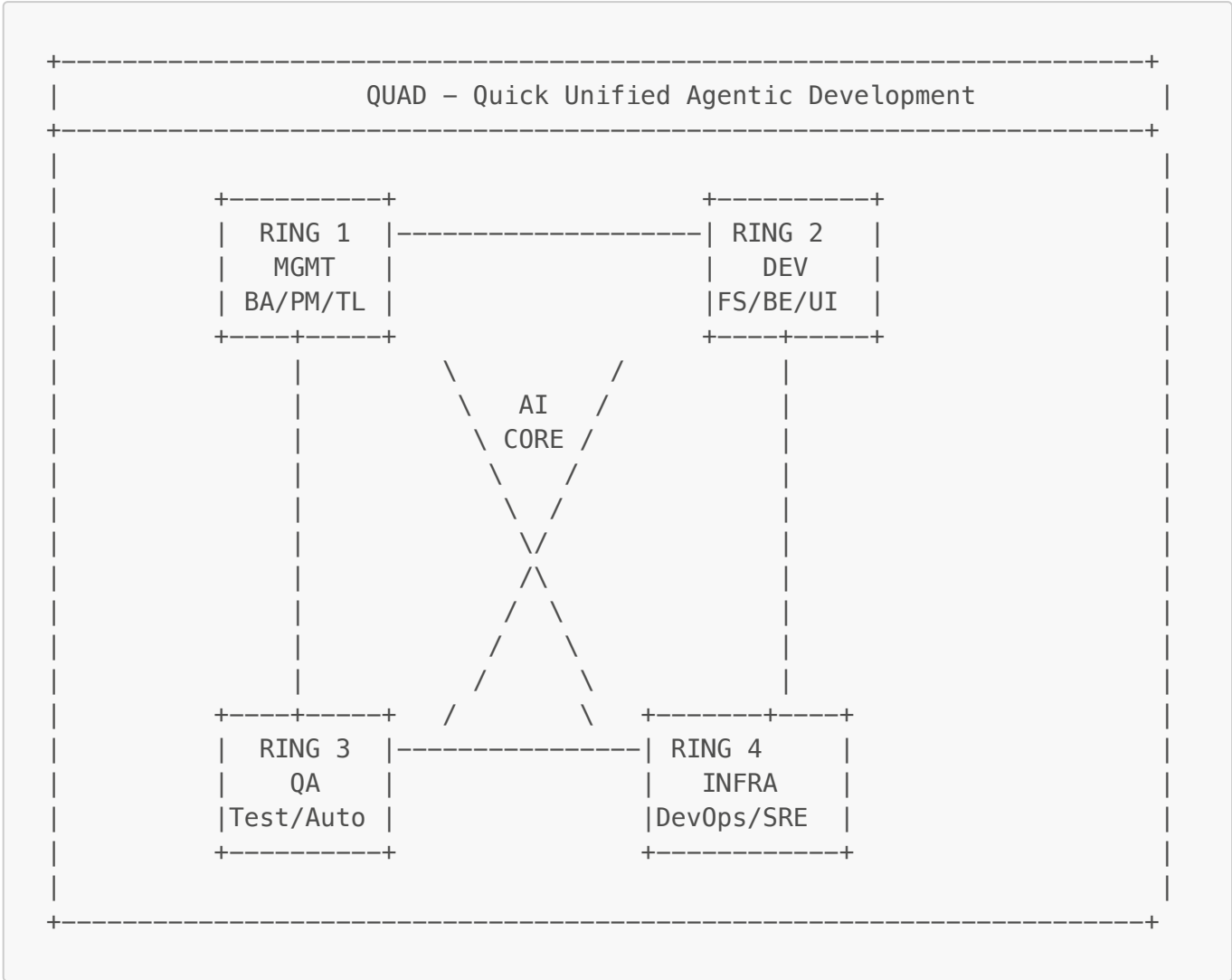
QUAD™ - Quick Unified Agentic Development



> **QUAD™** = 4 Rings + AI Agents + Docs-First Development

Table of Contents

What is QUAD?

QUAD (Quick Unified Agentic Development) is a modern software development methodology designed for the AI era. It combines **4 functional rings** with **AI agents at every step** and a **documentation-first approach**.

The Name

Letter	Meaning
Q	Quick - Faster development with AI assistance
U	Unified - 4 rings working together seamlessly
A	Agentic - AI agents at every step
D	Development - Software development methodology

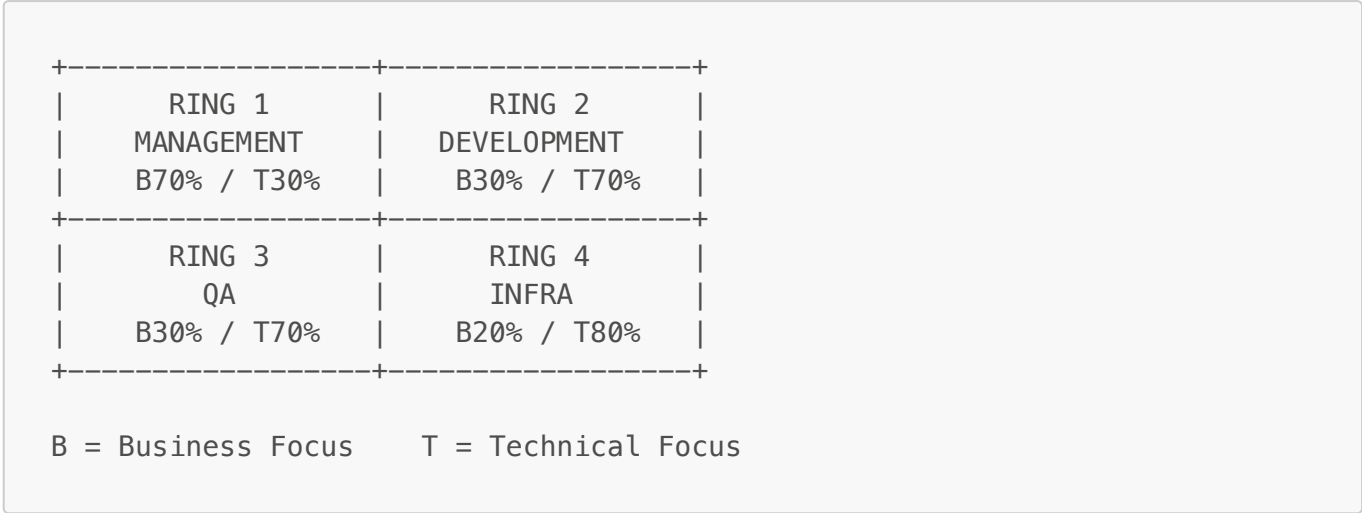
Core Principles

Principle	Description
4 Rings	Management, Development, QA, Infrastructure
AI Agents	Every ring has AI agent helpers
Docs-First	Documentation before and with code
One Source of Truth	Git-versioned documentation
Continuous Flow	Work flows through pipeline, monthly checkpoints
Human Approval	AI assists, humans decide

Key Differentiators

vs Traditional	QUAD Approach
Scrum Master	AI Scheduling Agent
Sprint pressure	Continuous flow with monthly checkpoints
Scattered docs	Flow documentation (UI + API + DB + Tests)
"I don't know what to test"	Test cases embedded in flow docs
Knowledge silos	One source of truth
8-hour days, 5 days	Potential 4-day work week

The 4 Rings



Ring 1: Management (Business 70% / Technical 30%)

Role	Responsibilities
Business Analyst	Requirements, user stories, acceptance criteria
Project Manager	Timelines, stakeholders, resource allocation
Tech Lead	Technical decisions, architecture oversight, code standards

AI Agents: Story Agent, Scheduling Agent, Documentation Agent, Estimation Agent

Ring 2: Development (Business 30% / Technical 70%)

Role	Responsibilities
Full Stack Developer	End-to-end feature development
Backend Developer	API services, business logic
UI Developer	Frontend interfaces, user experience
Mobile Developer	iOS, Android, Flutter/React Native

AI Agents: Dev Agent (UI), Dev Agent (API), Code Review Agent, Refactor Agent

Ring 3: QA (Business 30% / Technical 70%)

Role	Responsibilities
QA Engineer	Test planning, manual testing, exploratory testing
Automation Engineer	Test automation frameworks, scripts
Performance Tester	Load testing, performance optimization
Security Tester	Security testing, vulnerability assessment

AI Agents: UI Test Agent, API Test Agent, Performance Agent, Test Generator Agent

Ring 4: Infrastructure (Business 20% / Technical 80%)

