# Software Architecture Documentation

Variability Guide

A Comprehensive Guide to Documenting Architectural Variability, Configuration, and Product Line Flexibility

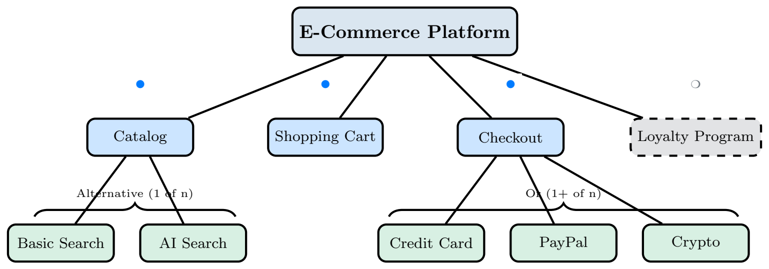
*Architecture Documentation Series*

Based on Software Product Line Engineering, Feature Modeling, and Industry Best Practices

December 4, 2025

**Abstract**

The Variability Guide documents how an architecture accommodates variation—the deliberate flexibility built into a system to support different configurations, product variants, deployment environments, and customer customizations. Effective variability management is essential for software product lines, configurable systems, and platforms that must serve diverse needs without fragmenting into unmaintainable branches. This comprehensive guide establishes principles and practices for documenting variation points, binding times, allowed variants, configuration mechanisms, and constraints. The document covers feature modeling, variation point taxonomy, binding time analysis, configuration management strategies, constraint specification, and governance processes for maintaining variability throughout the system lifecycle. Whether building a product line, a configurable enterprise system, or a multi-tenant platform, this guide provides the foundation for systematic variability documentation.



# Contents

# 1 Introduction to Variability

## 1.1 Definition and Purpose

Variability in software architecture refers to the ability of a system or architecture to be efficiently extended, changed, customized, or configured for use in a particular context. The Variability Guide documents how this flexibility is designed into the architecture and how it should be exercised.

**Definition**

**Architectural Variability** is the deliberate design of flexibility into a software architecture, enabling a single architecture to support multiple products, configurations, deployments, or customizations through well-defined variation points that can be bound at various times in the software lifecycle.

The Variability Guide serves several critical purposes. First, it enables **product line engineering** by documenting how a single architecture supports multiple products. Second, it supports **configuration management** by specifying what can be configured and how. Third, it facilitates **deployment flexibility** by explaining environment-specific variations. Fourth, it enables **customization** by defining how customer-specific adaptations are achieved. Fifth, it supports **evolution** by identifying extension points for future capabilities.

## 1.2 The Business Case for Variability

Well-managed variability provides significant business value through several mechanisms. It enables **mass customization** by allowing products to be tailored to customer needs without custom development. It accelerates **time-to-market** by enabling new product variants to be derived from existing assets. It reduces **maintenance costs** by sharing core components across product variants. It provides **market segmentation** by supporting different feature sets for different market segments. Finally, it ensures **future-proofing** by making the system adaptable to changing requirements.

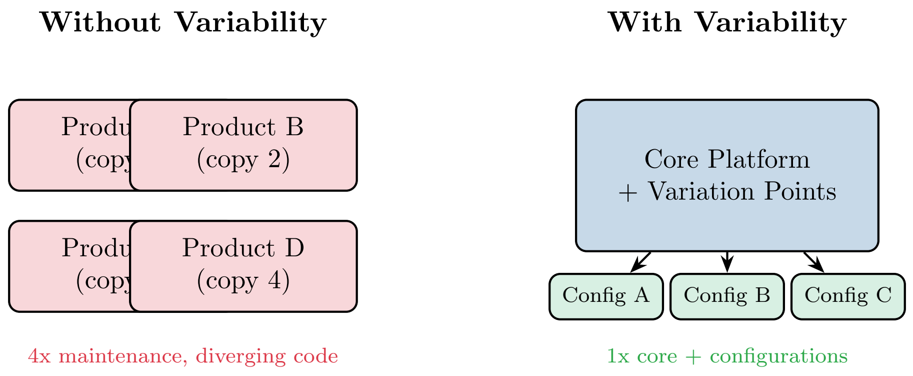


Figure 1: Value of Systematic Variability Management

## 1.3 Variability in the Architecture Documentation

The Variability Guide is part of the context documentation for an architectural view. It explains how the structure shown in the primary presentation can vary and under what conditions.

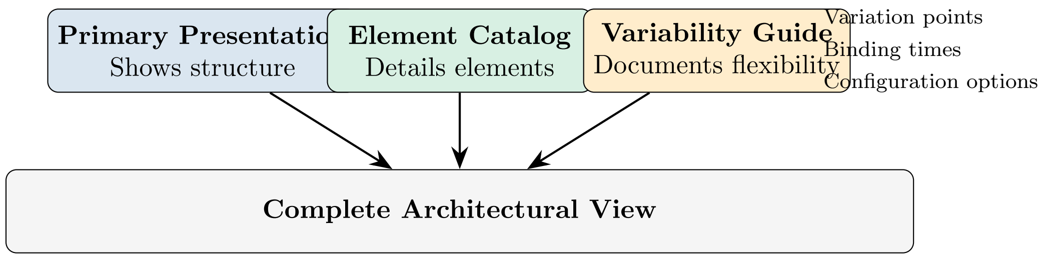


Figure 2: Variability Guide within View Documentation

### 1.4 Standards and Frameworks

Variability documentation draws from several established approaches. Software Product Line Engineering (SPLE) provides the foundational concepts for systematic variability management. Feature-Oriented Domain Analysis (FODA) introduced feature modeling for representing variability. Common Variability Language (CVL) offers a standardized notation for variability. Orthogonal Variability Model (OVM) provides a view-based approach to variability. ISO/IEC 26550 establishes reference models for software product lines.

# 2 Variability Concepts and Taxonomy

## 2.1 Core Concepts

Understanding variability requires familiarity with several key concepts.

**Definition**

A **Variation Point** is a location in the architecture where variation can occur—a place where different alternatives (variants) can be selected or configured to customize system behavior or structure.

**Definition**

A **Variant** is a specific alternative that can be selected at a variation point. Each variant represents one possible way to resolve the variability.

