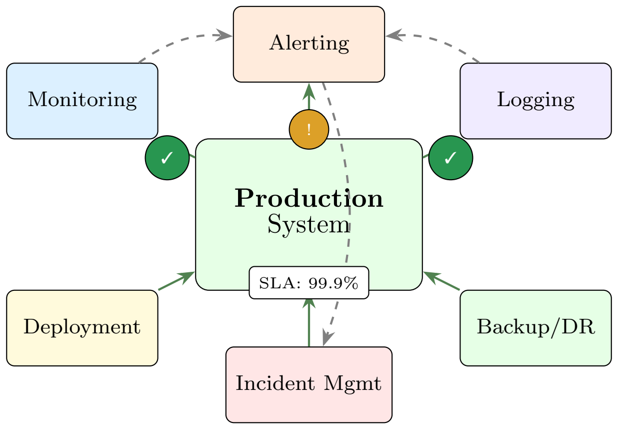
# Operational Viewpoint

Architecture Viewpoint Specification

System Operations, Monitoring & Administration



| | |
|---|---|
| **Version:** | 2.0 |
| **Status:** | Release |
| **Classification:** | ISO/IEC/IEEE 42010 Compliant |
| **Last Updated:** | December 12, 2025 |

Based on the Views and Beyond approach to software architecture documentation

# Contents

**1 Viewpoint Name** **3**
1.1 Viewpoint Classification . . . 3
1.2 Viewpoint Scope . . . 3

**2 Overview** **4**
2.1 Purpose and Scope . . . 4
2.2 Key Characteristics . . . 4
2.3 Relationship to Other Viewpoints . . . 5
2.4 Operational Architecture Overview . . . 6

**3 Concerns** **6**
3.1 Primary Concerns . . . 6
3.2 Concern-Quality Attribute Mapping . . . 8

**4 Anti-Concerns** **8**
4.1 Out of Scope Topics . . . 8

**5 Typical Stakeholders** **9**
5.1 Primary Stakeholders . . . 10
5.2 Secondary Stakeholders . . . 10
5.3 Stakeholder Concern Matrix . . . 11

**6 Model Types** **11**
6.1 Model Type Catalog . . . 11
6.2 Model Type Relationships . . . 13

**7 Model Languages** **13**
7.1 Observability Diagram Notation . . . 13
7.2 Incident Severity Levels . . . 14
7.3 SLO Specification Format . . . 14
7.4 Runbook Structure . . . 14
7.5 Deployment Pipeline Notation . . . 16
7.6 Tabular Specifications . . . 16
7.6.1 Alert Definition Table . . . 16
7.6.2 Backup Schedule Table . . . 16

**8 Viewpoint Metamodels** **16**
8.1 Core Metamodel . . . 17
8.2 Entity Definitions . . . 17
8.3 Relationship Definitions . . . 23

**9 Conforming Notations** **23**

9.1 ITIL Framework . . . 24
9.2 SRE Practices . . . 24
9.3 Prometheus/Grafana Ecosystem . . . 24
9.4 OpenTelemetry . . . 24
9.5 Operational Tools Comparison . . . 25

**10 Model Correspondence Rules 25**
10.1 Deployment View Correspondence . . . 25
10.2 Component-and-Connector View Correspondence . . . 26
10.3 Quality Attribute Correspondence . . . 26

**11 Operations on Views 26**
11.1 Creation Methods . . . 26
11.1.1 View Development Process . . . 27
11.1.2 Common Patterns . . . 28
11.2 Analysis Methods . . . 30
11.2.1 Availability Analysis . . . 30
11.2.2 Error Budget Analysis . . . 30
11.2.3 Incident Metrics Analysis . . . 30

**12 Examples 30**
12.1 Example 1: Observability Architecture . . . 31
12.2 Example 2: Incident Response Process . . . 31
12.3 Example 3: SLO Dashboard Specification . . . 32

**13 Notes 32**
13.1 Operational Maturity Model . . . 32
13.2 On-Call Best Practices . . . 33
13.3 Post-Mortem Best Practices . . . 33
13.4 Common Pitfalls . . . 33

**14 Sources 33**
14.1 Primary References . . . 34
14.2 Supplementary References . . . 34
14.3 Online Resources . . . 34

**A Operational View Checklist 35**

**B Glossary 35**

## 1 Viewpoint Name

**Viewpoint Identification**

| | |
|---|---|
| **Name:** | Operational Viewpoint |
| **Synonyms:** | Operations View, Production View, Administration Viewpoint, System Management View, Runtime Operations View, Site Reliability View, Platform Operations View |
| **Identifier:** | VP-OPS-001 |
| **Version:** | 2.0 |

### 1.1 Viewpoint Classification

The Operational Viewpoint addresses how a system is operated, administered, monitored, and maintained in production environments. While the Views and Beyond approach does not explicitly define an operational viewpoint as a primary style, operational concerns are critical cross-cutting aspects that intersect with deployment, component-and-connector, and quality attribute perspectives. This viewpoint provides essential guidance for running systems reliably in production.

Table 1: Viewpoint Classification Taxonomy

| Attribute | Value |
|---|---|
| Style Family | Cross-Cutting / Quality-Focused |
| Primary Focus | Production Operations and System Management |
| Abstraction Level | Operational / Runtime |
| Temporal Perspective | Runtime and Ongoing Operations |
| Related Styles | Deployment, C&C, Service-Oriented |
| IEEE 42010 Category | Operational Viewpoint |
| Industry Alignment | Site Reliability Engineering (SRE), DevOps, ITIL |

### 1.2 Viewpoint Scope

The Operational Viewpoint encompasses multiple aspects of system operations:

- **Observability:** Monitoring, logging, tracing, and metrics collection to understand system behavior and health in production.
- **Alerting and Incident Management:** Detection of anomalies, alert routing, incident response procedures, and escalation paths.
- **Deployment and Release:** Procedures for deploying, updating, and rolling back system changes safely.

- **Scaling and Capacity:** Mechanisms for handling load variations, capacity planning, and resource management.
- **Backup and Recovery:** Data protection, disaster recovery procedures, and business continuity planning.
- **Maintenance and Administration:** Routine operational tasks, configuration management, and system administration.
- **Service Level Management:** SLAs, SLOs, SLIs, error budgets, and service quality tracking.
- **Security Operations:** Runtime security monitoring, vulnerability management, and incident response.

## 2 Overview

The Operational Viewpoint provides a comprehensive framework for documenting how a software system is operated, monitored, and maintained throughout its production lifecycle. In modern systems where availability and reliability are critical, this viewpoint is essential for ensuring systems meet their service level commitments.

### 2.1 Purpose and Scope

The primary purpose of this viewpoint is to establish a clear understanding of operational requirements, procedures, and mechanisms that enable reliable system operation. This includes defining what needs to be monitored, how problems are detected and resolved, how changes are deployed safely, and how the system recovers from failures.