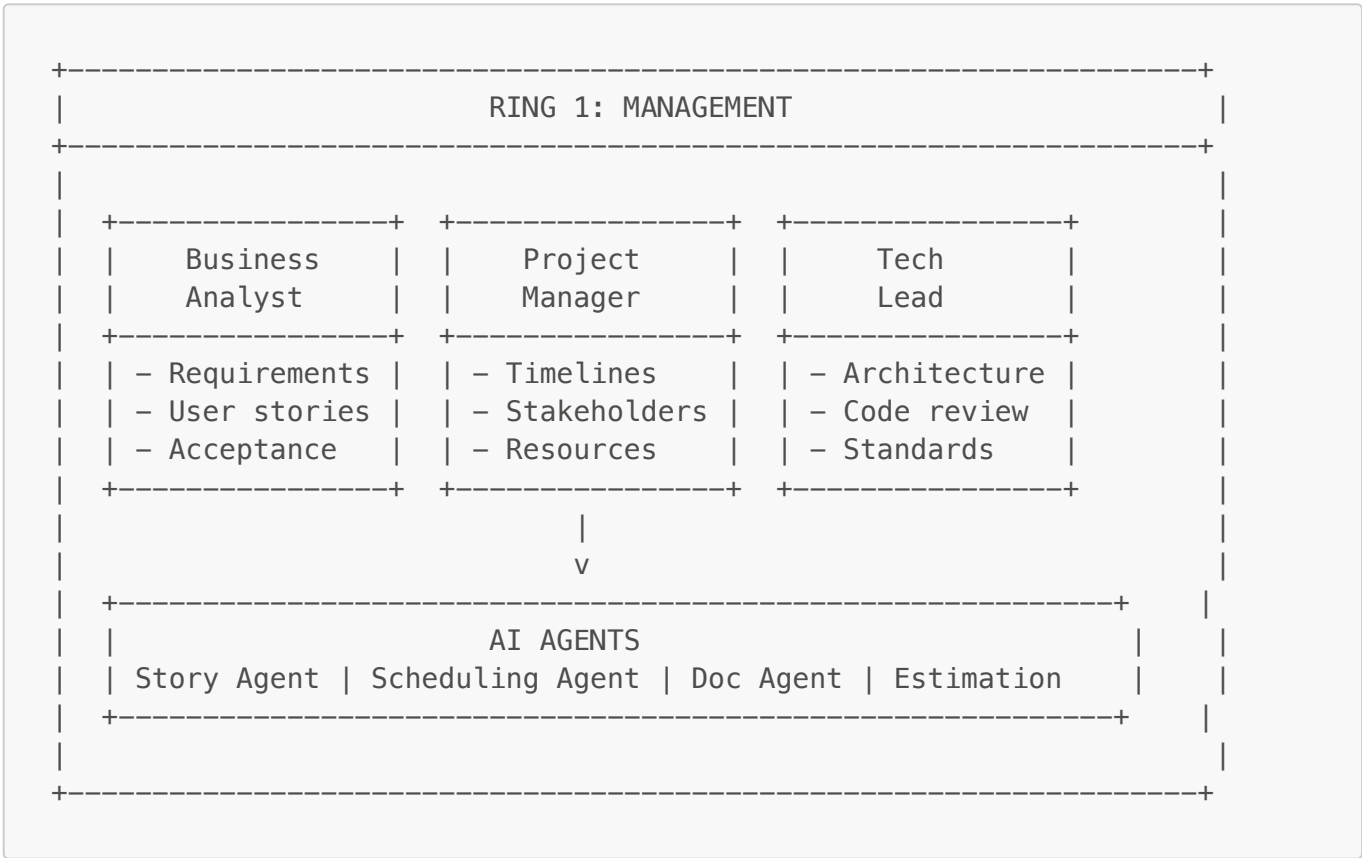
Role	Responsibilities
DevOps Engineer	CI/CD pipelines, deployments
SRE	Reliability, monitoring, incident response
Cloud Engineer	Cloud infrastructure, networking
DBA	Database administration, optimization

AI Agents: Deploy Agent (DEV/QA/PROD), Monitoring Agent, Incident Agent, Cost Agent

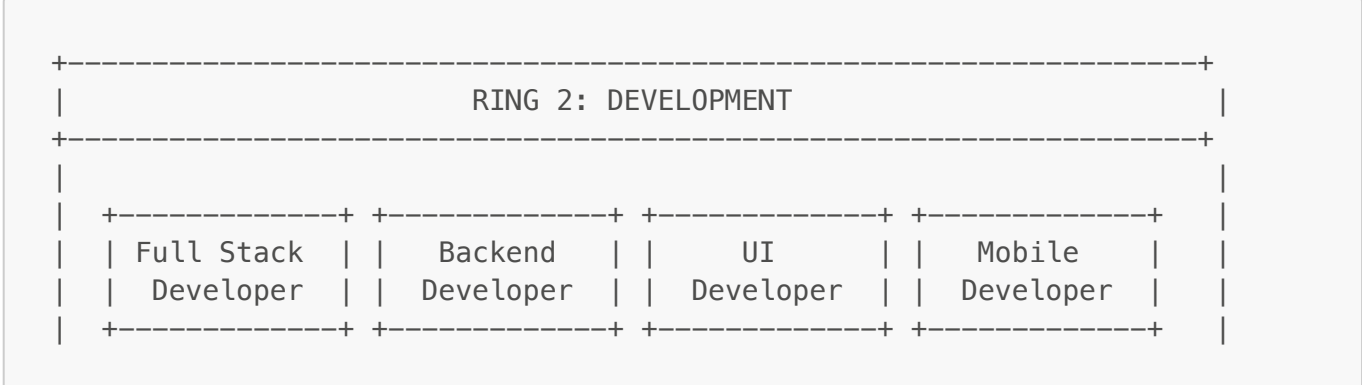
---

Roles Within Each Ring

Ring 1: Management



Ring 2: Development



```
|  | End-to-end  | | API/Services| | Frontend/UX | | iOS/Android |   |
|  +-------------+ +-------------+ +-------------+ +-------------+   |
|                             |                                       |
|                             v                                       |
|  +-------------------------------------------------------------+    |
|  |                      AI AGENTS                              |    |
|  | Dev Agent (UI) | Dev Agent (API) | Code Review | Refactor  |    |
|  +-------------------------------------------------------------+    |
|                                                                     |
+---------------------------------------------------------------------+
```

Ring 3: QA

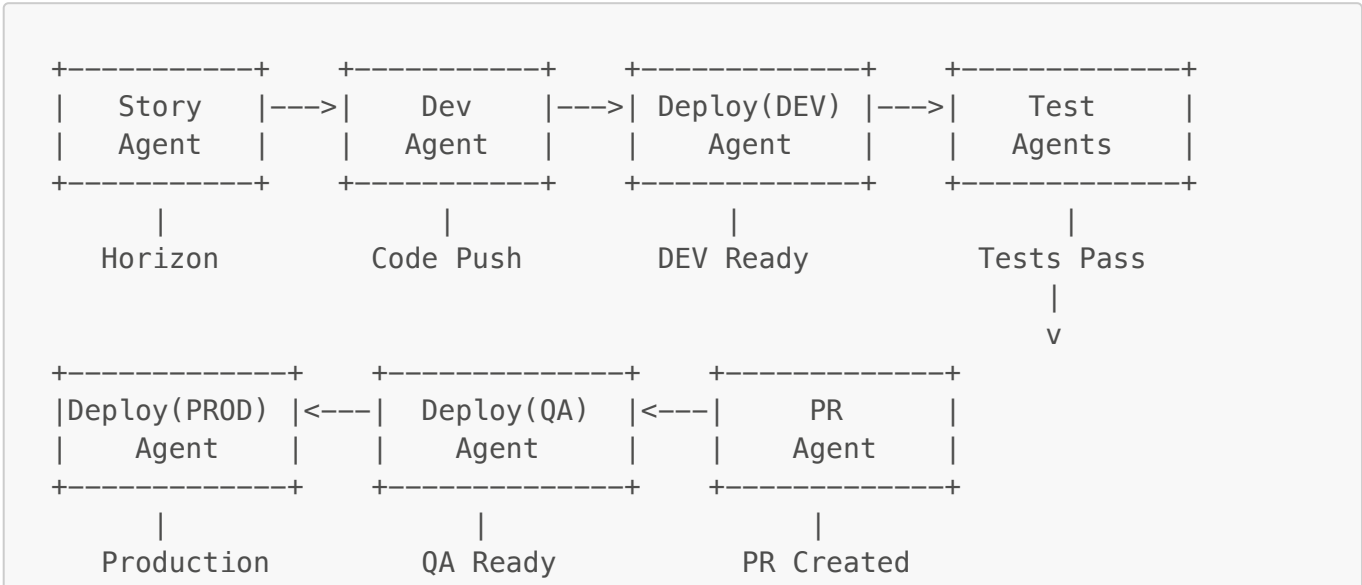
```
+---------------------------------------------------------------------+
|                          RING 3: QA                                 |
+---------------------------------------------------------------------+
|                                                                     |
|  +-------------+ +-------------+ +-------------+ +-------------+   |
|  |     QA      | | Automation  | | Performance | |  Security   |   |
|  |  Engineer   | |  Engineer   | |   Tester    | |   Tester    |   |
|  +-------------+ +-------------+ +-------------+ +-------------+   |
|  | Manual/Expl | | Frameworks  | | Load tests  | | Vuln assess |   |
|  +-------------+ +-------------+ +-------------+ +-------------+   |
|                             |                                       |
|                             v                                       |
|  +-------------------------------------------------------------+    |
|  |                      AI AGENTS                              |    |
|  | UI Test | API Test | Performance | Test Generator | Triage |    |
|  +-------------------------------------------------------------+    |
|                                                                     |
+---------------------------------------------------------------------+
```