**Definition**

**Binding Time** is the point in the software lifecycle when a variation point is resolved—when a specific variant is selected and the variability is eliminated.

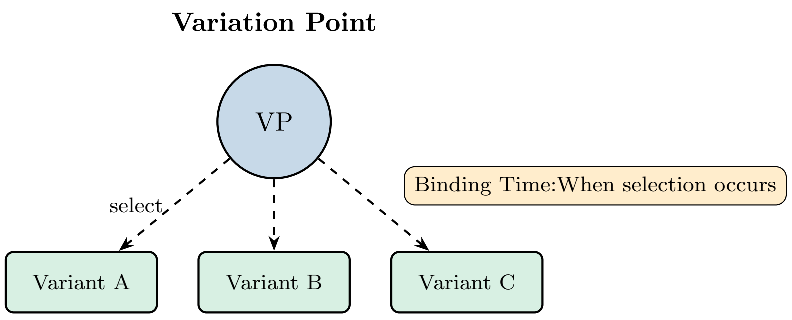


Figure 3: Variation Point, Variants, and Binding

## 2.2 Types of Variability

Variability can be classified along several dimensions.

### 2.2.1 By Subject

Table 1: Variability by Subject

| **Subject** | **What Varies** | **Examples** |
|---|---|---|
| Functional | Features and capabilities | Payment methods; report types; workflow steps |
| Data | Data models and schemas | Currency formats; product attributes; user fields |
| Platform | Runtime environment | OS; database; cloud provider |
| Interface | User interaction | Web UI; mobile app; API; CLI |
| Quality | Non-functional properties | Performance tiers; security levels; SLA options |
| Integration | External connections | Third-party APIs; partner systems; protocols |
| Localization | Regional adaptation | Language; date format; currency; regulations |

### 2.2.2 By Cardinality

Table 2: Variability by Cardinality

| **Cardinality** | **Selection Rule** | **Examples** |
|---|---|---|
| Mandatory | Must be included (no choice) | Core security module; logging |
| Optional | Zero or one selection | Premium features; optional integrations |
| Alternative | Exactly one from set | Database vendor; authentication provider |

| Cardinality | Selection Rule | Examples |
|---|---|---|
| Or | One or more from set | Payment methods; export formats |
| Multiple | Zero or more from set | Plugin modules; add-on features |

### 2.2.3 By Visibility

Table 3: Variability by Visibility

| Visibility | Who Controls | Examples |
|---|---|---|
| External | End users or customers | User preferences; tenant settings |
| Product Manager | Product configuration team | Feature toggles; edition definitions |
| Developer | Development team | Build variants; compiler flags |
| Operator | Operations/DevOps team | Deployment parameters; scaling settings |
| Internal | Hidden implementation details | Algorithm selection; optimization strategies |

## 2.3 Binding Time Spectrum

Binding time significantly affects how variability is implemented and managed.

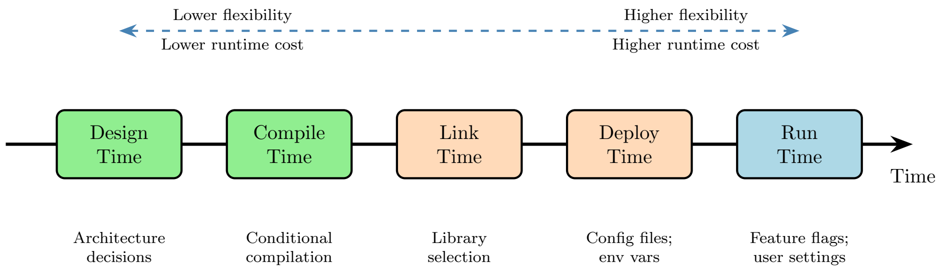


Figure 4: Binding Time Spectrum with Tradeoffs

### 2.3.1 Binding Time Details

Table 4: Binding Time Characteristics

| Binding Time | Mechanism | Advantages | Disadvantages |
|---|---|---|---|
| Design Time | Pattern selection; architecture | Maximum optimization | No flexibility post-design |
| Compile Time | Preprocessor; generics; build profiles | Type safety; optimization | Requires rebuild |

| Binding Time | Mechanism | Advantages | Disadvantages |
|---|---|---|---|
| Link Time | Static/dynamic libraries; DI containers | Modular builds | Deployment per variant |
| Deployment Time | Config files; environment variables | No code changes | Requires redeployment |
| Startup Time | Configuration loading; plugin discovery | Simple management | Requires restart |
| Runtime | Feature flags; settings; A/B tests | Maximum flexibility | Performance overhead |

## 3 Variation Point Documentation

### 3.1 Variation Point Registry

The variation point registry provides a comprehensive inventory of all points where the architecture supports variation.

Template

**Variation Point Specification Template**

**Identification**

- **VP ID:** Unique identifier
- **Name:** Descriptive name
- **Category:** Functional / Data / Platform / Interface / Quality

**Location**

- **Element:** Architectural element containing the VP
- **Implementation:** Code/config location

**Variability Characteristics**

- **Cardinality:** Mandatory / Optional / Alternative / Or
- **Binding Time:** When variation is resolved
- **Binding Mechanism:** How variation is realized

**Variants**

- **Available Variants:** List of options
- **Default:** Default variant if not specified

**Constraints**

- **Dependencies:** Required relationships with other VPs
- **Exclusions:** Incompatible combinations

### 3.2 Comprehensive Variation Point Registry

Table 5: Variation Point Registry

| VP ID | Name | Location | Binding | Description |
|---|---|---|---|---|
| VP-01 | Database Provider | Data Layer | Deploy | Selection of relational database (PostgreSQL, MySQL, SQL Server) |
| VP-02 | Authentication Method | Auth Service | Deploy | Choice of auth provider (OIDC, SAML, LDAP, Local) |
| VP-03 | Payment Gateway | Checkout | Runtime | Payment processor integration (Stripe, PayPal, Adyen) |
| VP-04 | Search Engine | Catalog | Deploy | Search implementation (Elasticsearch, Algolia, Native) |
| VP-05 | Notification Channels | Notify Service | Runtime | Enabled channels (Email, SMS, Push, Slack) |
| VP-06 | Storage Provider | File Service | Deploy | Object storage (S3, GCS, Azure Blob, MinIO) |
| VP-07 | Cache Strategy | All Services | Deploy | Caching implementation (Redis, Memcached, Local) |
| VP-08 | Feature Flags | All Modules | Runtime | Feature toggle states per environment |
| VP-09 | UI Theme | Frontend | Runtime | Visual theme and branding per tenant |
| VP-10 | Reporting Engine | Analytics | Compile | Report generation (Built-in, Crystal, SSRS) |
| VP-11 | Tax Calculator | Checkout | Runtime | Tax service integration (Avalara, TaxJar, Built-in) |
| VP-12 | Shipping Provider | Fulfillment | Runtime | Carrier integration (FedEx, UPS, DHL, Custom) |