**Viewpoint Definition**

The Operational Viewpoint defines the mechanisms, procedures, and organizational structures required to operate a system in production. It encompasses observability infrastructure, incident management processes, deployment procedures, capacity management, disaster recovery capabilities, and service level definitions. This viewpoint bridges development activities with production operations.

### 2.2 Key Characteristics

The Operational Viewpoint exhibits several distinctive characteristics:

**Production Focus:** This viewpoint is concerned exclusively with the system in production environments, addressing the realities of operating software at scale with real users and data.

**Process and Procedure Emphasis:** Beyond technical mechanisms, this viewpoint documents the human processes, runbooks, and procedures essential for effective operations.

**Continuous Improvement:** Operations is iterative; this viewpoint supports ongoing refinement based on incidents, metrics, and changing requirements.

**Cross-Functional Integration:** Effective operations require coordination between development, operations, security, and business teams. This viewpoint facilitates that integration.

**Measurability:** Operational concerns are quantified through metrics, SLOs, and SLAs, enabling objective assessment of operational effectiveness.

## 2.3 Relationship to Other Viewpoints

The Operational Viewpoint connects to other architectural viewpoints in critical ways:

Table 2: Relationships to Other Viewpoints

| Viewpoint | Relationship |
|---|---|
| Deployment | Deployment topology defines what is operated. Infrastructure choices affect operational capabilities. |
| Component-and-Connector | Runtime components are the units of monitoring. Connector protocols affect observability options. |
| Information/Data | Data stores require backup and recovery. Data flows affect monitoring points. |
| Development | Module structure affects deployment units. Build outputs become operational artifacts. |
| Security | Security monitoring integrates with operational monitoring. Incident response covers security incidents. |
| Quality Attribute | Availability, performance, and reliability requirements drive operational mechanisms. |

### 2.4 Operational Architecture Overview

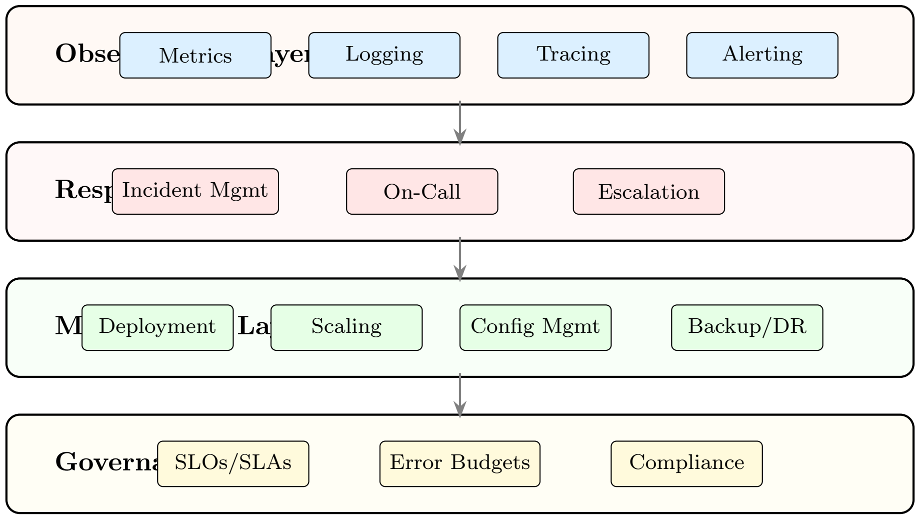


Figure 1: Operational Architecture Layers

## 3 Concerns

This section enumerates the architectural concerns that the Operational Viewpoint is designed to address. These concerns represent the fundamental questions stakeholders have about system operations.

### 3.1 Primary Concerns

**C1: System Observability**

- What metrics indicate system health and performance?
- How are logs collected, aggregated, and searched?
- How are distributed requests traced across services?
- What dashboards provide operational visibility?
- How is observability data retained and accessed?

**C2: Alerting and Notification**

- What conditions trigger alerts?
- How are alerts routed to appropriate responders?
- What severity levels and escalation paths exist?
- How is alert fatigue minimized?
- What notification channels are used?

**C3: Incident Management**

- How are incidents detected, classified, and tracked?
- What is the incident response process?

- How are on-call rotations managed?
- What escalation procedures exist?
- How are post-incident reviews conducted?

**C4: Deployment and Release**

- How are changes deployed to production?
- What deployment strategies are used (blue-green, canary, rolling)?
- How are rollbacks performed?
- What approval and change management processes exist?
- How is deployment frequency and lead time measured?

**C5: Scaling and Capacity**

- How does the system scale to handle load?
- What auto-scaling policies are configured?
- How is capacity planned and provisioned?
- What are the scaling limits and constraints?
- How is resource utilization optimized?

**C6: Backup and Disaster Recovery**

- What backup strategies protect data?
- What are the RPO and RTO requirements?
- How is disaster recovery tested?
- What failover mechanisms exist?
- How is business continuity ensured?

**C7: Configuration Management**

- How is configuration managed across environments?
- How are secrets and credentials handled?
- What configuration drift detection exists?
- How are configuration changes audited?
- What feature flag mechanisms are used?

**C8: Service Level Management**

- What SLOs define service quality targets?
- How are SLIs measured and reported?
- What SLAs are committed to customers?
- How are error budgets calculated and consumed?
- What happens when SLOs are breached?

**C9: Security Operations**

- How is security monitored in production?
- What vulnerability management processes exist?

- How are security incidents detected and responded to?
- What compliance monitoring is performed?
- How is access to production controlled?

**C10: Maintenance and Administration**

- What routine maintenance tasks are required?
- How are maintenance windows scheduled?
- What administrative interfaces exist?
- How is system documentation maintained?
- What automation reduces operational toil?

### 3.2 Concern-Quality Attribute Mapping

Table 3: Concern to Quality Attribute Mapping

| **Concern** | **Availability** | **Reliability** | **Performance** | **Security** | **Scalability** | **Mainttic.** | **Recovertic.** | **Operability** |
|---|---|---|---|---|---|---|---|---|
| Observability | ● | ● | ● | ○ | ○ | ● | ○ | ● |
| Alerting | ● | ● | ○ | ○ | – | ○ | ○ | ● |
| Incident Mgmt | ● | ● | ○ | ○ | – | ○ | ● | ● |
| Deployment | ● | ○ | ○ | ○ | ○ | ● | ○ | ● |
| Scaling | ● | ○ | ● | – | ● | ○ | ○ | ○ |
| Backup/DR | ● | ● | – | ○ | – | ○ | ● | ○ |
| Config Mgmt | ○ | ● | ○ | ● | ○ | ● | ○ | ● |
| SLO/SLA | ● | ● | ● | ○ | ○ | ○ | ○ | ● |
| Security Ops | ○ | ○ | – | ● | – | ○ | ○ | ○ |
| Maintenance | ○ | ● | ○ | ○ | ○ | ● | ○ | ● |

● = Primary impact, ○ = Secondary impact, – = Minimal impact

## 4 Anti-Concerns

Understanding what the Operational Viewpoint is *not* appropriate for helps stakeholders avoid misapplying this viewpoint.