Ring 4: Infrastructure

```
+---------------------------------------------------------------------+
|                      RING 4: INFRASTRUCTURE                         |
+---------------------------------------------------------------------+
|                                                                     |
|  +-------------+ +-------------+ +-------------+ +-------------+   |
|  |   DevOps    | |     SRE     | |    Cloud    | |     DBA     |   |
|  |  Engineer   | |             | |  Engineer   | |             |   |
|  +-------------+ +-------------+ +-------------+ +-------------+   |
|  | CI/CD/Deploy| | Reliability | | Infra/Cloud | | DB Optimize |   |
|  +-------------+ +-------------+ +-------------+ +-------------+   |
|                             |                                       |
|                             v                                       |
|  +-------------------------------------------------------------+    |
|  |                      AI AGENTS                              |    |
|  | Deploy (DEV/QA/PROD) | Monitoring | Incident | Cost Agent  |    |
```

AI Agents

Agent Pipeline



Agents by Ring

Ring 1: Management Agents

Agent	Trigger	What It Does	Replaces
Story Agent	BA writes requirement	Enhances with specs, acceptance criteria, edge cases, test cases	Junior BA work
Scheduling Agent	Meeting needed	Analyzes calendars, finds optimal time, tracks action items	Scrum Master
Documentation Agent	Feature complete	Auto-generates flow docs, updates wiki, links artifacts	Manual doc updates
Estimation Agent	Story ready	Suggests complexity based on historical data	Planning poker

Ring 2: Development Agents

Agent	Trigger	What It Does	Replaces
Dev Agent (UI)	Story assigned	Scaffolds UI components, generates platform-specific code	Boilerplate coding

Agent	Trigger	What It Does	Replaces
Dev Agent (API)	Story assigned	Generates controllers, services, DTOs, entities	Boilerplate coding
Code Review Agent	PR created	Pre-reviews for patterns, security, style	First-pass review
Refactor Agent	Code smell detected	Suggests improvements, removes duplication	Tech debt discovery

Ring 3: QA Agents

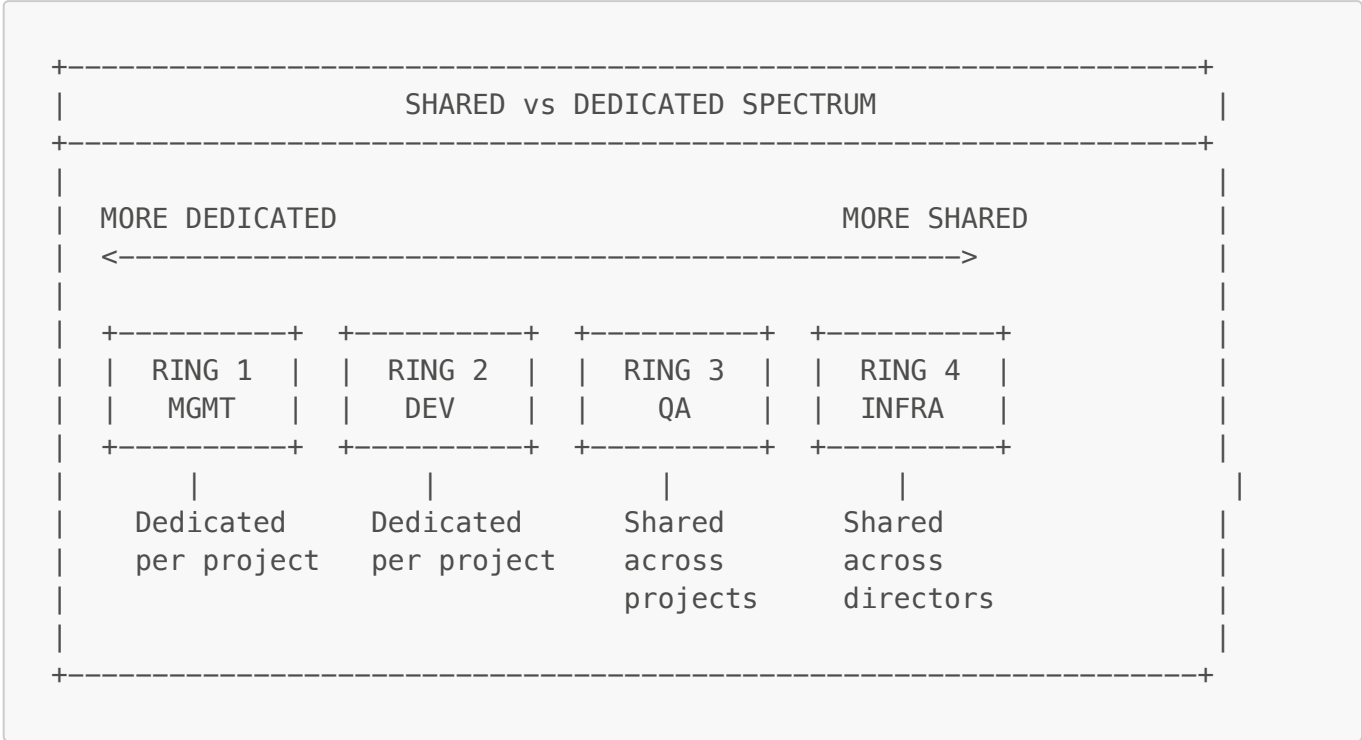
Agent	Trigger	What It Does	Replaces
UI Test Agent	DEV deployed	Runs Playwright/XCTest automation	Manual UI testing
API Test Agent	DEV deployed	Runs REST API test suites	Manual API testing
Performance Agent	QA deployed	Runs load tests, identifies bottlenecks	Manual perf testing
Test Generator Agent	New code pushed	Generates test cases from code and flow docs	Writing test cases
Bug Triage Agent	Bug reported	Categorizes, assigns severity, suggests owner	Manual triage

Ring 4: Infrastructure Agents

Agent	Trigger	What It Does	Replaces
Deploy Agent (DEV)	PR merged to develop	Builds and deploys to DEV	Manual deployment
Deploy Agent (QA)	Tests pass + PR approved	Deploys to QA environment	Manual deployment
Deploy Agent (PROD)	QA approved	Deploys to production with rollback ready	Manual deployment
Monitoring Agent	Always running	Watches logs, metrics, alerts on anomalies	Manual monitoring
Incident Agent	Alert triggered	Creates ticket, pages on-call, suggests runbook	Manual incident mgmt
Cost Agent	Daily/Weekly	Analyzes cloud spend, suggests optimizations	Manual cost review