## 3.3 Detailed Variation Point Specifications

### VP-01: Database Provider

**Identification**

- **ID:** VP-01
- **Name:** Database Provider
- **Category:** Platform
- **Owner:** Platform Team

**Location**

- **Element:** Data Access Layer
- **Implementation:** `config/database.yml`, `src/data/providers/`
- **View Reference:** [VIEW:DataFlow] Database component

**Variability Characteristics**

- **Cardinality:** Alternative (exactly one)
- **Binding Time:** Deployment time
- **Binding Mechanism:** Environment variable + configuration file
- **Rebinding:** Requires redeployment and data migration

**Rationale**

Different deployment environments and customer requirements necessitate support for multiple database vendors. Enterprise customers often mandate specific database vendors for compliance or existing infrastructure alignment. Cloud deployments may prefer managed services from specific providers.

**Variants**

| Variant | When to Use | Notes |
|---|---|---|
| PostgreSQL | Default; cloud deployments; new installations | Best feature support; recommended |
| MySQL | Legacy integrations; cost-sensitive deployments | Limited JSON support; avoid for complex queries |
| SQL Server | Enterprise Windows environments | License cost; excellent tooling |
| Aurora | AWS deployments requiring high availability | PostgreSQL-compatible; managed scaling |

**Default:** PostgreSQL

**Configuration**

```
# Environment variable
DATABASE_PROVIDER=postgresql

# config/database.yml
database:
  provider: ${DATABASE_PROVIDER}
  host: ${DB_HOST}
  port: ${DB_PORT}
  name: ${DB_NAME}
  pool_size: 20
```

## VP-03: Payment Gateway

**Identification**

- **ID:** VP-03
- **Name:** Payment Gateway
- **Category:** Integration
- **Owner:** Payments Team

**Location**

- **Element:** Payment Service, Checkout Module
- **Implementation:** `src/payments/gateways/`, feature flags
- **View Reference:** [VIEW:Component] Payment Service

**Variability Characteristics**

- **Cardinality:** Or (one or more from set)
- **Binding Time:** Runtime (per transaction)
- **Binding Mechanism:** Feature flags + customer preferences + availability
- **Rebinding:** Dynamic; no restart required

**Rationale**

Multiple payment gateways provide customer choice, geographic coverage, and redundancy. Customers may prefer specific payment methods. Gateway availability may vary, requiring failover capability. Different gateways offer different fee structures.

**Variants**

| Variant | Coverage | Strengths | Limitations |
|---|---|---|---|
| Stripe | Global | Developer experience; API quality | Higher fees in some regions |
| PayPal | Global | Consumer trust; buyer protection | Lower conversion for some segments |
| Adyen | Global | Enterprise; unified platform | Complex integration |
| Square | US, CA, UK, AU | SMB-focused; POS integration | Limited global coverage |

**Default:** Stripe (primary), PayPal (fallback)

**Selection Logic**

```
PaymentGateway selectGateway(Order order, Customer customer) {
    // Customer preference takes priority
    if (customer.preferredGateway != null
        && isAvailable(customer.preferredGateway)) {
        return customer.preferredGateway;
    }

    // Regional optimization
    Gateway regional = getRegionalOptimal(customer.country);
    if (isAvailable(regional)) {
        return regional;
    }

    // Fallback chain
    return getFirstAvailable(GATEWAY_PRIORITY_LIST);
}
```

## VP-08: Feature Flags

**Identification**

- **ID:** VP-08
- **Name:** Feature Flags
- **Category:** Functional
- **Owner:** Product Team

**Location**

- **Element:** All modules (cross-cutting)
- **Implementation:** LaunchDarkly integration, `src/common/features/`
- **View Reference:** [VIEW:Module] Cross-cutting concerns

**Variability Characteristics**

- **Cardinality:** Multiple (independent on/off per flag)
- **Binding Time:** Runtime (instant, per-request)
- **Binding Mechanism:** Feature flag service API
- **Rebinding:** Immediate; no deployment required

**Feature Flag Registry**

| Flag | Default | Scope | Description |
|---|---|---|---|
| `new-checkout-flow` | off | percentage | Redesigned checkout experience |
| `ai-recommendations` | off | user-segment | ML-powered product recommendations |
| `crypto-payments` | off | per-tenant | Enable cryptocurrency payments |
| `dark-mode` | on | per-user | Dark theme option |
| `beta-features` | off | per-user | Access to beta features |
| `maintenance-mode` | off | global | System maintenance indicator |

**Flag Evaluation Context**

```
// Feature flag evaluation with context
boolean isEnabled = featureFlags.evaluate(
    "new-checkout-flow",
    Context.builder()
        .user(currentUser.id)
        .tenant(currentTenant.id)
        .environment(Environment.PRODUCTION)
        .country(currentUser.country)
        .build()
);
```

**Lifecycle Management**

- New flags start in `development` environment only
- Gradual rollout: 1% → 10% → 50% → 100%
- Flags older than 90 days without changes flagged for review
- Permanent flags documented separately from release flags

# 4 Allowed Variants and Selection Rules

## 4.1 Variant Documentation Structure

Each variant should be documented with sufficient detail for selection decisions and implementation.

**Template**

**Variant Specification Template**

**Identity**

- **Variant ID:** Unique identifier
- **Name:** Descriptive name
- **Variation Point:** Parent VP reference

**Description**

- **Purpose:** What this variant provides
- **When to Use:** Selection criteria
- **When NOT to Use:** Counter-indications

**Technical Details**

- **Implementation:** How it's realized
- **Dependencies:** Required components/libraries
- **Configuration:** Settings required

**Impact Assessment**

- **Quality Attributes:** Effects on performance, security, etc.
- **Cost:** Licensing, infrastructure, operational costs
- **Risks:** Known issues or limitations

## 4.2 Variant Comparison Matrices

For each variation point with multiple significant variants, provide comparison matrices.