### 4.1 Out of Scope Topics

**AC1: Software Design and Architecture**

- Internal component design decisions
- Algorithm implementations
- Code structure and organization
- API design patterns

- Data model design

**AC2: Development Process**

- Coding standards and practices
- Code review procedures
- Development environment setup
- Unit testing strategies
- Technical debt management

**AC3: Business Requirements**

- Functional requirement specifications
- User stories and acceptance criteria
- Business process definitions
- Product roadmap planning
- Feature prioritization

**AC4: User Experience**

- UI/UX design specifications
- User journey mapping
- Accessibility requirements
- Usability testing
- Frontend architecture

**AC5: Project Management**

- Sprint planning and tracking
- Resource allocation
- Budget management
- Vendor negotiations
- Stakeholder communication plans

**Common Misapplications**

Avoid using the Operational Viewpoint for:

- Documenting code module structure (use Development Viewpoint)
- Specifying data schemas (use Information Viewpoint)
- Defining infrastructure provisioning (use Deployment Viewpoint)
- Detailing component interactions (use C&C Viewpoint)
- Specifying functional behavior (use Functional Viewpoint)

## 5 Typical Stakeholders

The Operational Viewpoint serves multiple stakeholder communities with operational responsibilities.

## 5.1 Primary Stakeholders

Table 4: Primary Stakeholder Analysis

| Stakeholder | Role Description | Primary Interests |
|---|---|---|
| SRE/Platform Engineers | Ensure system reliability | SLOs, error budgets, automation, incident response, observability |
| Operations Teams | Manage production systems | Monitoring, alerting, runbooks, maintenance, on-call |
| DevOps Engineers | Bridge dev and operations | CI/CD, deployment automation, infrastructure as code |
| Incident Managers | Coordinate incident response | Incident process, escalation, communication, post-mortems |
| On-Call Engineers | First responders to issues | Alert routing, runbooks, troubleshooting, escalation |
| Capacity Planners | Ensure adequate resources | Scaling policies, capacity forecasting, cost optimization |

## 5.2 Secondary Stakeholders

Table 5: Secondary Stakeholder Analysis

| Stakeholder | Role Description | Primary Interests |
|---|---|---|
| Development Teams | Build and maintain software | Deployment process, observability, debugging in production |
| Security Teams | Protect system assets | Security monitoring, incident response, compliance |
| Product Managers | Define product direction | SLA commitments, feature flags, release coordination |
| Customer Support | Assist customers | Incident status, service health, known issues |
| Executive Leadership | Business accountability | SLA compliance, incident trends, operational costs |
| Compliance Officers | Ensure regulatory compliance | Audit trails, control effectiveness, compliance reporting |

### 5.3 Stakeholder Concern Matrix

Table 6: Stakeholder-Concern Responsibility Matrix

| | Observe | Alerting | Incident | Deploy | Scaling | Backup | Config | SLO | SecOps | Maint. |
|---|---|---|---|---|---|---|---|---|---|---|
| SRE | R | R | A | A | R | A | A | R | C | A |
| Ops Team | A | A | R | C | A | R | R | C | C | R |
| DevOps | C | C | C | R | C | C | R | I | I | C |
| Incident Mgr | I | C | R | I | I | I | I | A | C | I |
| On-Call | C | R | R | I | I | C | C | I | C | C |
| Security | C | C | C | C | I | C | C | I | R | I |

R = Responsible, A = Accountable, C = Consulted, I = Informed

## 6 Model Types

The Operational Viewpoint employs several complementary model types to capture different aspects of system operations.

### 6.1 Model Type Catalog

**MT1: Observability Architecture Diagram**

- *Purpose:* Show monitoring, logging, and tracing infrastructure
- *Primary Elements:* Collectors, aggregators, storage, visualization
- *Key Relationships:* Collects-from, forwards-to, stores-in, displays
- *Typical Notation:* Infrastructure diagrams, data flow diagrams

**MT2: Alert Configuration Model**

- *Purpose:* Document alerting rules, thresholds, and routing
- *Primary Elements:* Alerts, conditions, thresholds, recipients, channels
- *Key Relationships:* Triggers, routes-to, escalates-to
- *Typical Notation:* Alert matrices, routing tables

**MT3: Incident Response Process**

- *Purpose:* Define incident lifecycle and response procedures
- *Primary Elements:* States, roles, actions, escalation paths
- *Key Relationships:* Transitions-to, escalates-to, notifies
- *Typical Notation:* Process diagrams, state machines, RACI matrices

**MT4: Deployment Pipeline Diagram**

- *Purpose:* Show deployment stages and promotion paths
- *Primary Elements:* Stages, gates, environments, artifacts

- *Key Relationships:* Promotes-to, requires-approval, deploys-to
- *Typical Notation:* Pipeline diagrams, stage-gate diagrams

**MT5: Scaling Policy Model**

- *Purpose:* Document auto-scaling rules and capacity management
- *Primary Elements:* Metrics, thresholds, actions, limits
- *Key Relationships:* Triggers, scales, limits
- *Typical Notation:* Policy tables, threshold diagrams

**MT6: Disaster Recovery Plan**

- *Purpose:* Define backup, recovery, and continuity procedures
- *Primary Elements:* Components, RPO/RTO, procedures, sites
- *Key Relationships:* Backs-up, replicates-to, fails-over-to
- *Typical Notation:* DR topology diagrams, recovery procedures

**MT7: SLO/SLA Specification**

- *Purpose:* Define service level objectives and agreements
- *Primary Elements:* SLIs, SLOs, SLAs, error budgets
- *Key Relationships:* Measures, targets, commits-to
- *Typical Notation:* SLO tables, error budget charts

**MT8: Runbook/Playbook**

- *Purpose:* Document operational procedures and troubleshooting
- *Primary Elements:* Steps, decisions, commands, verification
- *Key Relationships:* Follows, branches-to, executes
- *Typical Notation:* Step-by-step procedures, flowcharts

**MT9: On-Call Schedule Model**

- *Purpose:* Define on-call rotations and coverage
- *Primary Elements:* Teams, schedules, escalation tiers
- *Key Relationships:* Covers, escalates-to, overrides
- *Typical Notation:* Schedule calendars, rotation tables

### 6.2 Model Type Relationships

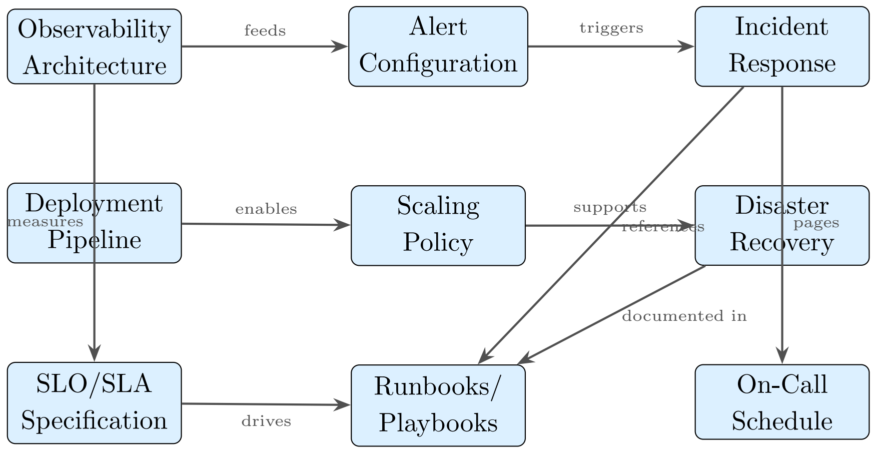


Figure 2: Model Type Dependency Relationships

## 7 Model Languages

For each model type, specific languages, notations, and techniques are prescribed. This section documents the vocabulary for constructing operational views.

