

System Architecture Documentation Guide

Technical Documentation Standards

Emmanuel BRUNO

Université de Toulon, LIS UMR CNRS 7020

emmanuel.bruno@univ-tln.fr

2026-10-14

Table des matières

1	Introduction	1
1.1	Documentation Structure	1
1.2	Architecture Views	2
2	Architecture in Agile Development	2
2.1	Purpose & Goals	2
2.2	Quick Start Guide	2
2.3	Documentation Types & When to Use	3
2.3.a	Essential Documents	3
2.4	Documentation Hierarchy	3
2.5	Document Dependencies	3
2.6	Maintenance Schedule	3
2.7	Documentation Automation	3
2.8	Review Process	4
2.9	Agile Documentation Principles	4
2.10	Template Usage in Agile	4
2.11	Architecture Documentation	5
2.11.a	Documentation Structure	5
2.11.b	Core Templates	5
2.12	Recommended Tools	5
2.13	Architecture Metrics	5
2.13.a	Key Performance Indicators	5
2.14	Troubleshooting Guide	6
2.14.a	Common Architecture Issues	6
2.15	Documentation Flow	6
3	Conclusion	6

1 Introduction

Effective system architecture documentation is crucial for successful software development, especially in agile environments. This guide provides practical approaches to create and maintain architecture documentation that balances comprehensive technical information with agile principles of simplicity and just-in-time delivery.

Whether you're an architect, tech lead, or developer, this guide will help you create documentation that adds value without becoming a maintenance burden. The focus is on « documentation that matters » - artifacts that support decision-making, onboarding, and technical governance while avoiding unnecessary overhead.

1.1 Documentation Structure

Core Documents:

- **System Context:** System boundaries and interfaces
- **Component Architecture:** Service and module design
- **Integration Architecture:** System interactions
- **Infrastructure Architecture:** Deployment and operations

Support Documents:

- Technical Guides: Implementation details
- API Specifications: Interface contracts
- Data Models: Storage and flow

1.2 Architecture Views

```
graph TD
    A[System Context] --> B[Component Architecture]
    A[System Context] --> C[Integration Architecture]
    B --> D[Technical Implementation]
    C --> D[Technical Implementation]
    D --> E[Infrastructure Architecture]
```

2 Architecture in Agile Development

This guide provides a practical, lightweight approach to architecture decisions and documentation in an agile context, focusing on delivering value while maintaining technical excellence.

2.1 Purpose & Goals

- Enable fast, informed technical decisions
- Maintain system clarity as it evolves
- Balance agility with architectural integrity
- Support distributed team collaboration

2.2 Quick Start Guide

1. Identify Architecture Decision Points

- New epic starting
- Technical pivot needed
- Scaling challenge
- Integration requirement
- Risk mitigation

2. Agile Decision Process

```
graph TD
    A[Identify Need] --> B[Quick Research]
    B --> C[Team Discussion]
    C --> D[Prototype/Spike]
    D --> E[Document Decision]
    E --> F[Implement & Learn]
    F --> G[Refine/Adapt]
    G --> C
```

3. Enhanced Decision Template

Architecture Decision:

Title: [Brief, descriptive name]

ID: [ARCH-{number}]

Status: [Proposed|Accepted|Superseded|Deprecated]

Date: [YYYY-MM-DD]

Context:

Problem: [Clear problem statement]

Constraints: [Key limitations]

Assumptions: [Important assumptions]

Options Considered:

- Option: [Description]
- Pros: [Bulleted list]
- Cons: [Bulleted list]
- Risks: [Key risks]
- Cost: [Implementation effort]

Decision:

Choice: [Selected option]

Rationale: [Key deciding factors]

Scope: [Affected components]

Implementation:

Plan: [Key steps]

Timeline: [Expected duration]

Dependencies: [Required changes]

Validation:

Metrics: [Success metrics]

Tests: [Validation approach]

Reviews: [Review points]

2.3 Documentation Types & When to Use

2.3.a Essential Documents

1. System Overview (1-2 pages)

- Start of project
- Major pivots
- New team members onboarding

2. Key Decisions Log

- Ongoing
- Keep brief
- Link to discussions

3. Technical Debt Register

- Update each sprint
- Prioritize with product owner
- Link to user stories

2.4 Documentation Hierarchy

```
graph TD
    A[System Context] --> B[Architecture Vision]
    B --> C[Technical Architecture]
    B --> D[Data Architecture]
    C --> E[Component Specs]
    C --> F[Integration Architecture]
    C --> G[Quality Attributes]
    D --> H[Process Views]
    E --> I[Development View]
```

2.5 Document Dependencies

Document	Prerequisites	Dependencies
System Context	Business Requirements	None
Architecture Vision	System Context	Business Strategy
Technical Architecture	Architecture Vision	System Context
Data Architecture	Technical Architecture	System Context

Document	Prerequisites	Dependencies
Component Specs	Technical Architecture	Quality Attributes
Integration Architecture	Technical Architecture	External Systems
Quality Attributes	Technical Architecture	Business Requirements
Process Views	Technical Architecture	Data Architecture
Development View	Component Specs	Quality Attributes

2.6 Maintenance Schedule

Document Type	Update Frequency	Trigger Events
System Context	Quarterly	New integrations
Architecture Vision	Bi-annually	Strategy changes
Technical Architecture	Monthly	Major changes
Data Architecture	Monthly	Schema changes
Component Specs	Per release	New features
Quality Attributes	Quarterly	Performance review

2.7 Documentation Automation

Reduce manual documentation effort with these techniques:

1. Code-Generated Documentation

- API documentation from code comments (Swagger, JavaDoc, etc.)