Table 6: VP-02: Authentication Method Variants Comparison

| **Criterion** | **OIDC** | **SAML** | **LDAP** | **Local** | **Weight** |
|---|---|---|---|---|---|
| Modern standard | ✓✓ | ✓ | – | – | High |
| Enterprise adoption | ✓ | ✓✓ | ✓✓ | – | High |
| Mobile support | ✓✓ | ✓ | – | ✓ | Medium |
| Setup complexity | Low | High | Medium | Low | Medium |
| Self-contained | – | – | – | ✓✓ | Low |
| Federation support | ✓✓ | ✓✓ | – | – | Medium |

| **Criterion** | **OIDC** | **SAML** | **LDAP** | **Local** | **Weight** |
|---|---|---|---|---|---|
| License cost | Varies | Varies | – | – | Medium |
| **Best For** | Modern SaaS | | Enterprise | | SMB/Standalone |

## 4.3 Selection Decision Trees

Provide decision trees for complex variant selection.

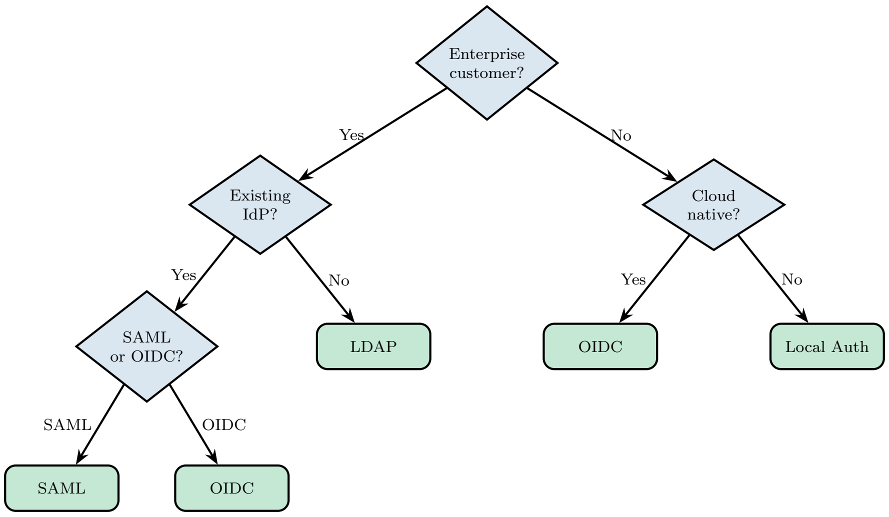


Figure 5: VP-02: Authentication Method Selection Decision Tree

## 4.4 Variant Selection Rules

Document formal rules that govern variant selection.

Table 7: Variant Selection Rules

| **Rule** | **Variation Point** | **Condition** | **Required Variant** |
|---|---|---|---|
| SR-01 | VP-01 (Database) | AWS deployment | Aurora or PostgreSQL |
| SR-02 | VP-02 (Auth) | Enterprise tier customer | OIDC or SAML |
| SR-03 | VP-02 (Auth) | Healthcare deployment (HIPAA) | SAML with MFA |
| SR-04 | VP-04 (Search) | Catalog > 100K products | Elasticsearch or Algolia |
| SR-05 | VP-06 (Storage) | Multi-region deployment | S3 or GCS (not MinIO) |
| SR-06 | VP-07 (Cache) | High availability required | Redis with Sentinel |

| Rule | Variation Point | Condition | Required Variant |
|---|---|---|---|
| SR-07 | VP-11 (Tax) | US customers present | Avalara or TaxJar |
| SR-08 | VP-03 (Payment) | EU customers present | Stripe or Adyen |

## 5 Configuration Mechanisms

### 5.1 Configuration Mechanism Overview

Different variability types require different configuration mechanisms. The choice of mechanism affects flexibility, safety, and operational complexity.

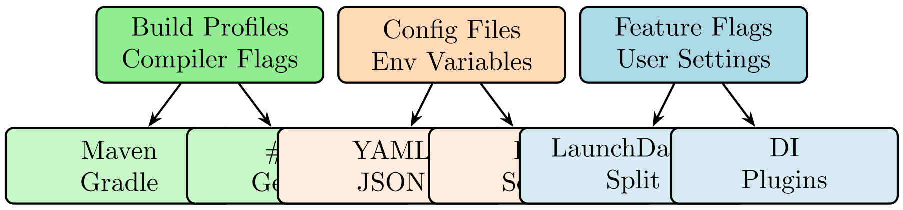


Figure 6: Configuration Mechanisms by Binding Time

### 5.2 Configuration Mechanism Catalog

Table 8: Configuration Mechanism Catalog

| Mechanism | Binding Time | Use Cases | Implementation |
|---|---|---|---|
| Build Profiles | Compile | Platform variants; optional modules | Maven/Gradle profiles |
| Conditional Compilation | Compile | Platform-specific code; debug code | Preprocessor; annotations |
| Dependency Injection | Startup | Service implementations; strategies | Spring; Guice; CDI |
| Configuration Files | Deploy/Startup | Environment settings; credentials | YAML; JSON; Properties |
| Environment Variables | Deploy | Container config; secrets | Docker ENV; K8s ConfigMaps |
| Feature Flags | Runtime | Gradual rollout; A/B testing | LaunchDarkly; Split; Flagsmith |
| Plugin Architecture | Runtime | Extensions; integrations | OSGi; SPI; Module systems |

| **Mechanism** | **Binding Time** | **Use Cases** | **Implementation** |
|---|---|---|---|
| User Preferences | Runtime | Personalization; UI settings | Database; local storage |
| Tenant Configuration | Runtime | Multi-tenant customization | Database per tenant |
| Dynamic Discovery | Runtime | Service instances; resources | Consul; Eureka; K8s Services |