### 7.1 Observability Diagram Notation

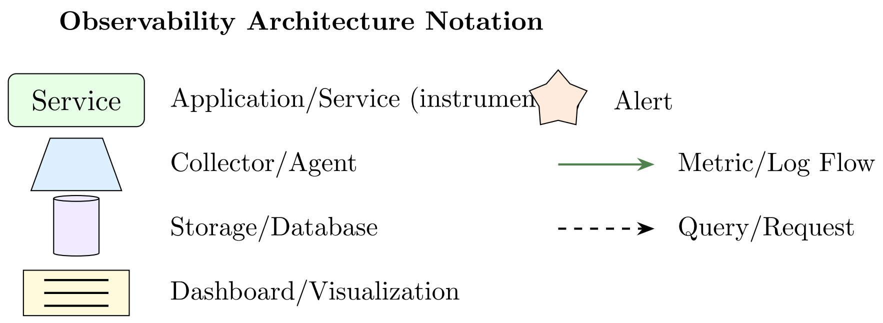


Figure 3: Observability Architecture Notation Legend

## 7.2 Incident Severity Levels

Table 7: Standard Incident Severity Classification

| Level | Name | Criteria | Response |
|---|---|---|---|
| **SEV1** | Critical | Complete service outage, data loss risk, security breach | Immediate all-hands, exec notification |
| **SEV2** | Major | Significant degradation, major feature unavailable | Immediate on-call response, escalation ready |
| **SEV3** | Minor | Limited impact, workaround available | Business hours response, next-day acceptable |
| **SEV4** | Low | Cosmetic issues, minor inconvenience | Scheduled maintenance window |

## 7.3 SLO Specification Format

**SLO Definition Template**

| | |
|---|---|
| **SLO Name:** | [Descriptive name for the objective] |
| **Service:** | [Service or system this SLO applies to] |
| **SLI (Indicator):** | [What is measured - e.g., success rate, latency] |
| **Target:** | [Numeric target - e.g., 99.9%, ¡ 200ms p99] |
| **Window:** | [Measurement window - e.g., rolling 30 days] |
| **Error Budget:** | [Allowed failures - e.g., 43.2 minutes/month] |
| **Owner:** | [Team responsible for this SLO] |
| **Rationale:** | [Business justification for target level] |

## 7.4 Runbook Structure

```
================================================================================
RUNBOOK: [Runbook Title]
================================================================================

METADATA
--------
Runbook ID:      [RB-XXX]
Last Updated:    [Date]
Owner:           [Team/Individual]
Review Date:     [Next review date]
Related Alerts:  [Alert names that trigger this runbook]

OVERVIEW
```

```
--------
Purpose:         [What this runbook addresses]
Symptoms:        [Observable symptoms that indicate this issue]
Impact:          [Business/user impact of this issue]
Urgency:         [SEV level, response time expectations]

PREREQUISITES
-------------
- Access to [systems/tools required]
- Permissions for [actions needed]
- Knowledge of [relevant background]

DIAGNOSTIC STEPS
----------------
1. [First diagnostic action]
   Expected: [What you should see if this is the issue]
   Command:  [Specific command if applicable]

2. [Second diagnostic action]
   ...

RESOLUTION STEPS
----------------
1. [First resolution action]
   Verification: [How to verify this step succeeded]

2. [Second resolution action]
   ...

ROLLBACK PROCEDURE
------------------
If resolution fails:
1. [Rollback step 1]
2. [Rollback step 2]

ESCALATION
----------
If unresolved after [time]:
- Escalate to: [Team/Individual]
- Contact: [Contact information]
- Provide: [Information needed for escalation]

POST-RESOLUTION
---------------
- [ ] Verify service restored
- [ ] Update incident ticket
- [ ] Notify stakeholders
- [ ] Schedule post-mortem if needed
==========================================================================================
```

Listing 1: Runbook Template Structure

## 7.5 Deployment Pipeline Notation

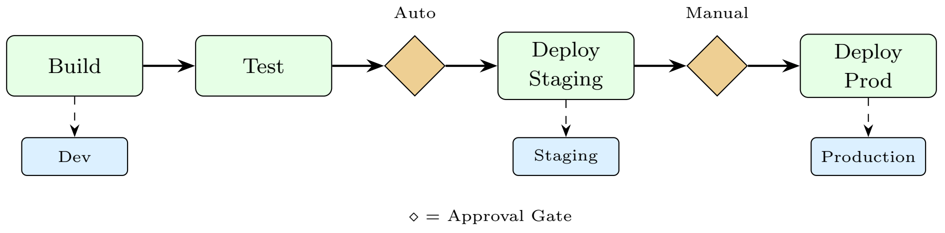


Figure 4: Deployment Pipeline Notation Example

## 7.6 Tabular Specifications

### 7.6.1 Alert Definition Table

Table 8: Example Alert Definition Format

| **Alert** | **Condition** | **Threshold** | **Severity** | **Team** | **Runbook** |
|---|---|---|---|---|---|
| HighErrorRate | error_rate ¿ X | ¿ 1% for 5m | SEV2 | Platform | RB-001 |
| HighLatency | p99_latency ¿ X | ¿ 500ms for 10m | SEV3 | Platform | RB-002 |
| DiskSpaceLow | disk_used ¿ X | ¿ 85% | SEV3 | Infra | RB-003 |
| ServiceDown | health_check | failed 3x | SEV1 | On-Call | RB-004 |

### 7.6.2 Backup Schedule Table

Table 9: Example Backup Schedule Format

| **System** | **Type** | **Frequency** | **Retention** | **RPO** | **Location** |
|---|---|---|---|---|---|
| Primary DB | Full | Daily 2AM | 30 days | 24 hours | S3 us-east-1 |
| Primary DB | Incremental | Hourly | 7 days | 1 hour | S3 us-east-1 |
| User Files | Full | Weekly Sun | 90 days | 7 days | S3 us-west-2 |
| Config Store | Full | Daily | 14 days | 24 hours | S3 us-east-1 |

# 8 Viewpoint Metamodels

This section defines the conceptual metamodel underlying the Operational Viewpoint.