Shared vs Dedicated Resources

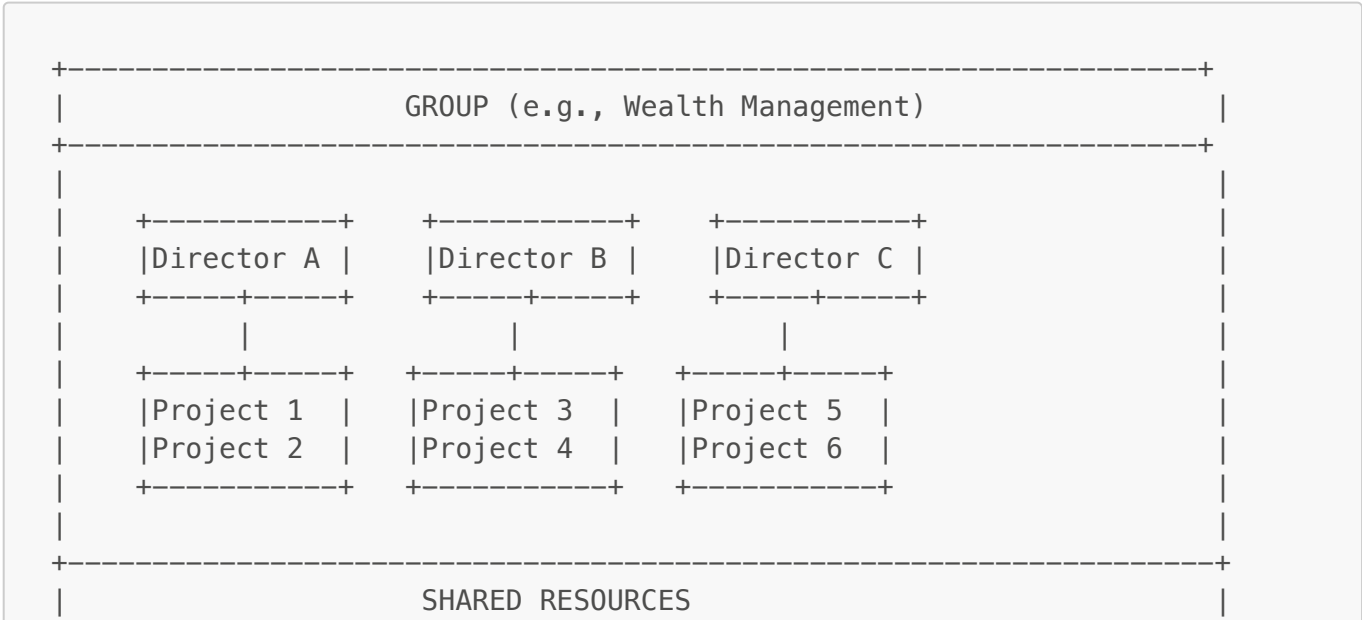
QUAD allows flexible resource allocation based on project needs:



Default Allocation

Ring	Default Mode	Can Be Changed?
Ring 1 (Management)	More Dedicated	Yes - can share BA/PM across small projects
Ring 2 (Development)	More Dedicated	Yes - devs can help other projects
Ring 3 (QA)	More Shared	Yes - can dedicate if project has enough work
Ring 4 (Infrastructure)	Purely Shared	Typically shared across all directors

Organizational Structure



```
|  +-------------+  +-------------+  +-------------------+           |
|  |   Ring 4    |  |   Ring 3    |  | Enabling Teams    |           |
|  |   (Infra)   |  |  (QA Pool)  |  | (Architect, etc.) |           |
|  |   ALWAYS    |  |   USUALLY   |  |     OPTIONAL      |           |
|  |   SHARED    |  |   SHARED    |  |                   |           |
|  +-------------+  +-------------+  +-------------------+           |
+-----------------------------------------------------------------+
```

QUAD Hierarchy & Rules

Rule Hierarchy

```
+---------------------------------------+
|     BASE RULES (QUAD Platform)        |  <-- Universal do's and don'ts
+---------------------------------------+
|     COMPANY-WIDE RESTRICTIONS         |  <-- Entity/Org level policies
+---------------------------------------+
|     DIRECTOR RESTRICTIONS             |  <-- Program/Portfolio level
+---------------------------------------+
|     TECH LEAD RESTRICTIONS            |  <-- Team level
+---------------------------------------+
|     TEAM MEMBER CUSTOMIZATION         |  <-- Personal preferences
+---------------------------------------+

RULE: Each level can ADD restrictions, never CONTRADICT upper levels
```

Example Rules

Level	Example Rule
Base Rules	"Never commit secrets to Git"
Company-wide	"All PRs require 2 reviewers"
Director	"Use Java 21 only for this program"
Tech Lead	"Follow our naming conventions"
Team Member	"I prefer dark mode in IDE"

What Can Be Customized

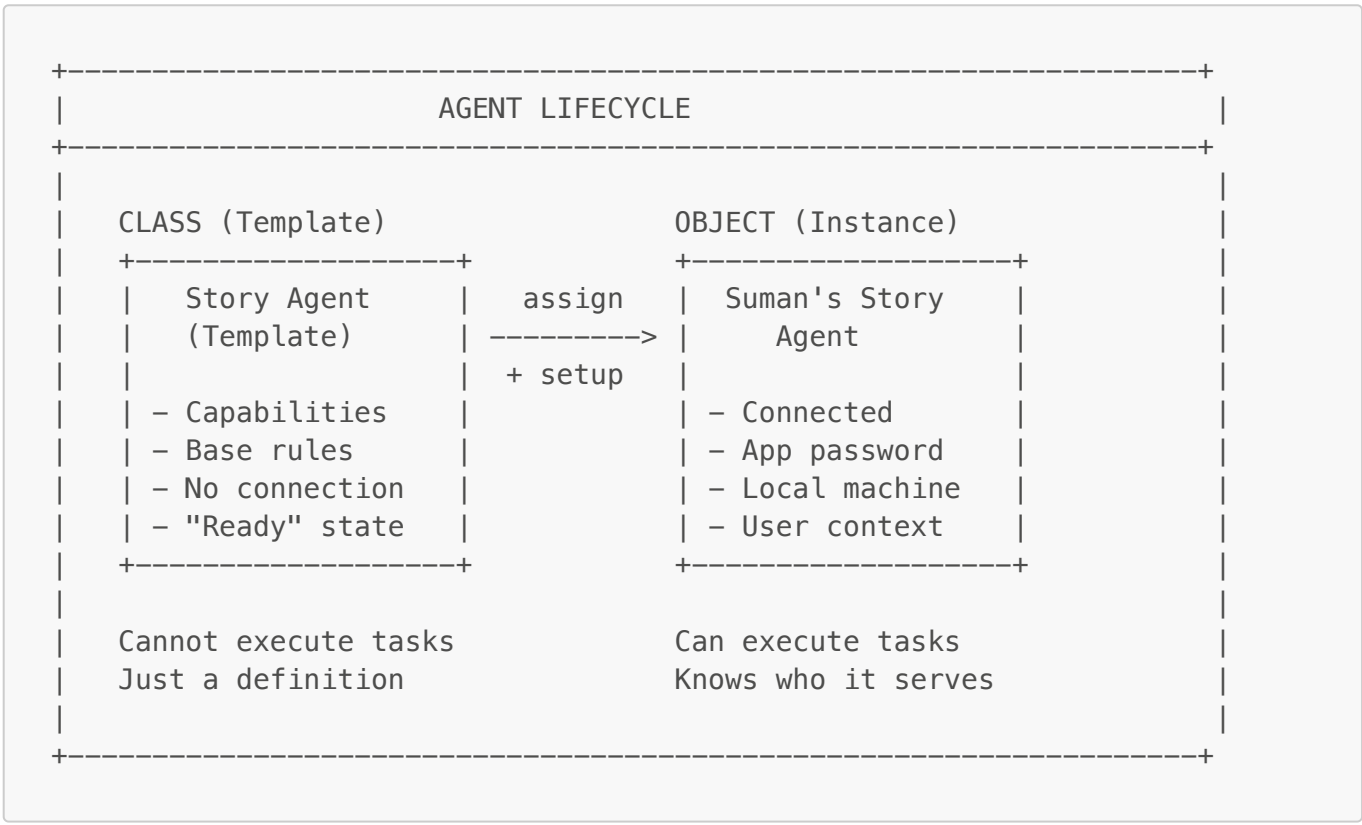
Setting	Levels That Can Set
Agent on/off	All levels
Approval requirements	Company → Director → TL
Code standards	Company → Director → TL

Setting	Levels That Can Set
Tool choices	Director → TL
Personal preferences	Team Member
Security rules	Company only

Agent Class-Object Pattern

QUAD agents follow a class-object instantiation pattern:

How It Works



Setup Flow

Step	What Happens	How
1	Agent Template exists	QUAD Platform provides
2	User joins team	Gets web app access
3	App Password created	Not SSO - restricted permissions
4	Local machine setup	Agent installed locally
5	Agent = Object	Connected, contextualized, active