## 5.3 Configuration File Specifications

**Primary Configuration: application.yml**

**Purpose:** Central configuration for deployment-time variability
**Location:** `config/application.yml`, overridden by environment-specific files

```
# Application Configuration Schema
app:
  name: ecommerce-platform
  environment: ${APP_ENV:development}

# VP-01: Database Provider
database:
  provider: ${DB_PROVIDER:postgresql}  # postgresql, mysql, sqlserver
  host: ${DB_HOST:localhost}
  port: ${DB_PORT:5432}
  name: ${DB_NAME:ecommerce}
  pool:
    min: ${DB_POOL_MIN:5}
    max: ${DB_POOL_MAX:20}

# VP-02: Authentication Method
auth:
  provider: ${AUTH_PROVIDER:oidc}  # oidc, saml, ldap, local
  oidc:
    issuer: ${OIDC_ISSUER}
    client-id: ${OIDC_CLIENT_ID}
    client-secret: ${OIDC_CLIENT_SECRET}
  saml:
    metadata-url: ${SAML_METADATA_URL}
    entity-id: ${SAML_ENTITY_ID}

# VP-04: Search Engine
search:
  provider: ${SEARCH_PROVIDER:elasticsearch}
  elasticsearch:
    hosts: ${ES_HOSTS:localhost:9200}
    index-prefix: ${ES_INDEX_PREFIX:ecom}
  algolia:
    app-id: ${ALGOLIA_APP_ID}
    api-key: ${ALGOLIA_API_KEY}

# VP-07: Cache Strategy
cache:
  provider: ${CACHE_PROVIDER:redis}
  redis:
    host: ${REDIS_HOST:localhost}
    port: ${REDIS_PORT:6379}
    ttl-seconds: ${CACHE_TTL:3600}

# VP-05: Notification Channels (multi-select)
notifications:
  channels:
    email:
      enabled: ${NOTIFY_EMAIL_ENABLED:true}
      provider: ${EMAIL_PROVIDER:sendgrid}
    sms:
      enabled: ${NOTIFY_SMS_ENABLED:false}
      provider: ${SMS_PROVIDER:twilio}
    push:
```

## 5.4 Feature Flag Configuration

**Feature Flag Configuration**

**Purpose:** Runtime variability for gradual rollout and experimentation
**Provider:** LaunchDarkly

```
# Feature Flag Definitions (feature-flags.yml)
flags:
  new-checkout-flow:
    key: new-checkout-flow
    name: "New Checkout Experience"
    description: "Redesigned checkout with fewer steps"
    type: boolean
    default: false
    targeting:
      - environment: production
        rules:
          - percentage: 10  # 10% of users
          - segments: [beta-testers]  # All beta testers
      - environment: staging
        default: true  # Always on in staging
    metrics:
      - conversion-rate
      - checkout-completion-time

  ai-recommendations:
    key: ai-recommendations
    name: "AI Product Recommendations"
    description: "ML-powered product recommendations"
    type: boolean
    default: false
    targeting:
      - environment: production
        rules:
          - segments: [premium-tier]  # Premium customers only
    prerequisites:
      - flag: data-collection-consent
        value: true

  payment-methods:
    key: payment-methods
    name: "Enabled Payment Methods"
    description: "Available payment options"
    type: multivariate
    variants:
      - value: [stripe, paypal]
        name: "Standard"
      - value: [stripe, paypal, crypto]
        name: "With Crypto"
      - value: [stripe, paypal, affirm]
        name: "With BNPL"
    default: [stripe, paypal]
```

**Flag Evaluation Code:**

```
@Service
public class FeatureFlagService {
    private final LDClient ldClient;

    public boolean isEnabled(String flagKey, User user) {
```

## 5.5 Plugin and Extension Mechanism

**Plugin Architecture**

**Purpose:** Extensibility for custom integrations and third-party extensions
**Mechanism:** Java SPI (Service Provider Interface) + Spring auto-configuration

```java
// Plugin interface definition
public interface PaymentGatewayPlugin {
    String getGatewayId();
    String getDisplayName();
    boolean supportsCountry(String countryCode);
    PaymentResult processPayment(PaymentRequest request);
    RefundResult processRefund(RefundRequest request);
}

// Plugin registration (META-INF/services)
// File: META-INF/services/com.platform.PaymentGatewayPlugin
com.stripe.StripePaymentGateway
com.paypal.PayPalPaymentGateway
com.custom.CustomPaymentGateway

// Plugin discovery and loading
@Configuration
public class PaymentGatewayConfiguration {
    @Bean
    public PaymentGatewayRegistry gatewayRegistry() {
        ServiceLoader<PaymentGatewayPlugin> loader =
            ServiceLoader.load(PaymentGatewayPlugin.class);

        PaymentGatewayRegistry registry = new PaymentGatewayRegistry
           ();
        for (PaymentGatewayPlugin plugin : loader) {
            registry.register(plugin.getGatewayId(), plugin);
            log.info("Registered payment gateway: {}",
                     plugin.getDisplayName());
        }
        return registry;
    }
}
```

**Plugin Discovery Locations:**

- `/plugins/` directory scanned at startup
- Maven/Gradle dependencies with `@AutoConfiguration`
- Database-registered plugins for runtime loading

**Plugin Lifecycle:**

1. Discovery: Plugins found via SPI or classpath scanning
2. Validation: Plugin compatibility and security verification
3. Registration: Plugin added to registry
4. Activation: Plugin enabled via configuration
5. Monitoring: Plugin health and usage tracked

# 6 Constraints and Invariants

## 6.1 Constraint Types

Constraints restrict which combinations of variants are valid. Proper constraint documentation prevents invalid configurations.