## 8.1 Core Metamodel

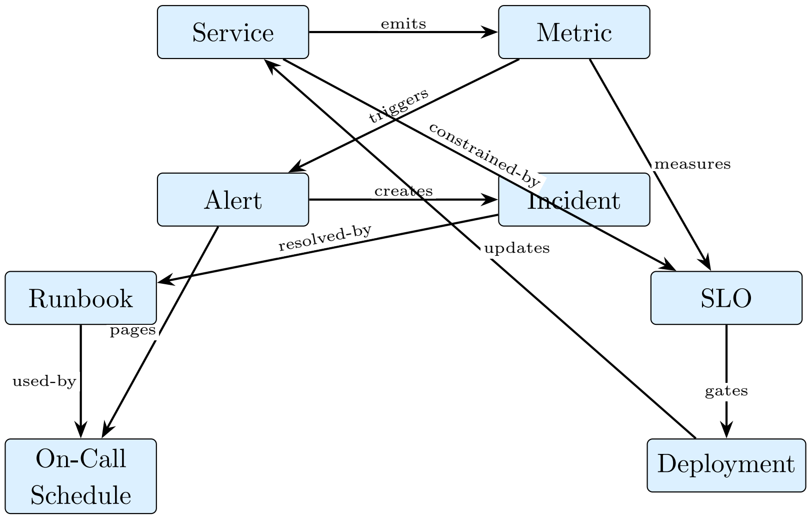


Figure 5: Operational Viewpoint Core Metamodel

## 8.2 Entity Definitions

**Entity: Service**

**Definition:** An operational unit that provides business functionality and is monitored, deployed, and managed as a cohesive entity.

**Attributes:**

- `serviceId`: Unique identifier
- `name`: Service name
- `description`: Service purpose
- `owner`: Owning team
- `tier`: Criticality tier (Tier 0, 1, 2, 3)
- `dependencies`: Upstream/downstream services
- `healthEndpoint`: Health check URL
- `documentation`: Links to service docs
- `onCallTeam`: Responsible on-call team
- `sloTargets`: Associated SLO references

**Constraints:**

- Every service must have a defined owner
- Tier 0/1 services must have SLOs defined
- Services must have health check endpoints
- On-call coverage must be assigned

**Entity: Metric**

**Definition:** A quantitative measurement collected from a service that indicates its health, performance, or behavior.

**Attributes:**

- `metricId`: Unique identifier
- `name`: Metric name (following naming conventions)
- `type`: Metric type (counter, gauge, histogram, summary)
- `unit`: Unit of measurement
- `labels`: Dimensional labels for filtering
- `source`: Service or component emitting metric
- `frequency`: Collection frequency
- `retention`: How long metric data is retained
- `aggregations`: Supported aggregation methods

**Constraints:**

- Metric names must follow naming conventions
- Label cardinality must be bounded
- Critical metrics must have defined retention

**Entity: Alert**

**Definition:** A notification triggered when a metric or condition crosses a defined threshold, indicating a potential problem.

**Attributes:**

- `alertId`: Unique identifier
- `name`: Alert name
- `description`: What the alert indicates
- `condition`: Triggering condition/query
- `threshold`: Threshold value(s)
- `duration`: How long condition must persist
- `severity`: Alert severity level
- `recipients`: Teams/individuals to notify
- `channels`: Notification channels (PagerDuty, Slack, email)
- `runbookLink`: Associated runbook
- `suppressionRules`: When to suppress the alert

**Constraints:**

- Alerts must have associated runbooks
- SEV1/2 alerts must page on-call
- Alerts should have clear, actionable descriptions
- Suppression rules must be documented

**Entity: Incident**

**Definition:** An unplanned event that disrupts or degrades service, requiring response and resolution.

**Attributes:**

- `incidentId`: Unique identifier
- `title`: Brief description
- `severity`: Incident severity level
- `status`: Current status (open, investigating, mitigating, resolved)
- `affectedServices`: Services impacted
- `startTime`: When incident began
- `detectionTime`: When incident was detected
- `mitigationTime`: When impact was mitigated
- `resolutionTime`: When fully resolved
- `commander`: Incident commander
- `responders`: Response team members
- `timeline`: Sequence of events and actions
- `rootCause`: Identified root cause
- `actionItems`: Follow-up actions

**Constraints:**

- SEV1/2 incidents require incident commander
- Incidents must have post-mortem within SLA
- Timeline must be maintained during incident
- Action items must be tracked to completion

**Entity: SLO (Service Level Objective)**

**Definition:** A target level of service quality measured by a specific indicator, defining acceptable performance boundaries.

**Attributes:**

- `sloId`: Unique identifier
- `name`: SLO name
- `service`: Associated service
- `sliDefinition`: Service Level Indicator specification
- `target`: Target percentage or value
- `window`: Measurement window (rolling or calendar)
- `errorBudget`: Calculated allowable failures
- `burnRate`: Current consumption rate
- `owner`: Responsible team
- `reviewCadence`: How often SLO is reviewed

**Constraints:**

- SLO targets must be achievable based on historical data
- SLIs must be measurable and automated
- Error budgets must have defined policies
- SLOs must be reviewed quarterly

**Entity: Runbook**

**Definition:** A documented procedure for diagnosing and resolving a specific operational issue or performing a routine task.

**Attributes:**

- `runbookId`: Unique identifier
- `title`: Descriptive title
- `description`: What the runbook addresses
- `triggeringAlerts`: Alerts that reference this runbook
- `prerequisites`: Required access and knowledge
- `diagnosticSteps`: Investigation procedures
- `resolutionSteps`: Fix procedures
- `rollbackSteps`: How to undo changes
- `escalationPath`: When and how to escalate
- `owner`: Maintaining team
- `lastTested`: When last verified
- `lastUpdated`: Last modification date

**Constraints:**

- Runbooks must be tested at least quarterly
- Critical runbooks must have multiple reviewers
- Steps must be specific and executable
- Runbooks must be kept current with system changes

**Entity: Deployment**

**Definition:** A release of software changes to a production environment, including the process, artifacts, and verification.

**Attributes:**

- `deploymentId`: Unique identifier
- `service`: Service being deployed
- `version`: Version being deployed
- `previousVersion`: Version being replaced
- `strategy`: Deployment strategy (rolling, blue-green, canary)
- `artifacts`: Deployed artifacts
- `configChanges`: Configuration changes included
- `deployer`: Person/system initiating deployment
- `approvals`: Required approvals obtained
- `startTime`: Deployment start time
- `endTime`: Deployment completion time
- `status`: Current status (pending, in-progress, completed, rolled-back)
- `verificationResults`: Post-deployment checks

**Constraints:**

- Production deployments require approval
- Deployments must have rollback capability
- Verification checks must pass before completion
- Deployments must be auditable

**Entity: On-Call Schedule**

**Definition:** A rotation defining who is responsible for responding to alerts and incidents during specific time periods.