Key Points

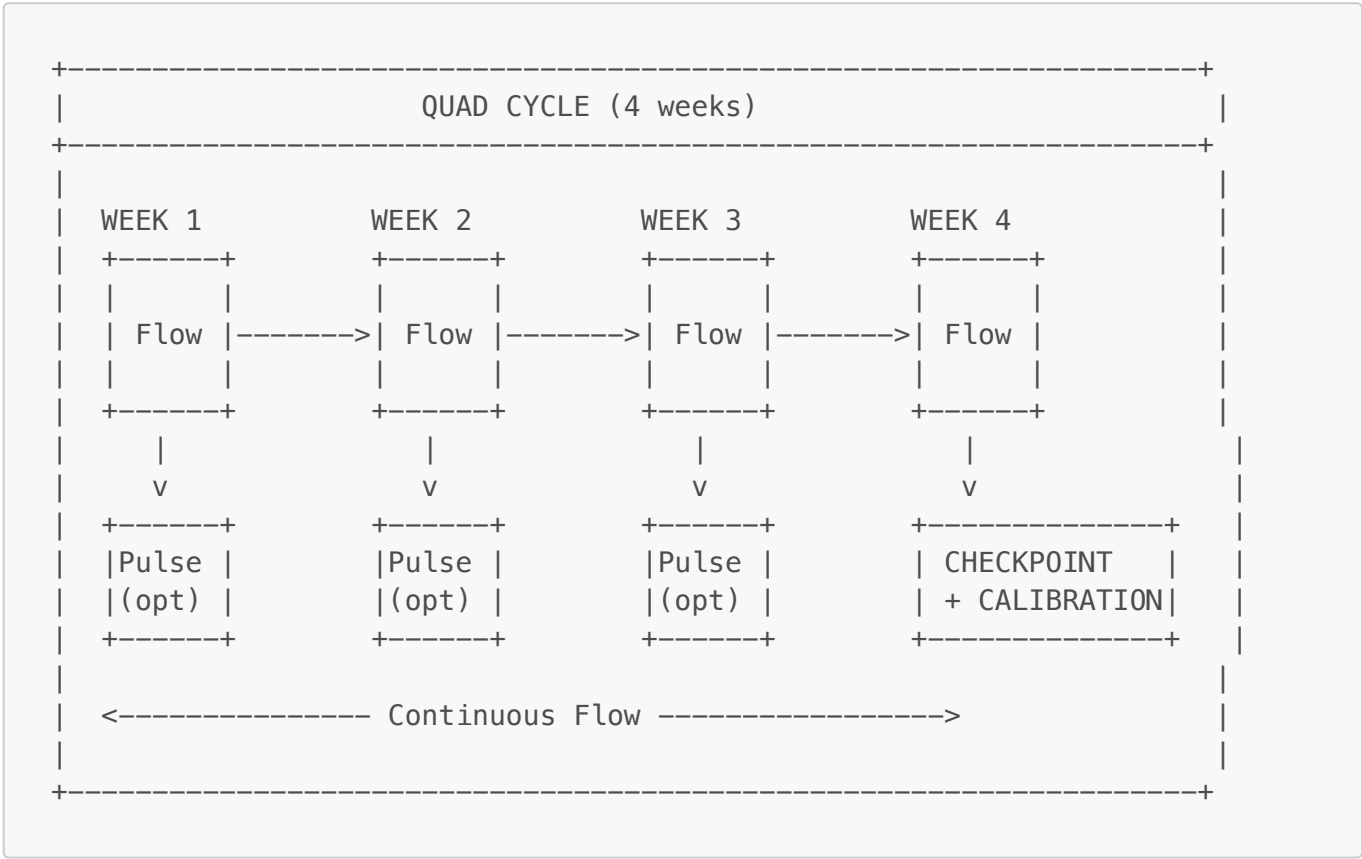
- Agent uses **App Password**, not user's SSO

- App Password has **restricted permissions** per hierarchy
- Agent runs on **user's local machine**
- Each user has their **own agent instance**
- Templates may have **readonly access** in future

QUAD Cycle (Milestone Flow)

QUAD uses a hybrid approach: **Monthly Checkpoints** with **Continuous Flow** within.

Cycle Structure



Cycle Events

Event	Frequency	Duration	Purpose
Pulse	Weekly (optional)	5-15 min	Quick sync, blockers, AI dashboard review
Checkpoint	Monthly	1 hour	Demo to stakeholders, release decision
Calibration	Monthly	30-60 min	Retrospective, agent learning review
Trajectory	Monthly	1-2 hours	Planning priorities for next cycle

Why Monthly Checkpoints?

Reason	Benefit
AI works continuously	No artificial 2-week boundaries

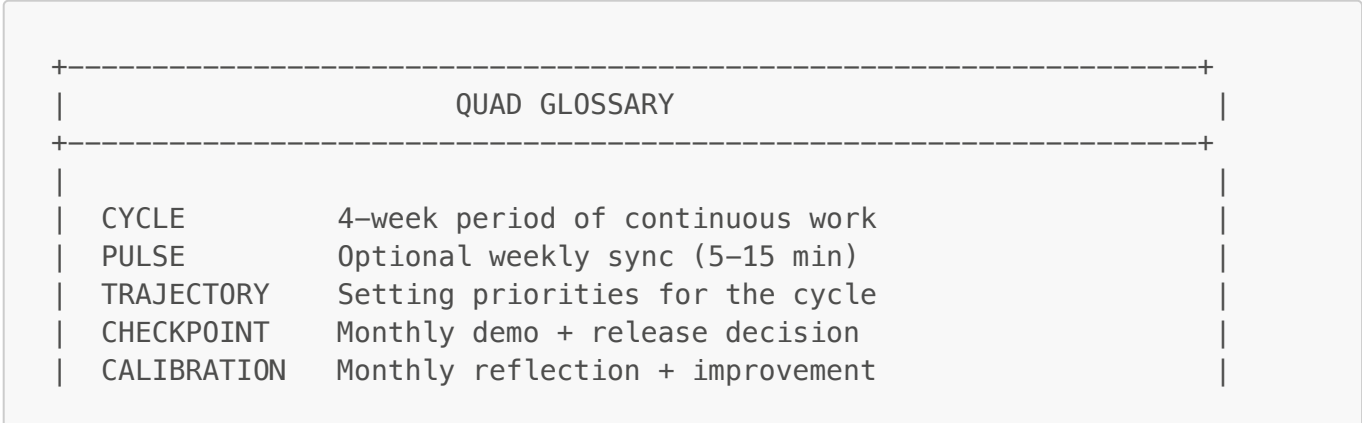
Reason	Benefit
Humans need rhythm	Monthly demo keeps stakeholders engaged
Business needs dates	"Q1 cycle" easier than "whenever ready"
Less pressure	No sprint-end crunch
4-day week fits	No Friday cramming

QUAD Terminology

QUAD uses new terms instead of Scrum jargon:

Old Term	QUAD Term	Meaning
Sprint	**Cycle**	4-week period of continuous work
Daily Standup	**Pulse**	Optional weekly sync (5-15 min)
Sprint Planning	**Trajectory**	Setting priorities for the cycle
Sprint Review	**Checkpoint**	Monthly demo + release decision
Retrospective	**Calibration**	Monthly reflection + improvement
Backlog	**Horizon**	Work ahead, prioritized queue
Backlog Grooming	**Refinement**	Breaking down and clarifying work
Sprint Backlog	**Cycle Queue**	Work selected for current cycle
Story Points	**Complexity**	Estimation using chosen method
Velocity	**Flow Rate**	Speed of work through pipeline
Burndown	**Progression**	Visual of work remaining
Definition of Done	**Completion Criteria**	When work is truly done
Scrum Master	**Scheduling Agent**	AI handles coordination
Product Owner	**Ring 1 Lead**	BA/PM/TL in management ring

QUAD Glossary Card



```
|  HORIZON       Work backlog (what's ahead)                          |
|  REFINEMENT    Breaking down and clarifying work                    |
|  CYCLE QUEUE   Work selected for current cycle                      |
|  FLOW RATE     Speed of work through pipeline                       |
|  PROGRESSION   Visual of work remaining                             |
|  COMPLETION    Definition of done criteria                          |
|  IMPEDIMENT    Blocker that needs resolution                        |
|                                                                     |
|  RINGS         4 functional teams (Mgmt, Dev, QA, Infra)           |
|  AGENTS        AI helpers assigned to each ring                     |
|  ENABLING      Optional support teams (Architect, Security)         |
|                                                                     |
+---------------------------------------------------------------------+
```

Estimation Methods

QUAD offers multiple estimation methods. Teams choose in settings:

Option 1: Platonic Solids (Default)

Shape	Faces	Complexity	Equivalent
Tetrahedron	4	Trivial/Simple	1-2
Cube	6	Moderate	3-5
Octahedron	8	Medium	5-8
Dodecahedron	12	Complex	8-13
Icosahedron	20	Very Complex	13-21

Option 2: Dimensions

Shape	Dimensions	Complexity
Point	0D	Trivial (config change)
Line	1D	Simple (one file)
Triangle	2D	Moderate (few files)
Square	2D+	Medium (multiple components)
Cube	3D	Complex (cross-system)
Tesseract	4D	Very Complex (architecture)