- Architecture diagrams from code structure ((C4)[https://c4model.com/]C4, PlantUML)
 - Dependency graphs from build tools
2. **Documentation as Code**
 - Store docs in version control alongside code
 - Use Markdown/AsciiDoc for easier maintenance
 - Implement CI/CD pipelines for documentation validation
 3. **Automated Quality Checks**
 - Link validation
 - Diagram consistency
 - Terminology compliance
 - Readability metrics
 4. **Practical Example: Documentation Pipeline**

```
graph LR
    A[Code Repository] --> B[Documentation Extraction]
    B --> C[Format Conversion]
    C --> D[Validation]
    D --> E[Publication]
    F[Manual Docs] --> C
```

2.8 Review Process

1. **Quick Review** (1-2 days)
 - Self-review checklist
 - Automated checks (diagrams, links)
 - Technical writer review
 - Request peer feedback
2. **Team Review** (2-3 days)
 - Architecture review meeting
 - Code/design alignment check
 - Performance impact review
 - Security assessment
3. **Validation** (1-2 days)
 - Test against requirements
 - Verify metrics & SLAs
 - Check implementation feasibility

- Document feedback

2.9 Agile Documentation Principles

1. **Minimize Documentation Debt**
 - Write during development
 - Update or delete outdated docs
 - Automate where possible
 - Use code as documentation
2. **Make it Accessible**
 - Clear navigation structure
 - Consistent formatting
 - Searchable content
 - Version controlled
3. **Keep it Relevant**
 - Link to source code
 - Include examples
 - Add troubleshooting guides
 - Regular cleanup sprints

2.10 Template Usage in Agile

Docu- ment	When	Format	Review
System Over- view	Project start, quarter- ly	Wiki/ MD	Team re- view
Archi- tecture Decision	As needed	ADR	Team discus- sion
Com- ponent Guide	Per com- ponent	Code + Docs	PR re- view
Inte- gration Spec	Per inte- gration	API doc	Team + Partners

2.11 Architecture Documentation

2.11.a Documentation Structure

```
graph TD
    subgraph "Core Documentation"
        SC[System Context]
        CM[Component Model]
        DM[Deployment Model]
    end

    subgraph "Implementation"
        TG[Technical Guide]
        API[API Spec]
        DT[Data Model]
    end

    SC --> CM
    SC --> DM
    CM --> API
    CM --> DT
    DM --> TG
```

2.11.b Core Templates

Template	Purpose	Owner
System Context	System boundaries	Architect
Logical Architecture	System structure	Tech Lead
Component Architecture	Component design	Tech Lead
Integration Architecture	System interfaces	Integration Team
Infrastructure Architecture	Infrastructure	DevOps
Security Architecture	Security controls	Security Team
Data Architecture	Data structures	Data Team

2.12 Recommended Tools

Category	Tool	Best For	Learning Curve
Dia-grams	Mermaid	Version-controlled diagrams	Low
Dia-grams	PlantUML	UML diagrams	Medium
Dia-grams	C4 Model	System architecture	Medium
Documentation	Quarto	Technical publishing	Low
Collaboration	Confluence	Team documentation	Medium
API Docs	Swagger/OpenAPI	REST API documentation	Low
Version Control	Git	Documentation versioning	Medium

2.13 Architecture Metrics

2.13.a Key Performance Indicators

1. Technical Debt Ratio

- Measurement: Issues/KLOC
- Target: < 5%
- Review: Monthly

2. Architecture Compliance

- Measurement: % adherence
- Target: > 95%
- Review: Weekly

3. System Stability

- Measurement: MTBF
- Target: > 99.9%
- Review: Daily

2.14 Troubleshooting Guide

2.14.a Common Architecture Issues

Issue	Symptoms	Quick Fix	Long-term Solution	Solution
Tight Coupling	High change impact	Interface abstraction	Domain-driven design	
Performance issues	Slow response times	Caching	Architecture review	
Scalability	Resource bottlenecks	Load balancing	Service decomposition	

2.15 Documentation Flow

```
graph TD
    A[Business Requirements] --> B[System Context]
    B --> C[Architecture Vision]
    C --> D[Technical Design]
    D --> E[Implementation Guide]

    subgraph "Review Points"
        F[Architecture Review]
        G[Security Review]
        H[Performance Review]
    end

    D --> F
    D --> G
    D --> H

    subgraph "Validation"
```

```
I[Automated Tests]
J[Load Tests]
K[Security Scans]
end
```

```
F --> I
G --> K
H --> J
```

3 Conclusion

Effective architecture documentation is a balance between capturing essential design decisions and maintaining agility. This guide provides templates and processes that can be adapted to your team's specific needs and scaled according to project complexity.

Remember these key principles:

- Document with a purpose and audience in mind
- Keep documentation lightweight but sufficient
- Maintain documentation as the system evolves
- Automate what can be automated