**Attributes:**

- `scheduleId`: Unique identifier
- `name`: Schedule name
- `team`: Associated team
- `services`: Services covered
- `rotationPeriod`: Length of each rotation (daily, weekly)
- `participants`: Team members in rotation
- `escalationPolicy`: How to escalate if primary unavailable
- `overrides`: Scheduled coverage overrides
- `handoffProcedure`: How shifts are transferred

**Constraints:**

- 24/7 coverage must be maintained for critical services
- Escalation contacts must always be defined
- On-call load should be distributed fairly
- Handoff procedures must be followed

## 8.3 Relationship Definitions

Table 10: Metamodel Relationship Definitions

| Relationship | Source | Target | Description |
|---|---|---|---|
| emits | Service | Metric | Service produces this metric |
| triggers | Metric | Alert | Metric condition triggers alert |
| creates | Alert | Incident | Firing alert creates incident |
| constrained-by | Service | SLO | Service must meet this SLO |
| measures | Metric | SLO | Metric is used to calculate SLO |
| resolved-by | Incident | Runbook | Runbook used to resolve incident |
| pages | Alert | Schedule | Alert notifies on-call via schedule |
| updates | Deployment | Service | Deployment changes service |
| gates | SLO | Deployment | SLO status can block deployment |
| depends-on | Service | Service | Service requires another service |

# 9 Conforming Notations

Several existing notations and frameworks align with the Operational Viewpoint.

## 9.1 ITIL Framework

ITIL (Information Technology Infrastructure Library) provides comprehensive service management practices that align with operational concerns.

**Conformance Level:** Framework alignment for service management processes.

**Key Practices:** Incident Management, Problem Management, Change Management, Service Level Management.

## 9.2 SRE Practices

Google's Site Reliability Engineering practices provide detailed guidance for operational excellence.

**Conformance Level:** Full alignment for reliability engineering.

**Key Concepts:** SLOs, Error Budgets, Toil Reduction, Blameless Post-mortems.

## 9.3 Prometheus/Grafana Ecosystem

The Prometheus metrics format and Grafana dashboarding provide de facto standards for observability.

**Conformance Level:** Tool-specific implementation for monitoring.

**Key Formats:** PromQL, Grafana JSON dashboard model, Alertmanager configuration.

## 9.4 OpenTelemetry

OpenTelemetry provides vendor-neutral standards for telemetry data.

**Conformance Level:** Standard for traces, metrics, and logs.

**Key Specifications:** OTLP protocol, semantic conventions, SDK specifications.

### 9.5 Operational Tools Comparison

Table 11: Operational Tool Categories

| Category | Tools | Purpose |
|---|---|---|
| Metrics | Prometheus, Datadog, New Relic, CloudWatch | Time-series metrics collection and querying |
| Logging | ELK Stack, Splunk, Loki, CloudWatch Logs | Log aggregation, search, and analysis |
| Tracing | Jaeger, Zipkin, Datadog APM, X-Ray | Distributed request tracing |
| Alerting | PagerDuty, OpsGenie, VictorOps | Alert routing and incident management |
| Dashboarding | Grafana, Datadog, Kibana | Visualization and dashboards |
| Incident Mgmt | PagerDuty, Jira, ServiceNow | Incident tracking and coordination |
| On-Call | PagerDuty, OpsGenie, Rootly | On-call scheduling and escalation |
| Deployment | ArgoCD, Spinnaker, Jenkins, GitHub Actions | CI/CD and deployment automation |

## 10 Model Correspondence Rules

Model correspondence rules define how elements in operational models relate to elements in other architectural views.

### 10.1 Deployment View Correspondence

**Correspondence Rule CR-01: Service to Deployment Mapping**

**Rule:** Every operational service must correspond to deployed artifacts in the deployment view.

**Formal Expression:**

$$\forall s \in Services_{Ops} : \exists a \in Artifacts_{Deploy} : implements(a, s)$$

**Rationale:** Ensures operational services have physical deployment presence.

**Verification:** Deployment manifest review.

**Correspondence Rule CR-02: Monitoring Coverage**

**Rule:** Every deployed component must have corresponding observability instrumentation.

**Formal Expression:**

$$\forall c \in Components_{Deploy} : \exists M \subseteq Metrics : monitors(M, c)$$

**Rationale:** Ensures complete visibility into deployed systems.

**Verification:** Metric coverage analysis.

### 10.2 Component-and-Connector View Correspondence

**Correspondence Rule CR-03: Health Check Alignment**

**Rule:** Runtime components must have health endpoints that correspond to operational health checks.

**Formal Expression:**

$$\forall c \in Components_{C\&C} : \exists h \in HealthChecks : validates(h, c)$$

**Rationale:** Ensures runtime health can be assessed operationally.

**Verification:** Health endpoint testing.

### 10.3 Quality Attribute Correspondence

**Correspondence Rule CR-04: SLO to Requirement Mapping**

**Rule:** Every availability/performance quality requirement must have a corresponding SLO.

**Formal Expression:**

$$\forall qr \in QualityRequirements : type(qr) \in \{availability, performance\} \Rightarrow \exists slo \in SLOs : tracks(slo, qr)$$

**Rationale:** Ensures quality requirements are operationally measurable.

**Verification:** Requirements-to-SLO traceability.

## 11 Operations on Views

This section defines the methods for creating, interpreting, analyzing, and implementing operational views.

### 11.1 Creation Methods

### 11.1.1 View Development Process

**Step 1: Establish Operational Context**

1. Identify services and components requiring operational support
2. Gather availability and reliability requirements
3. Review existing operational processes and tools
4. Identify operational stakeholders and responsibilities
5. Assess current operational maturity level

**Step 2: Define Observability Strategy**

1. Identify key metrics for each service (RED: Rate, Errors, Duration)
2. Define logging strategy and log levels
3. Plan distributed tracing implementation
4. Select observability tooling stack
5. Define retention policies for telemetry data

**Step 3: Establish SLOs**

1. Define SLIs based on user-facing experience
2. Set initial SLO targets based on historical performance
3. Calculate error budgets
4. Define error budget policies
5. Establish SLO review cadence

**Step 4: Design Alerting Strategy**

1. Define alert severity levels and criteria
2. Create alerts based on SLOs (multi-window, multi-burn-rate)
3. Configure alert routing and escalation
4. Implement alert suppression for maintenance
5. Document alert-to-runbook mappings

**Step 5: Develop Incident Management Process**

1. Define incident severity classification
2. Establish incident response procedures
3. Configure on-call schedules and escalation
4. Create incident communication templates
5. Define post-mortem process

**Step 6: Plan Deployment and Recovery**

1. Design deployment pipeline with appropriate gates
2. Define rollback procedures
3. Establish backup and recovery procedures
4. Plan disaster recovery and business continuity
5. Document deployment runbooks