Option 3: Powers

Notation	Value	Complexity
2^0	1	Trivial

Notation	Value	Complexity
2^1	2	Simple
2^2	4	Moderate
2^3	8	Complex
2^4	16	Very Complex

Option 4: Classic Fibonacci

Points	Complexity
1	Trivial
2	Simple
3	Moderate
5	Medium
8	Complex
13	Very Complex

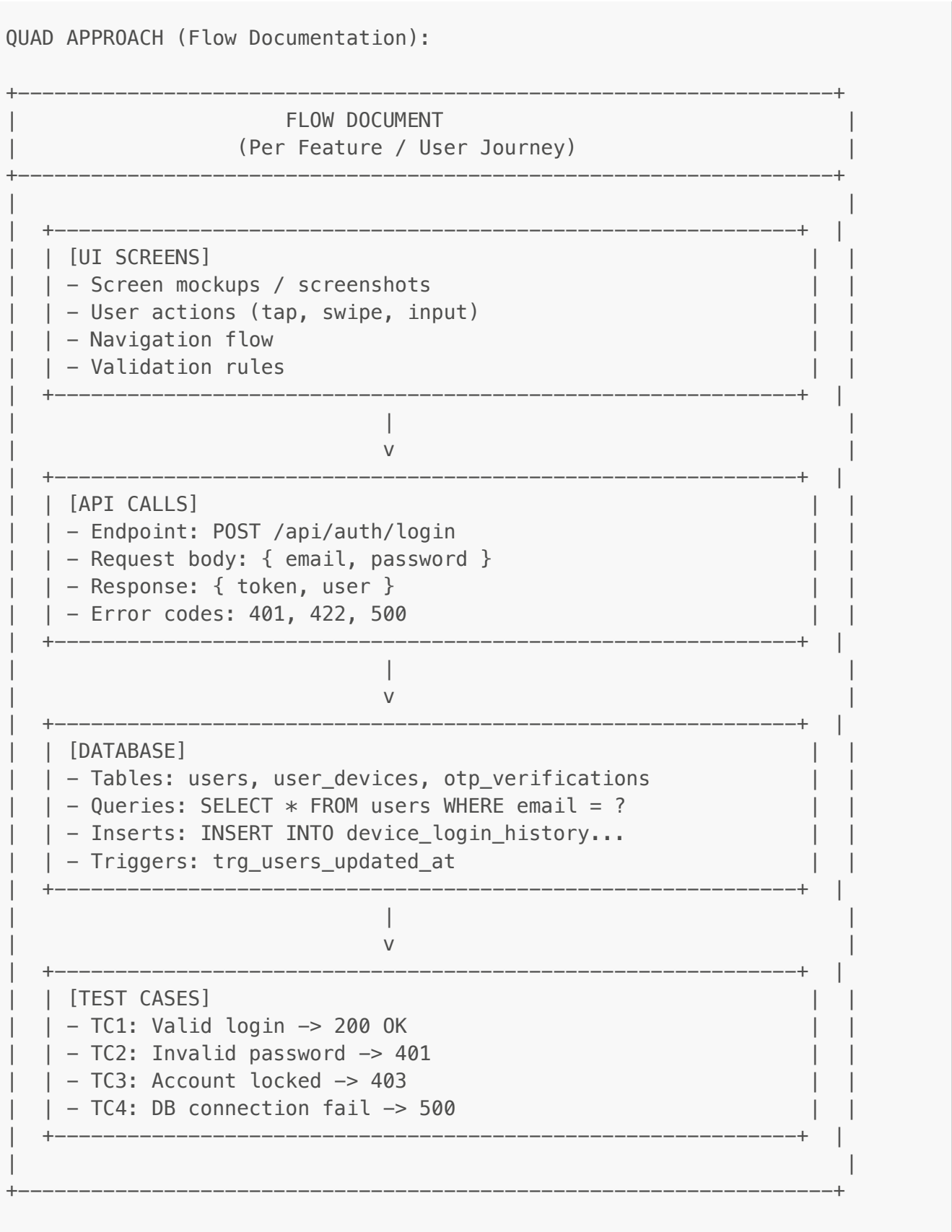
Docs-First Approach

QUAD is built on documentation-first development. Documentation is not an afterthought - it's created before and with code.

The Problem QUAD Solves

```
TRADITIONAL APPROACH (Scattered Documentation):

+----------+    +----------+    +----------+    +----------+
|    UI    |    |   API    |    |    DB    |    |   Test   |
|   Specs  |    |   Docs   |    |  Schema  |    |  Cases   |
|  (Figma) |    | (Swagger)|    |  (Wiki)  |    | (Excel)  |
+----------+    +----------+    +----------+    +----------+
      |              |              |              |
      +-------+------+------+------+
              |
         ALL SEPARATE!
         Not linked
         Gets stale
         QA: "What should I test?"
         Dev: "Where's the spec?"
```

QUAD Solution: One Source of Truth



Benefits

Role	How They Use Flow Docs	Benefit
Ring 1	Creates flow, defines acceptance	Clear requirements
Ring 2	Sees exact API spec, DB queries	No guessing

Role	How They Use Flow Docs	Benefit
Ring 3	Test cases already defined	"I know what to test"
Ring 4	Sees DB tables, plans performance	Proactive optimization

Flow Documentation

Flow Document Structure

```
+-----------------------------------------------------------------+
|                    FLOW DOCUMENT TEMPLATE                       |
+-----------------------------------------------------------------+

# FLOW: [Feature Name]

## Overview
| Field        | Value                    |
|--------------|--------------------------|
| Flow ID      | FLOW_XXX_001             |
| Owner        | Ring 1 (BA/PM/TL)        |
| Last Updated | YYYY-MM-DD               |
| Status       | Draft / Active / Retired |

---

## Step 1: [Action Name]

### UI
| Element      | Type       | Validation           |
|--------------|------------|----------------------|
| Field name   | TextInput  | Required, format     |
| Button       | Button     | Disabled until valid |

### Screenshot
[Include screenshot or mockup]

### API
    POST /api/endpoint
    Content-Type: application/json

    Request:
    {
      "field": "value"
    }

    Response (200):
    {
      "result": "data"
    }

    Response (4xx/5xx):
```

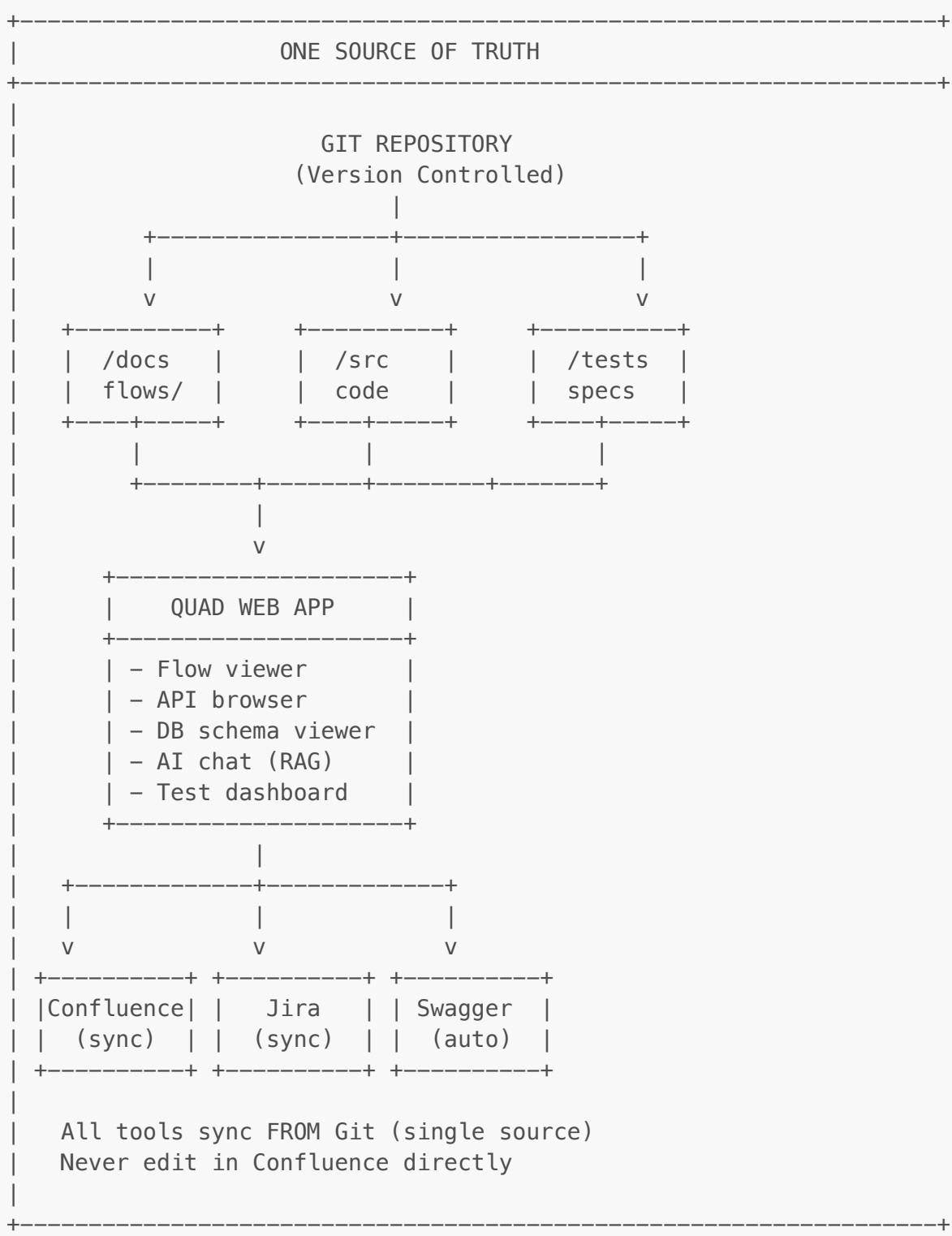
```
    {
      "error": "message",
      "code": "ERROR_CODE"
    }

### Database
    -- Query executed
    SELECT columns FROM table WHERE condition;

    -- Insert/Update
    INSERT INTO table (columns) VALUES (values);

### Test Cases
| TC ID | Scenario        | Input       | Expected | API | DB Check   |
|-------|-----------------|-------------|----------|-----|------------|
| TC001 | Happy path      | valid input | 200      | Yes | row added |
| TC002 | Invalid input   | bad input   | 422      | No  | no change |

---

## Step 2: [Next Action]
...

+---------------------------------------------------------------------+
```