Table 9: Constraint Types

| Type | Description | Example |
|---|---|---|
| Requires | One variant requires another | Crypto payments requires non-built-in tax calculator |
| Excludes | Two variants cannot coexist | LDAP auth excludes OAuth social login |
| Implies | Selecting one implies another | Enterprise tier implies OIDC or SAML auth |
| Cardinality | Limits on number of selections | Maximum 3 payment gateways active |
| Conditional | Context-dependent constraints | AWS deployment requires S3 storage |
| Temporal | Time-based constraints | Cannot change database provider mid-contract |

## 6.2 Constraint Specification

Table 10: Configuration Constraints Registry

| ID | Type | Constraint | Rationale / Validation |
|---|---|---|---|
| CON-01 | Requires | VP-03.Crypto $\rightarrow$ VP-11 $\neq$ BuiltIn | Crypto requires proper tax calculation for regulatory compliance |
| CON-02 | Excludes | VP-02.LDAP $\times$ VP-02.SocialLogin | LDAP enterprise environments typically prohibit social login |
| CON-03 | Implies | Enterprise Tier $\rightarrow$ VP-02 $\in$ {OIDC, SAML} | Enterprise customers require SSO integration |
| CON-04 | Cardinality | $\lvert$VP-03 selections$\rvert$ $\leq 3$ | Performance and maintenance limits |
| CON-05 | Conditional | AWS $\rightarrow$ VP-06 $\in$ {S3, Aurora} | AWS deployment optimized for AWS services |

*Continued on next page*

| ID | Type | Constraint | Rationale / Validation |
|---|---|---|---|
| CON-06 | Requires | VP-04.Algolia → VP-07.Redis | Algolia sync requires Redis for queue management |
| CON-07 | Excludes | VP-01.MySQL × VP-10.Crystal | Crystal Reports has known MySQL compatibility issues |
| CON-08 | Requires | VP-08.AiRecommendations → DataConsent=true | ML features require data collection consent |
| CON-09 | Cardinality | \|VP-05 selections\| ≥ 1 | At least one notification channel required |
| CON-10 | Temporal | VP-01 cannot change during contract | Database migration disruptive; plan at renewal |

## 6.3 Constraint Visualization

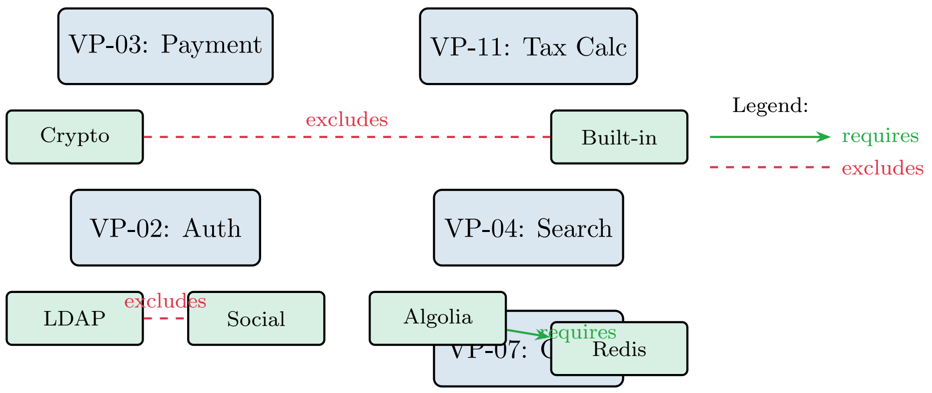


Figure 7: Constraint Relationships Visualization

## 6.4 Constraint Validation

Listing 1: Configuration Validation Implementation

```java
@Component
public class ConfigurationValidator {

    public ValidationResult validate(SystemConfiguration config) {
        List<ConstraintViolation> violations = new ArrayList<>();

        // CON-01: Crypto requires non-built-in tax
        if (config.paymentGateways.contains("crypto")
            && config.taxCalculator.equals("builtin")) {
            violations.add(new ConstraintViolation("CON-01",
                "Crypto payments require external tax calculator"));
        }
        // CON-02: LDAP excludes social login
        if (config.authProvider.equals("ldap")
            && config.socialLoginEnabled) {
            violations.add(new ConstraintViolation("CON-02",
                "LDAP auth is incompatible with social login"));
        }

        // CON-04: Max 3 payment gateways
        if (config.paymentGateways.size() > 3) {
            violations.add(new ConstraintViolation("CON-04",
                "Maximum 3 payment gateways allowed"));
        }

        // CON-06: Algolia requires Redis
        if (config.searchProvider.equals("algolia")
            && !config.cacheProvider.equals("redis")) {
            violations.add(new ConstraintViolation("CON-06",
                "Algolia search requires Redis cache"));
        }

        return new ValidationResult(violations);
    }
}
```

# 7 Configuration Examples and Scenarios

## 7.1 Product Configuration Profiles

Define standard configuration profiles for common deployment scenarios.

**Configuration Profile: Startup Edition**

**Target:** Small businesses, startups, individual sellers
**Characteristics:** Low cost, simple setup, limited scale

**Variation Point Selections:**

| Variation Point | Selected Variant | Rationale |
|---|---|---|
| VP-01: Database | PostgreSQL (shared) | Cost-effective; sufficient for scale |
| VP-02: Authentication | Local Auth | Simple; no SSO needed |
| VP-03: Payment Gateway | Stripe only | Single provider; lower complexity |
| VP-04: Search | Native (built-in) | Adequate for small catalogs |
| VP-05: Notifications | Email only | Basic notifications sufficient |
| VP-06: Storage | S3 (shared bucket) | Cost-effective object storage |
| VP-07: Cache | Local (in-memory) | Sufficient for single instance |
| VP-08: Feature Flags | Basic set only | Limited feature set |
| VP-11: Tax Calculator | Built-in (US only) | Simple tax rules |

**Infrastructure:**

- Single application instance
- Shared PostgreSQL database
- Shared S3 bucket with tenant prefix
- No Redis required

**Limitations:**

- Maximum 10,000 products
- Maximum 1,000 orders/day
- US market only (tax limitation)
- No SSO integration

**Configuration File:**

```
edition: startup
database:
  provider: postgresql
  multi-tenant: shared
auth:
  provider: local
payment:
  gateways: [stripe]
search:
  provider: native
```

**Configuration Profile: Enterprise Edition**

**Target:** Large enterprises, high-volume retailers
**Characteristics:** High scale, full features, enterprise integration

**Variation Point Selections:**

| Variation Point | Selected Variant | Rationale |
|---|---|---|
| VP-01: Database | Aurora (dedicated) | High availability; auto-scaling |
| VP-02: Authentication | SAML + OIDC | Enterprise SSO integration |
| VP-03: Payment Gateway | Stripe + PayPal + Adyen | Global coverage; redundancy |
| VP-04: Search | Elasticsearch (dedicated) | High-volume catalog support |
| VP-05: Notifications | All channels | Full communication capability |
| VP-06: Storage | S3 (dedicated bucket) | Isolation; compliance |
| VP-07: Cache | Redis Cluster | High availability caching |
| VP-08: Feature Flags | Full access + custom | All features; A/B testing |
| VP-11: Tax Calculator | Avalara | Global tax compliance |