**Step 7: Create Runbooks**

1. Identify scenarios requiring runbooks
2. Document diagnostic and resolution procedures
3. Include escalation paths
4. Test and validate runbooks
5. Establish runbook review cycle

### 11.1.2 Common Patterns

**Pattern: RED Method Metrics**

**Context:** Need consistent service-level metrics for request-driven services.

**Solution:** Instrument every service with Rate, Errors, and Duration metrics.

**Metrics:**

- **Rate:** Requests per second
- **Errors:** Failed requests per second
- **Duration:** Distribution of request latencies

**Use When:** Monitoring microservices, APIs, web applications.

**Pattern: USE Method Metrics**

**Context:** Need consistent resource-level metrics for infrastructure.

**Solution:** Monitor Utilization, Saturation, and Errors for each resource.

**Metrics:**

- **Utilization:** Percentage of resource capacity used
- **Saturation:** Degree of queuing/waiting
- **Errors:** Error events for the resource

**Use When:** Monitoring infrastructure, databases, queues.

**Pattern: Multi-Window, Multi-Burn-Rate Alerts**

**Context:** Need alerts that balance responsiveness with noise reduction.
**Solution:** Use multiple time windows and burn rates to trigger alerts.
**Configuration:**

- Short window (5m) + high burn rate (14x) = Page immediately
- Medium window (1h) + medium burn rate (6x) = Page
- Long window (6h) + low burn rate (1x) = Ticket

**Use When:** SLO-based alerting for critical services.

**Pattern: Canary Deployment**

**Context:** Need to reduce risk when deploying changes to production.
**Solution:** Route small percentage of traffic to new version, monitor, then gradually increase.
**Process:**

1. Deploy new version alongside current version
2. Route 1-5% of traffic to canary
3. Monitor error rates and latency
4. Gradually increase traffic if healthy
5. Auto-rollback if metrics degrade

**Use When:** Risk-sensitive deployments, services with high traffic.

Table 12: Operational Patterns Summary

| **Pattern** | **Description** | **Use When** |
|---|---|---|
| RED Metrics | Rate, Errors, Duration for services | Request-driven services |
| USE Metrics | Utilization, Saturation, Errors | Infrastructure monitoring |
| Golden Signals | Latency, Traffic, Errors, Saturation | Comprehensive service monitoring |
| Circuit Breaker | Fail fast when dependency fails | Cascading failure prevention |
| Bulkhead | Isolate failures to components | Fault isolation |
| Blue-Green Deploy | Two identical environments | Zero-downtime deployment |
| Canary Deploy | Gradual traffic shift | Risk-sensitive deployment |
| Feature Flags | Toggle features without deploy | Progressive rollout |
| Chaos Engineering | Inject failures deliberately | Resilience validation |

## 11.2 Analysis Methods

### 11.2.1 Availability Analysis

**Availability Calculation**

**Purpose:** Calculate and track service availability.

**Formula:**

$$Availability = \frac{TotalTime - Downtime}{TotalTime} \times 100\%$$

**Common Targets:**

- 99% = 7.3 hours downtime/month
- 99.9% = 43.8 minutes downtime/month
- 99.95% = 21.9 minutes downtime/month
- 99.99% = 4.38 minutes downtime/month

### 11.2.2 Error Budget Analysis

**Error Budget Calculation**

**Purpose:** Track remaining budget for failures/changes.

**Formula:**

$$ErrorBudget = (1 - SLOTarget) \times Window$$
$$BudgetRemaining = ErrorBudget - Consumed$$
$$BurnRate = \frac{ErrorRate}{1 - SLOTarget}$$

**Example:** 99.9% SLO over 30 days

- Error Budget = 0.1% × 30 days = 43.2 minutes
- If 20 minutes consumed, 23.2 minutes remaining
- Burn Rate of 2x means budget exhausted in 15 days

### 11.2.3 Incident Metrics Analysis

**Key Incident Metrics**

**Purpose:** Measure incident management effectiveness.

**Key Metrics:**

- **MTTD (Mean Time to Detect):** Time from incident start to detection
- **MTTA (Mean Time to Acknowledge):** Time from alert to acknowledgment
- **MTTM (Mean Time to Mitigate):** Time from start to impact mitigation
- **MTTR (Mean Time to Resolve):** Time from start to full resolution
- **Incident Frequency:** Number of incidents per time period
- **Change Failure Rate:** Percentage of deployments causing incidents

# 12 Examples

### 12.1 Example 1: Observability Architecture

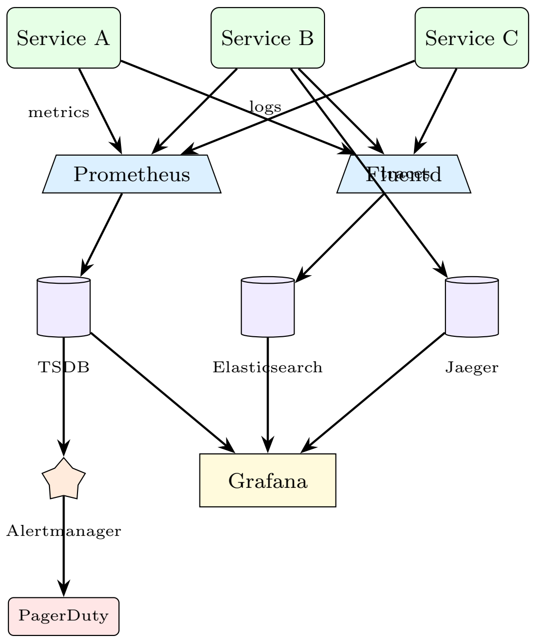


Figure 6: Observability Architecture Example

### 12.2 Example 2: Incident Response Process

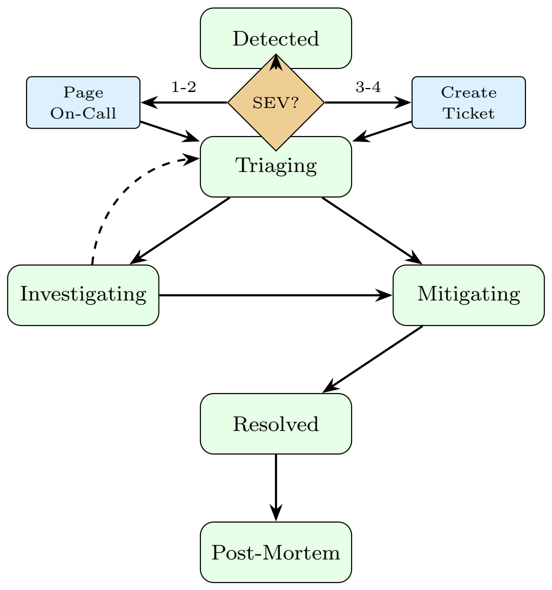


Figure 7: Incident Response State Machine

### 12.3 Example 3: SLO Dashboard Specification

Table 13: Example SLO Specification

| **Service: Payment API** | |
|---|---|
| **SLO Name** | Payment API Availability |
| **SLI** | Successful requests / Total requests (excluding 4xx) |
| **Target** | 99.95% |
| **Window** | Rolling 30 days |
| **Error Budget** | 21.9 minutes/month |
| **Current Status** | 99.97% (8.7 min consumed, 13.2 min remaining) |
| **Burn Rate** | 0.4x (healthy) |
| **Owner** | Payments Team |
| **SLO Name** | Payment API Latency |
| **SLI** | Requests completing ¡ 500ms / Total requests |
| **Target** | 99% |
| **Window** | Rolling 30 days |
| **Error Budget** | 7.3 hours/month |
| **Current Status** | 99.2% (5.8 hours consumed, 1.5 hours remaining) |
| **Burn Rate** | 0.8x (caution) |
| **Owner** | Payments Team |