QA Testing View (Auto-Generated from Flow)

```
+---------------------------------------------------------------------+
|                      QA TESTING VIEW                                |
+---------------------------------------------------------------------+
|                                                                     |
|  Flow: FLOW_LOGIN_001                                               |
|                                                                     |
|  Test Checklist (auto-generated from flow):                         |
|                                                                     |
|  UI Tests:                                                          |
|  [ ] Email field shows keyboard                                     |
|  [ ] Password field masks input                                     |
|  [ ] Login button disabled when empty                               |
|  [ ] Error message displays on failure                              |
|                                                                     |
|  API Tests:                                                         |
|  [ ] POST /api/auth/login returns 200 with valid creds              |
|  [ ] Returns 401 with invalid password                              |
|  [ ] Returns 422 with malformed email                               |
|  [ ] Response contains token and user object                        |
|                                                                     |
|  Database Tests:                                                    |
|  [ ] users.last_login_at updated on success                         |
|  [ ] device_login_history row created                               |
|  [ ] login_attempts incremented on failure                          |
```

```
|  [ ] No data changed on validation failure                          |
|                                                                     |
|  Edge Cases:                                                        |
|  [ ] SQL injection attempt blocked                                  |
|  [ ] Account lockout after 5 failures                               |
|  [ ] Concurrent login from 2 devices                                |
|                                                                     |
+---------------------------------------------------------------------+
```

One Source of Truth Architecture



Productivity Benefits

Metric	Without QUAD Docs	With QUAD Docs	Improvement
Onboarding time	2-4 weeks	3-5 days	75% faster
"Where is X?" questions	10+/day	Near zero	90% reduction
QA test case creation	2 hours/feature	Auto-generated	80% faster
Knowledge transfer	Meetings + shadowing	Read the flow	70% faster
Bug reproduction	"Works on my machine"	Check flow + DB	Much faster
New feature estimation	Guessing	Similar flows exist	More accurate

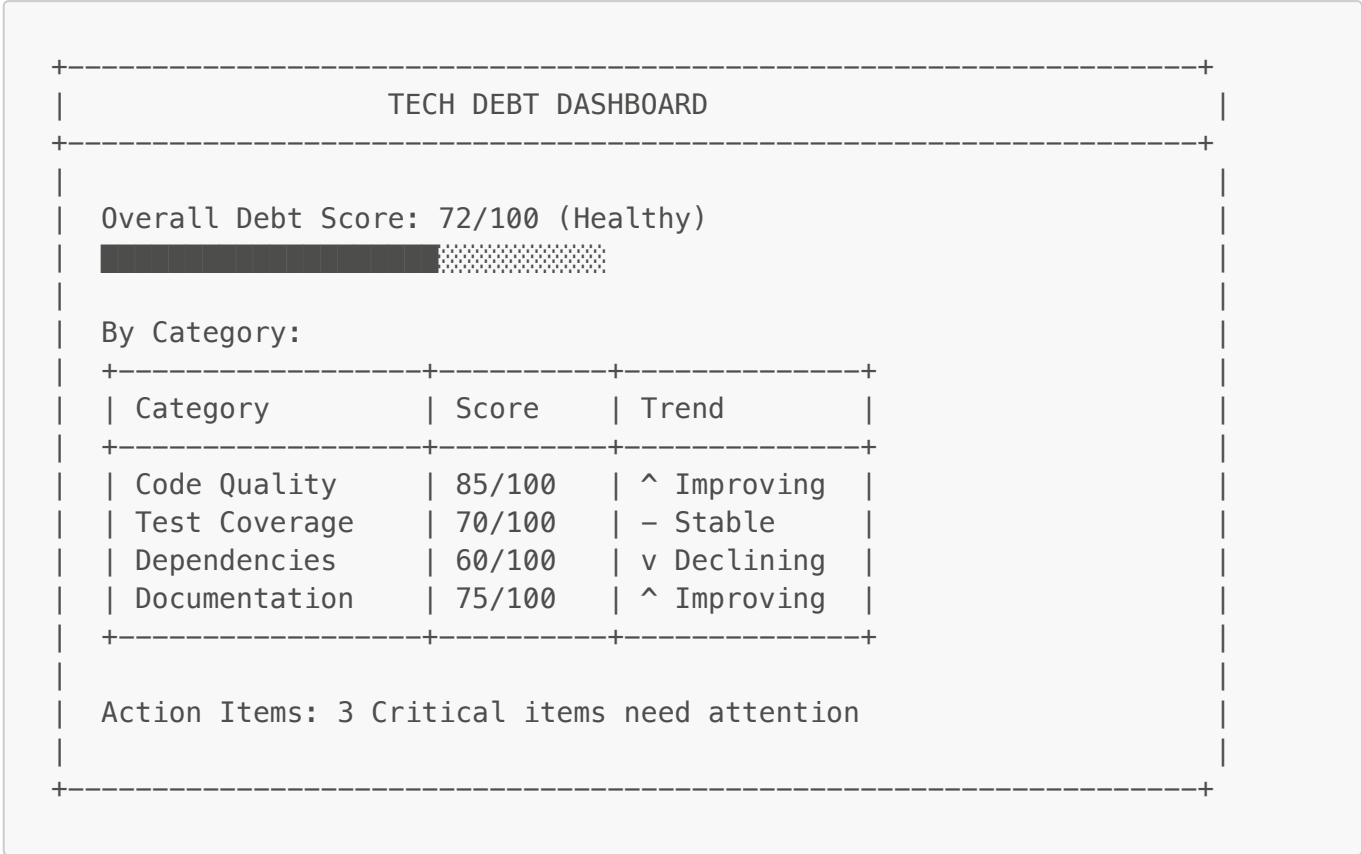
Technical Debt Handling

QUAD handles technical debt continuously, not in batches:

Approach: Continuous + Visibility

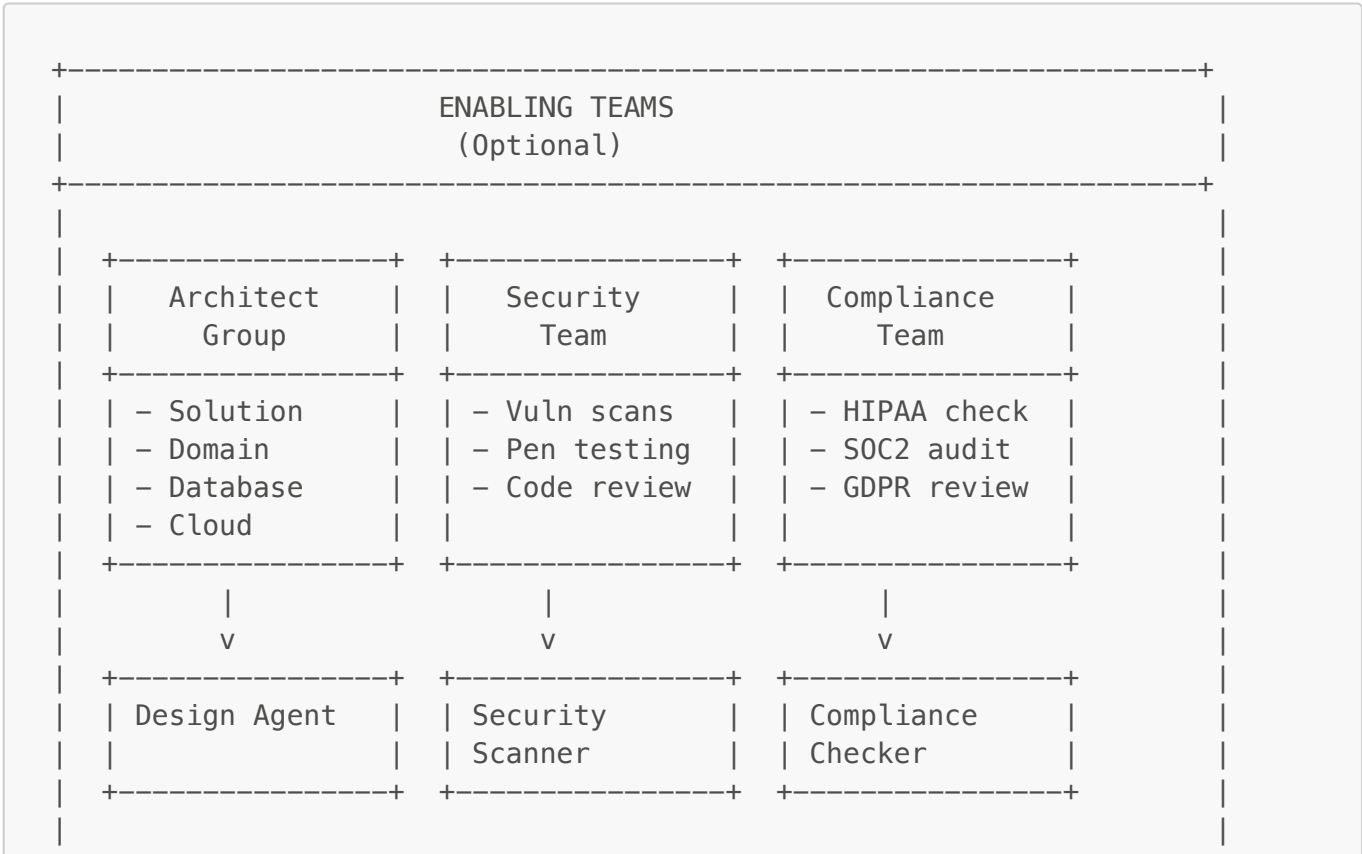
```
+-------------------------------------------------------------------+
|                  TECHNICAL DEBT APPROACH                          |
+-------------------------------------------------------------------+
|                                                                   |
|  1. REFACTOR AGENT (Continuous Scanning)                          |
|     - Scans for code duplication                                  |
|     - Identifies long methods (>50 lines)                         |
|     - Flags high cyclomatic complexity                            |
|     - Checks outdated dependencies                                |
|     - Detects security vulnerabilities                            |
|     - Finds missing tests                                         |
|     - Creates tickets automatically                               |
|                                                                   |
|  2. BOY SCOUT RULE                                                |
|     "Leave code better than you found it"                         |
|     - When dev touches a file, fix small issues                   |
|     - Include quick improvements with feature work                |
|     - No separate "tech debt stories" for small items             |
|                                                                   |
|  3. DEBT DASHBOARD (Visibility)                                   |
|     - Overall debt score visible to all                           |
|     - Trends: improving, stable, declining                        |
|     - Alerts when score drops below threshold                     |
|                                                                   |
|  4. DEBT CEILING                                                  |
|     - If score < 60, Director is alerted                          |
|     - May pause features until debt reduced                       |
|                                                                   |
+-------------------------------------------------------------------+
```

Tech Debt Dashboard



Enabling Teams

Enabling Teams are optional support groups that are NOT counted as QUAD rings:



```
+-------------------------------------------------------------------+

QUAD = 4 Rings + AI Core + Optional Enabling Teams
```

When to Use Enabling Teams

Team	When Needed
Architect Group	Large systems, complex integrations, shared databases
Security Team	Regulated industries, handling sensitive data
Compliance Team	Healthcare (HIPAA), Finance (SOC2), EU (GDPR)

Methodology Comparison

QUAD vs Other Methodologies

Aspect	Waterfall	Agile/Scrum	DevOps	QUAD
Planning	All upfront	Sprint-based	Continuous	AI-assisted
Cycle	Phases	2-week sprints	Continuous	Monthly checkpoint
Roles	Phase silos	Cross-functional	Dev + Ops	4 Rings + AI
Documentation	Heavy upfront	Light/skipped	Code-as-docs	Docs-First
Testing	End phase	Per sprint	Automated	AI Test Agents
Meetings	Phase gates	Daily standup	As needed	Pulse (optional)
Feedback	Months	2-4 weeks	Days	Real-time
AI Integration	None	Optional	Optional	Core

When to Use QUAD

Project Type	QUAD Fit	Notes
Greenfield (new product)	Excellent	Full QUAD from start
Brownfield (legacy)	Moderate	Gradual adoption
Maintenance/Support	Good	Ring 3 + 4 heavy
Regulated (Healthcare)	Good	More approval gates
Startup (MVP)	Excellent	Small team + AI = fast

QUAD Benefits Summary

Benefit	How QUAD Achieves It

Benefit	How QUAD Achieves It
Faster delivery	AI handles repetitive tasks
Better quality	Docs-First, automated testing
Less meetings	AI dashboard replaces standups
Knowledge sharing	One source of truth
Employee retention	Less stress, potential 4-day week
No single dependency	Everything documented
Easy transitions	People can move between projects

Getting Started

Minimum Viable QUAD

Start small, add agents gradually:

Week	Enable	Learn
1-2	Story Agent only	AI-enhanced requirements
3-4	+ Dev Agent (one platform)	AI code scaffolding
5-6	+ Deploy Agent (DEV only)	Automated deployments
7-8	+ Test Agents	Automated testing
9+	+ Full pipeline	End-to-end automation

Team Setup Checklist

Step	Action	Owner
1	Assign 4 Ring roles to team	Director
2	Set up docs-first repository	Ring 4
3	Configure Story Agent	Ring 1
4	Configure Dev Agents	Ring 2
5	Set up CI/CD with Deploy Agents	Ring 4
6	Configure Test Agents	Ring 3
7	Create team customization rules	Tech Lead
8	Train team on QUAD concepts	All

Project Lifecycle (12-Month Example)

Phase	Months	Rings Active	Focus
Phase 1	1-3	Ring 1 + Ring 4	Requirements + Infra setup
Phase 2	4-6	Ring 1 + Ring 2 + Ring 4	Development + Environments
Phase 3	7-9	Ring 2 + Ring 3 + Ring 4	Dev + QA + CI/CD
Phase 4	10-12	Ring 3 + Ring 4	Testing + Production

License & Attribution

Copyright

QUAD™ (Quick Unified Agentic Development)

Trademark Notice

QUAD™ and **Quick Unified Agentic Development™** are trademarks of A2 Vibe Creators LLC.

The 4-ring model, AI agent pipeline, and Docs-First approach are original works of Suman Addanke.

License

This methodology documentation is licensed under **Creative Commons Attribution 4.0 International (CC BY 4.0)**.

You May	Condition
Share	Copy and redistribute in any medium
Adapt	Remix, transform, build upon
Commercial	Use for commercial purposes

Required Attribution:

> "QUAD Methodology by Suman Addanke / A2 Vibe Creators"

Contact

Channel	Link
Website	https://a2vibecreators.com
GitHub	https://github.com/a2vibecreators
Author	Suman Addanke

Version History

Version	Date	Changes
2.0	Dec 2025	Complete rewrite: 4 Rings, Hierarchy, Agent Pattern, Cycle, Terminology, Docs-First
1.2	Dec 2025	Added methodology comparison, flow documentation
1.1	Dec 2025	Added practical guide
1.0	Dec 2025	Initial QUAD methodology

QUAD™ - A methodology by A2 Vibe Creators | First Published: December 2025