

Component Architecture

[Author Name]

Université de Toulon, LIS UMR CNRS 7020

2026-10-03

Table des matières

1 Purpose of this Document	1
2 Component Overview	1
3 Functional Responsibilities	1
4 Technical Architecture	1
5 Component Structure	1
6 Interface Specifications	2
7 Event Interfaces	2
8 Dependencies	2
9 Technical Implementation	2
10 State Model	2
11 Quality Requirements	3
12 System Integration	3
13 Scalability Design	3
14 Infrastructure Requirements	3
15 Observability	4
16 Deployment Process	4
17 Example Implementation	4
18 Template Validation Checklist	4
19 Related Views	4

1 Purpose of this Document

This document provides detailed architectural information about the [Component Name] component, including its structure, interfaces, and technical specifications.

2 Component Overview

Name: [Component name]
Type: [Service|Library|Module|Application]

Pattern: [Microservice|Serverless|Batch|etc.]
Scale: [Usage metrics/anticipated load]
Status: [Active|Planned|Deprecated|In Development]
Owner: [Team or individual responsible]

3 Functional Responsibilities

- [Primary function 1]
- [Primary function 2]
- [Primary function 3]

4 Technical Architecture

```
graph TD
  A[Public API] --> B[Core Logic]
  B --> C[Data Access]
  C --> D[(Storage)]
  B --> E[External Services]
  F[Event Publisher] --> B
  B --> G[Event Consumer]
```

5 Component Structure

```
@startuml
package "Component" {
    [Core Logic]
    interface "Public API"
    interface "Events"
    database "Storage"
```

```

[Public API] --> [Core Logic]
[Core Logic] --> [Events]
[Core Logic] --> [Storage]
}
@enduml

```

6 Interface Specifications

```

APIs:
  Public:
    Protocol: [REST/GraphQL/gRPC/
etc.]
    Format: [JSON/XML/Proto/etc.]
    Version: [Semantic versioning/
date-based/etc.]
    Endpoints:
      - Path: /api/v1/resource
        Methods: [GET, POST, PUT,
DELETE]
        Auth: [JWT/OAuth/API Key/etc.]
        Rate Limit: [limits]
        Documentation: [link to API
docs]

      - Path: /api/v1/another-resource
        # Additional endpoints as needed

```

```

Internal:
  Protocol: [Protocol]
  Format: [Format]
  Endpoints:
    # As above for internal APIs

```

7 Event Interfaces

```

Published Events:
- Name: [event.name]
  Format: [Format]
  Schema: [Link or inline]
  Frequency: [Estimated volume]
  Consumers: [Known consumers]

```

```

Consumed Events:
- Name: [event.name]
  Source: [Producer name]
    Processing: [Synchronous/
Asynchronous]
  Handling: [Processing approach]

```

8 Dependencies

```

Dependencies:
  Required:
    - Name: [Dependency name]
      Version: [Version range]
      Purpose: [Why it's needed]
        Criticality: [Critical/
Important/Optional]

  Optional:
    - Name: [Dependency name]
      Version: [Version range]
      Purpose: [Why it's needed]
      Activation: [When it's used]

```

9 Technical Implementation

```

Framework: [Framework name + version]
Language: [Language + version]
Build Tool: [Build tool + version]
Test Framework: [Test framework]
Code Style: [Style guide reference]
Code Quality:
  Coverage: [Target percentage]
  Complexity: [Target limit]
  Dependencies: [Direct only/
Transitive allowed/etc.]

```

10 State Model

```

@startuml
[*] --> Initialized
Initialized --> Active : configure()

```

```

Active      -->      Processing      :
receive_request()
Processing --> Active : complete
Processing --> Error  : exception
Error --> Active  : recover()
Active --> [*] : shutdown()
@enduml

```

11 Quality Requirements

```

Performance:
  Latency: [Target, e.g., <100ms p95]
  Throughput: [Target, e.g., 1000 rps]
  Memory: [Target usage]
  CPU: [Target usage]

Reliability:
  Availability: [SLA target]
  MTTR: [Mean time to recovery]
  Backup: [Backup strategy]
  Error Handling: [Approach]

Security:
  Authentication: [Mechanism]
  Authorization: [Approach]
  Data Classification: [Level]
  Encryption: [Standards]

```

12 System Integration

```

External Dependencies:
  APIs:
    - System: [External system name]
      Protocol: [REST/GraphQL/gRPC/
etc.]
    Purpose: [Why we integrate]
    SLA: [Required service level]
    Fallback: [Contingency if
unavailable]

  Events:
    - Stream: [Event stream name]

```

```

Pattern: [pub/sub|queue|etc.]
Volume: [Expected events/second]
Handling: [Processing approach]

```

```

Storage:
  - Type: [SQL/NoSQL/Cache/etc.]
    Scale: [Size/operations per
second]
  Backup: [Strategy]
  Retention: [Policy]

```

13 Scalability Design

```

Horizontal Scaling:
  - Service Instances: [Strategy]
  - Data Partitioning: [Approach]
    - Caching Strategy:
[Implementation]

Vertical Scaling:
  - Resource Limits: [Bounds]
  - Performance Tuning: [Areas]
  - Optimization Areas: [Key focus]

Resilience:
  - Circuit Breakers:
[Implementation]
  - Rate Limiting: [Approach]
  - Fallback Mechanisms: [Strategies]
  - Retry Policies: [Specifications]

```

14 Infrastructure Requirements

```

Compute:
  CPU: [Cores/vCPUs]
  Memory: [GB]
  Instances: [Min/max/target]

Network:
  Bandwidth: [Mbps requirements]
  Latency: [Requirements]

```

```
Protocols: [TCP/UDP/etc.]
Ports: [Required open ports]
```

```
Storage:
  Type: [Block/object/file]
  IOPS: [Requirements]
  Capacity: [GB/TB]
  Growth: [Anticipated rate]
```

15 Observability

```
Metrics:
  - Name: [metric_name]
    Type: [Counter/Gauge/Histogram/
etc.]
    Purpose: [What it measures]
    Alert: [When to alert]

Logging:
  - Level: [Debug/Info/Warn/Error]
    Content: [Key information]
    Storage: [Retention policy]

Tracing:
  - Spans: [Key points]
    Sampling: [Rate]
    Propagation: [Method]
```

16 Deployment Process

```
CI/CD:
  Pipeline: [Tool/approach]
  Build: [Process]
  Test: [Strategy]
  Deploy: [Method]

Release:
  Strategy: [Blue/Green, Canary,
Rolling, etc.]
  Frequency: [Expected cadence]
  Rollback: [Process]
```

17 Example Implementation

```
// Sample code snippet demonstrating
core component pattern
public class ComponentExample {
    public void processRequest(Request
request) {
        // Input validation
        validate(request);

        // Core business logic
        Result result =
businessLogic.execute(request);

        // Persistence
        repository.save(result);

        // Event publication
        eventPublisher.publish(new
ResultEvent(result));
    }
}
```

18 Template Validation Checklist

- ☐ Component interfaces clearly defined
- ☐ API specifications documented
- ☐ Quality requirements specified
- ☐ Dependencies listed
- ☐ System integration patterns documented
- ☐ Scalability requirements specified
- ☐ Infrastructure requirements detailed
- ☐ Security controls verified
- ☐ Observability strategy defined
- ☐ Team review completed

19 Related Views

- System Context - Overall system context
- Infrastructure Architecture - Deployment details

- Integration Architecture - Integration patterns
- Development Guide - Implementation details