## 13 Notes

### 13.1 Operational Maturity Model

Table 14: Operational Maturity Levels

| **Level** | **Name** | **Characteristics** | **Capabilities** |
|---|---|---|---|
| 1 | Reactive | Ad-hoc response, manual processes | Basic monitoring, manual deployment |
| 2 | Defined | Documented processes, some automation | Defined runbooks, CI/CD basics |
| 3 | Measured | SLOs defined, metrics-driven | SLO tracking, automated alerting |
| 4 | Managed | Error budgets, proactive management | Predictive scaling, chaos engineering |
| 5 | Optimized | Continuous improvement, self-healing | Auto-remediation, ML-based ops |

### 13.2 On-Call Best Practices

**On-Call Guidelines**

- Limit on-call shifts to reasonable durations (max 1 week)
- Ensure adequate rest between on-call periods
- Provide compensation or time-off for on-call duty
- Maintain clear escalation paths
- Ensure on-call engineers have access to all needed tools
- Conduct regular on-call handoffs
- Track and reduce on-call burden (toil)
- Avoid paging for non-actionable alerts

### 13.3 Post-Mortem Best Practices

**Blameless Post-Mortem Guidelines**

- Focus on systems and processes, not individuals
- Assume everyone acted with best intentions and available information
- Document the complete timeline with evidence
- Identify multiple contributing factors
- Generate actionable, assigned, and tracked follow-ups
- Share learnings broadly across the organization
- Review follow-up item completion
- Use post-mortems as learning opportunities

### 13.4 Common Pitfalls

**Common Mistakes to Avoid**

1. **Alert Fatigue:** Too many alerts leading to ignored pages
2. **Missing Runbooks:** Alerts without documented response procedures
3. **SLO Without Consequences:** SLOs that don't affect decisions
4. **Untested DR:** Disaster recovery plans never validated
5. **Manual Toil:** Repetitive tasks not automated
6. **Blame Culture:** Post-mortems that assign blame
7. **Incomplete Observability:** Blind spots in monitoring coverage
8. **Overly Complex Deployments:** Deployments without rollback capability

## 14 Sources

## 14.1 Primary References

1. Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (2016). *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media.
2. Beyer, B., Murphy, N. R., Rensin, D. K., Kawahara, K., & Thorne, S. (2018). *The Site Reliability Workbook*. O'Reilly Media.
3. Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The DevOps Handbook*. IT Revolution Press.
4. Limoncelli, T. A., Chalup, S. R., & Hogan, C. J. (2014). *The Practice of Cloud System Administration*. Addison-Wesley Professional.
5. Majors, C., Fong-Jones, L., & Miranda, G. (2022). *Observability Engineering*. O'Reilly Media.

## 14.2 Supplementary References

6. Sridharan, C. (2018). *Distributed Systems Observability*. O'Reilly Media.
7. Burns, B. (2018). *Designing Distributed Systems*. O'Reilly Media.
8. Rosenthal, C., & Jones, N. (2020). *Chaos Engineering*. O'Reilly Media.
9. Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate: The Science of Lean Software and DevOps*. IT Revolution Press.
10. AXELOS. (2019). *ITIL Foundation: ITIL 4 Edition*. TSO.

## 14.3 Online Resources

- Google SRE Books: `https://sre.google/books/`
- Prometheus Documentation: `https://prometheus.io/docs/`
- OpenTelemetry: `https://opentelemetry.io/`
- PagerDuty Incident Response: `https://response.pagerduty.com/`
- Grafana Documentation: `https://grafana.com/docs/`

## A Operational View Checklist

| Item | Complete? |
|---|---|
| **Observability** | |
| Metrics collection configured for all services | ☐ |
| Logging aggregation in place | ☐ |
| Distributed tracing enabled | ☐ |
| Dashboards created for key services | ☐ |
| Retention policies defined | ☐ |
| **Alerting** | |
| Alert rules defined based on SLOs | ☐ |
| Alert routing configured | ☐ |
| Escalation policies in place | ☐ |
| All alerts linked to runbooks | ☐ |
| **Incident Management** | |
| Incident severity levels defined | ☐ |
| Incident response process documented | ☐ |
| On-call schedules configured | ☐ |
| Communication templates ready | ☐ |
| Post-mortem process defined | ☐ |
| **Service Levels** | |
| SLIs defined for critical services | ☐ |
| SLO targets established | ☐ |
| Error budgets calculated | ☐ |
| SLO dashboards created | ☐ |
| **Deployment & Recovery** | |
| Deployment pipeline documented | ☐ |
| Rollback procedures defined | ☐ |
| Backup strategy implemented | ☐ |
| DR plan documented and tested | ☐ |
| **Documentation** | |
| Runbooks created for common issues | ☐ |
| Service catalog maintained | ☐ |
| Architecture diagrams current | ☐ |
| On-call documentation complete | ☐ |

## B Glossary

| | |
|---|---|
| **Alert** | A notification triggered when a monitored condition exceeds a threshold. |
| **Burn Rate** | The rate at which error budget is being consumed relative to the SLO window. |
| **Error Budget** | The amount of unreliability allowed by an SLO over a given time period. |

| | |
|---|---|
| **Incident** | An unplanned event causing service disruption or degradation. |
| **Mean Time to Detect (MTTD)** | Average time from incident start to detection. |
| **Mean Time to Resolve (MTTR)** | Average time from incident start to full resolution. |
| **Observability** | The ability to understand system state from external outputs (metrics, logs, traces). |
| **On-Call** | A rotation of engineers responsible for responding to production issues. |
| **Post-Mortem** | A review conducted after an incident to identify causes and improvements. |
| **Recovery Point Objective (RPO)** | Maximum acceptable data loss measured in time. |
| **Recovery Time Objective (RTO)** | Maximum acceptable time to restore service after failure. |
| **Runbook** | A documented procedure for handling operational tasks or incidents. |
| **Service Level Agreement (SLA)** | A contractual commitment to service quality with consequences. |
| **Service Level Indicator (SLI)** | A quantitative measure of service quality. |
| **Service Level Objective (SLO)** | A target value for an SLI representing desired service quality. |
| **Toil** | Repetitive, manual operational work that scales linearly with service growth. |