**Infrastructure:**

- Multi-AZ deployment (3+ instances)
- Dedicated Aurora cluster with read replicas
- Dedicated Elasticsearch cluster
- Redis Cluster with 6 nodes
- Dedicated S3 bucket with encryption

**Capabilities:**

- Unlimited products
- 100,000+ orders/day
- Global market support
- Full SSO integration
- 99.95% SLA

**Configuration File:**

```
edition: enterprise
database:
  provider: aurora
  multi-tenant: dedicated
  replicas: 2
auth:
  provider: saml
  fallback: oidc
  mfa: required
payment:
  gateways: [stripe, paypal, adyen]
  failover: true
search:
  provider: elasticsearch
  cluster-size: 3
cache:
  provider: redis
```

### 7.2 Migration Scenarios

Document how to transition between configurations.

**Example**

**Migration Scenario: Startup to Professional Edition**
**Trigger:** Customer exceeds 10,000 products or 1,000 orders/day
**Changes Required:**

1. VP-01: Migrate from shared to dedicated PostgreSQL
2. VP-04: Migrate from native search to Elasticsearch
3. VP-07: Add Redis cache layer
4. VP-03: Add PayPal as second payment option

**Migration Steps:**

1. Provision dedicated PostgreSQL instance
2. Replicate data from shared database
3. Deploy Elasticsearch and index catalog
4. Update configuration to point to new services
5. Enable Redis cache
6. Switch traffic to new configuration
7. Verify functionality
8. Decommission shared resources

**Downtime:** Zero (blue-green deployment)
**Duration:** 2-4 hours (automated)
**Rollback:** Revert DNS; maintain old configuration for 24 hours

## 8 Traceability

### 8.1 Variability to Requirements Traceability

Table 11: Variation Points to Requirements Mapping

| VP ID | Requirements | Features/Capabilities |
|---|---|---|
| VP-01 | REQ-NFR-001 (Performance), REQ-NFR-003 (Scalability) | Platform flexibility |
| VP-02 | REQ-SEC-001 (Authentication), REQ-SEC-005 (SSO) | Enterprise SSO, Security |

| VP ID | Requirements | Features/Capabilities |
|---|---|---|
| VP-03 | REQ-FUNC-042 (Payments), REQ-FUNC-045 (Multi-currency) | Payment processing |
| VP-04 | REQ-FUNC-010 (Product Search), REQ-NFR-002 (Search Performance) | Catalog search |
| VP-05 | REQ-FUNC-060 (Notifications), REQ-FUNC-062 (Multi-channel) | Customer communication |
| VP-08 | REQ-OPS-010 (Feature Control), REQ-PROD-001 (A/B Testing) | Release management |
| VP-11 | REQ-COMPL-001 (Tax Compliance), REQ-FUNC-050 (Tax Calculation) | Tax handling |

## 8.2 Variability to Implementation Traceability

Table 12: Variation Points to Implementation Mapping

| VP ID | Implementation Artifacts | Configuration | Tests |
|---|---|---|---|
| VP-01 | `src/data/providers/` | `database.*` | `DatabaseProviderTest` |
| VP-02 | `src/auth/providers/` | `auth.*` | `AuthIntegrationTest` |
| VP-03 | `src/payments/gateways/` | `payment.gateways` | `PaymentGatewayTest` |
| VP-04 | `src/search/engines/` | `search.*` | `SearchEngineTest` |
| VP-07 | `src/cache/providers/` | `cache.*` | `CacheProviderTest` |
| VP-08 | `src/common/features/` | LaunchDarkly | `FeatureFlagTest` |

## 8.3 Variability to Deployment Traceability

Table 13: Variation Points to Deployment Mapping

| VP ID | Infrastructure | Kubernetes Resources | Monitoring |
|---|---|---|---|
| VP-01 | RDS / Aurora / CloudSQL | `database-secret` | DB connection metrics |

| VP ID | Infrastructure | Kubernetes Resources | Monitoring |
|---|---|---|---|
| VP-02 | Identity provider | `auth-config` | Auth success/failure rates |
| VP-03 | Payment gateway accounts | `payment-secrets` | Transaction success rates |
| VP-04 | Elasticsearch cluster | `search-config` | Search latency, indexing |
| VP-06 | S3 / GCS bucket | `storage-secret` | Storage usage, latency |
| VP-07 | Redis cluster | `redis-config` | Cache hit rates, memory |

# 9 Variability Governance

## 9.1 Variability Management Process

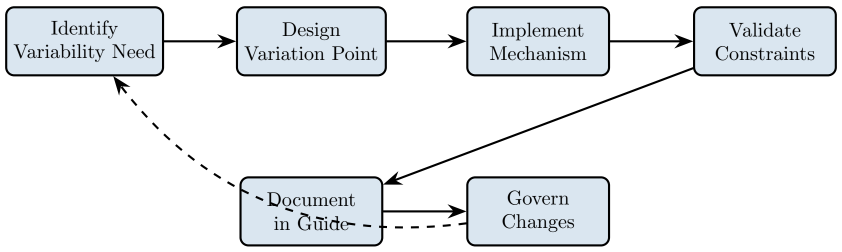


Figure 8: Variability Management Lifecycle

## 9.2 Change Management for Variability

Table 14: Variability Change Categories

| Change Type | Examples | Approval | Impact |
|---|---|---|---|
| Add Variant | New payment gateway | Tech Lead | Low |
| Remove Variant | Deprecate old provider | Architecture Board | Medium |
| Add VP | New variation point | Architecture Board | High |
| Remove VP | Eliminate variability | Architecture Board | High |

| Change Type | Examples | Approval | Impact |
|---|---|---|---|
| Modify Constraint | Change compatibility rules | Tech Lead | Medium |
| Change Binding Time | Move from deploy to runtime | Architecture Board | High |

