaml**

Single source of truth for each artifact type

Schema > Style Guide

🏷️ ID Naming Standards

Dual ID Notation System

Document Reference (Dash)

Format: `TYPE-NN`

Purpose: References complete document

Examples:

```
ADR-33 → ADR-33_risk_limit.md
SPEC-01 → SPEC-01_api.yaml
CTR-05 → CTR-05_data.md/.yaml
```

Tag Usage:

```
@adr: ADR-33, @spec: SPEC-01
```

Element Reference (Dot)

Format: `TYPE.NN.TT.SS`

Purpose: References specific element

Components:

TYPE = Document type (BRD, PRD...)
NN = Document number
TT = Element type (01=Func Req...)
SS = Sequential number

Examples:

```
BRD.07.01.01 → Func Req #1
```

💎 Key Benefits

For Organizations

- ✓ Complete audit trails from business to code
- ✓ Regulatory compliance ready (FDA, ISO, SOC2)
- ✓ Instant impact analysis for changes
- ✓ Reduced documentation drift
- ✓ Standardized processes across teams

For Development Teams

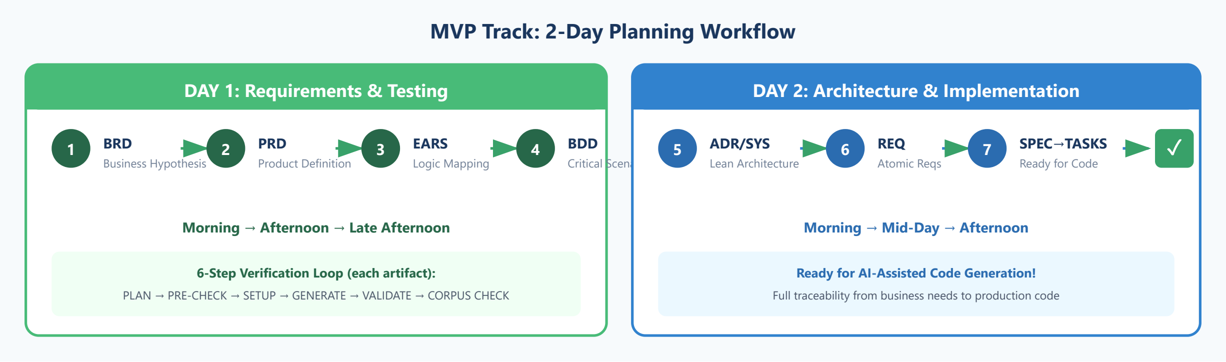
- ✓ 48× faster AI code generation
- ✓ AI-optimized YAML specifications

✓ Automated validation prevents errors

✓ Clear traceability for debugging

✓ Two tracks: MVP speed or full rigor

📋 MVP Workflow: 7-Step Process



MVP Track: 2-Day Planning Workflow

DAY 1: Requirements & Testing

1 BRD Business Hypothesis → 2 PRD Product Definition → 3 EARS Logic Mapping → 4 BDD Critical Scen

Morning → Afternoon → Late Afternoon

6-Step Verification Loop (each artifact):

PLAN → PRE-CHECK → SETUP → GENERATE → VALIDATE → CORPUS CHECK

DAY 2: Architecture & Implementation

5 ADR/SYS Lean Architecture → 6 REQ Atomic Reqs → 7 SPEC→TASKS Ready for Code → ✓

Morning → Mid-Day → Afternoon

Ready for AI-Assisted Code Generation!

Full traceability from business needs to production code

⚙️ Threshold Management

> **Threshold Definition Strategy:** Thresholds are defined in source documents (BRD/PRD/ADR) and referenced via `@threshold:` tags in downstream artifacts—eliminating separate registry documents while maintaining full traceability.

Threshold Categories

Category	Examples	Defined In
quota	User limits, resource caps, tier allowances	BRD, PRD
rate	API rate limits, request throttling, burst limits	BRD, PRD
perf	P95/P99 latency, throughput targets, SLA metrics	ADR
timeout	API timeouts, session expiry, cache TTL	ADR
circuit	Failure counts, reset timeouts, pool sizes	ADR
alert	Warning thresholds, critical limits, escalation triggers	ADR

Tag Format

```
@threshold: {DOC_TYPE}.{DOC_NUM}.{threshold_key}
```

Examples:

```
@threshold: PRD.01.quota.user.daily
@threshold: PRD.01.rate.api.standard
@threshold: ADR.15.circuit.failure.count
@threshold: ADR.15.perf.api.p95
```

📐 Document Structure Rules

File Organization

Structure	When to Use
Flat (Monolithic)	Single file <25KB, MVP templates
Nested (Sections)	Documents >25KB, complex structure

MVP Note: MVP track uses flat files only—document splitting rules don't apply.

CTR Dual-File Format

> API Contracts require **both files**:
>
> `CTR-NN_slug.md` — Human-readable context
> `CTR-NN_slug.yaml` — Machine-readable schema
>
> Slugs must match exactly between files

🚀 Getting Started

1 **Choose Your Track**
MVP Track for rapid prototyping (1-2 days) or Full Framework for enterprise projects (1-2 weeks)

2 **Set Up Templates**
Copy templates to your project: `cp -r ai_dev_flow/ <your_project>/docs/`

3 **Run MVP Autopilot (for MVP track)**

```
python3 scripts/mvp_autopilot.py --root ai_dev_flow --intent "My MVP" --
auto-fix
```

4 Validate Continuously

```
python3 scripts/validate_all.py ai_dev_flow --all --report markdown
```

5 Generate Code

Use SPEC/TASKS artifacts as deterministic input for AI code generation with full traceability

📚 Framework Documents Summary

Document	Purpose	Key Contents
README.md	Framework overview	Architecture, workflow, directory structure
TRACEABILITY.md	Traceability guidelines	Tag formats, cumulative hierarchy, validation
ID_NAMING_STANDARDS.md	Naming conventions	Document/element IDs, file patterns
VALIDATION_STANDARDS.md	Validation system	Error codes, validators, CI/CD integration

Document	Purpose	Key Contents
MVP_WORKFLOW_GUIDE.md	MVP development	7-step workflow, verification loop
MVP_AUTOPILOT.md	Automation guide	Commands, auto-fix, CI integration
THRESHOLD_NAMING_RULES.md	Threshold management	Categories, naming, environment overrides
SPEC_DRIVEN_DEVELOPMENT_GUIDE.md	Complete methodology	Principles, quality gates, best practices

AI Dev Flow Framework v2.2 | Production Ready | Last Updated: November 2025

Enabling AI-assisted software development through structured, traceable specifications