### 9.3 Version History

Table 15: Variability Guide Version History

| Version | Date | Author | Changes |
|---|---|---|---|
| 1.0.0 | 2024-01-15 | Platform Team | Initial variability guide |
| 1.1.0 | 2024-02-20 | J. Smith | Added VP-08 (Feature Flags) |
| 1.2.0 | 2024-03-15 | A. Jones | Added Crypto payment variant to VP-03 |
| 1.3.0 | 2024-04-10 | Platform Team | Added Enterprise configuration profile |
| 2.0.0 | 2024-06-01 | Architecture Team | Major revision; added constraint validation |

## 10 Open Issues and Future Variability

### 10.1 Pending Variability Decisions

Table 16: Pending Variability Issues

| ID | Issue | Impact | Target Date | Owner |
|---|---|---|---|---|
| OI-01 | GraphQL API variant needed? | New VP required | Q3 2024 | API Team |
| OI-02 | Multi-region deployment variants | VP-01, VP-06, VP-07 affected | Q4 2024 | Platform |
| OI-03 | Mobile SDK variability | May need new VPs | Q3 2024 | Mobile Team |
| OI-04 | White-label customization depth | UI variability scope | Q2 2024 | Product |

### 10.2 Planned Variability Extensions

Table 17: Planned Variability Additions

| Variation Point | Description | Variants Planned | Timeline |
|---|---|---|---|
| VP-13: CDN Provider | Content delivery network selection | CloudFront, Cloudflare, Fastly | Q3 2024 |
| VP-14: ML Platform | Machine learning infrastructure | SageMaker, Vertex AI, Self-hosted | Q4 2024 |
| VP-15: Workflow Engine | Business process automation | Temporal, Camunda, Built-in | Q4 2024 |

## 11 Appendix A: Feature Model Notation

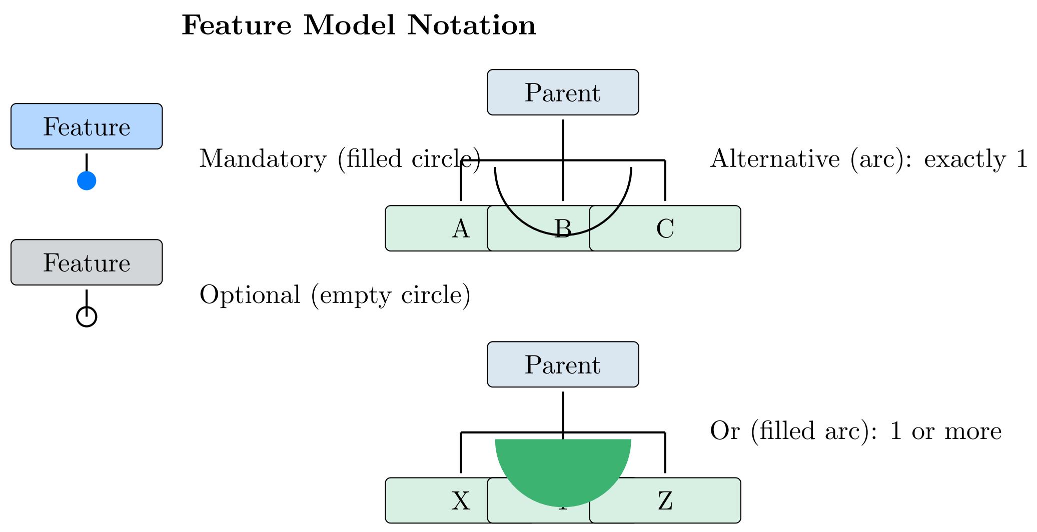

Feature Model Notation
Feature
Mandatory (filled circle)
Parent
A
B
C
Alternative (arc): exactly 1
Feature
Optional (empty circle)
Parent
X
Y
Z
Or (filled arc): 1 or more

Figure 9: Feature Model Notation Reference

## 12 Appendix B: Configuration Checklist

### 12.1 New Variation Point Checklist

- ☐ Variation point ID assigned
- ☐ Location in architecture identified
- ☐ Binding time determined and justified
- ☐ All variants documented
- ☐ Default variant specified

- ☐ Selection rules defined
- ☐ Constraints with other VPs identified
- ☐ Configuration mechanism implemented
- ☐ Validation logic added
- ☐ Tests for all variants created
- ☐ Documentation updated

### 12.2 Configuration Validation Checklist

- ☐ All required variation points have selections
- ☐ No constraint violations
- ☐ Selected variants are compatible
- ☐ Required dependencies satisfied
- ☐ Infrastructure requirements met
- ☐ License requirements satisfied
- ☐ Security requirements met for selected variants
- ☐ Performance impact assessed

## 13 Appendix C: Glossary

| | |
|---|---|
| **Binding** | The act of selecting a specific variant for a variation point |
| **Binding Time** | The point in the lifecycle when a variation point is resolved |
| **Cardinality** | The number of variants that can/must be selected |
| **Configuration** | A complete set of variant selections for all variation points |
| **Constraint** | A rule that restricts valid combinations of variants |
| **Feature Flag** | A runtime mechanism for enabling/disabling features |
| **Feature Model** | A hierarchical representation of features and their relationships |
| **Product Line** | A set of products sharing a common architecture and assets |
| **Variant** | A specific alternative that can be selected at a variation point |
| **Variation Point** | A location in the architecture where variation can occur |
| **Variability** | The ability of a system to be configured, customized, or extended |

# 14 Appendix D: References

1. Pohl, K., Böckle, G., & van der Linden, F. (2005). *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer.

2. Clements, P., & Northrop, L. (2001). *Software Product Lines: Practices and Patterns*. Addison-Wesley.

3. Kang, K., et al. (1990). "Feature-Oriented Domain Analysis (FODA) Feasibility Study." SEI Technical Report CMU/SEI-90-TR-021.

4. Czarnecki, K., & Eisenecker, U. (2000). *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley.

5. Apel, S., et al. (2013). *Feature-Oriented Software Product Lines: Concepts and Implementation*. Springer.

6. Hodgson, P. (2017). "Feature Toggles (aka Feature Flags)." Martin Fowler's Website. Retrieved from `https://martinfowler.com/articles/feature-toggles.html`

7. ISO/IEC 26550:2015. "Software and systems engineering – Reference model for product line engineering and management."

8. Clements, P., et al. (2010). *Documenting Software Architectures: Views and Beyond* (2nd ed.). Addison-Wesley.