金融系统架构与IT治理深度实践指南 v3.0

架构师与业务分析师的协作实战手册

Enterprise Financial System Architecture & IT Governance *架构驱动业务价值，业务引领技术方向* --- **版本**: v3.0 专业优化版 **适用范围**: 金融企业架构师、业务分析师、技术负责人、IT治理专家、产品经理 **适用行业**: 银行、证券、保险、金融科技、支付清算、消费金融 **目标读者**: 具备3年以上金融IT从业经验的专业人士

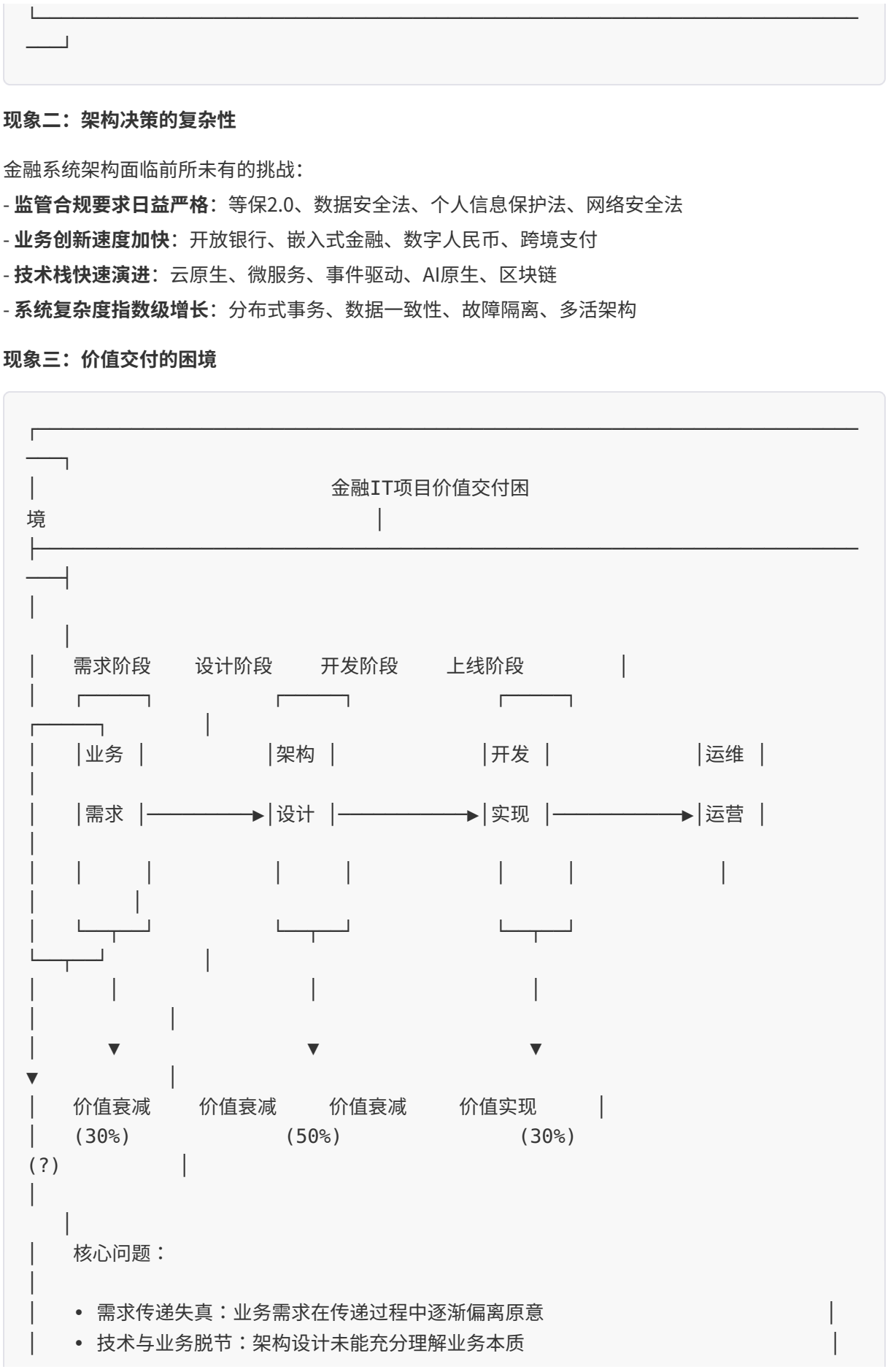
目录

前言：写给架构师和业务分析师

为什么需要这本指南

在金融数字化转型的浪潮中，我们观察到三个关键现象：

现象一：技术与业务的鸿沟

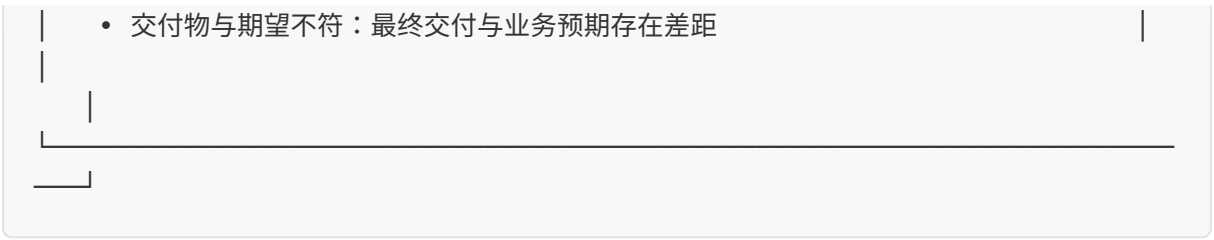


现象二：架构决策的复杂性

金融系统架构面临前所未有的挑战：

- **监管合规要求日益严格**：等保2.0、数据安全法、个人信息保护法、网络安全法
- **业务创新速度加快**：开放银行、嵌入式金融、数字人民币、跨境支付
- **技术栈快速演进**：云原生、微服务、事件驱动、AI原生、区块链
- **系统复杂度指数级增长**：分布式事务、数据一致性、故障隔离、多活架构

现象三：价值交付的困境

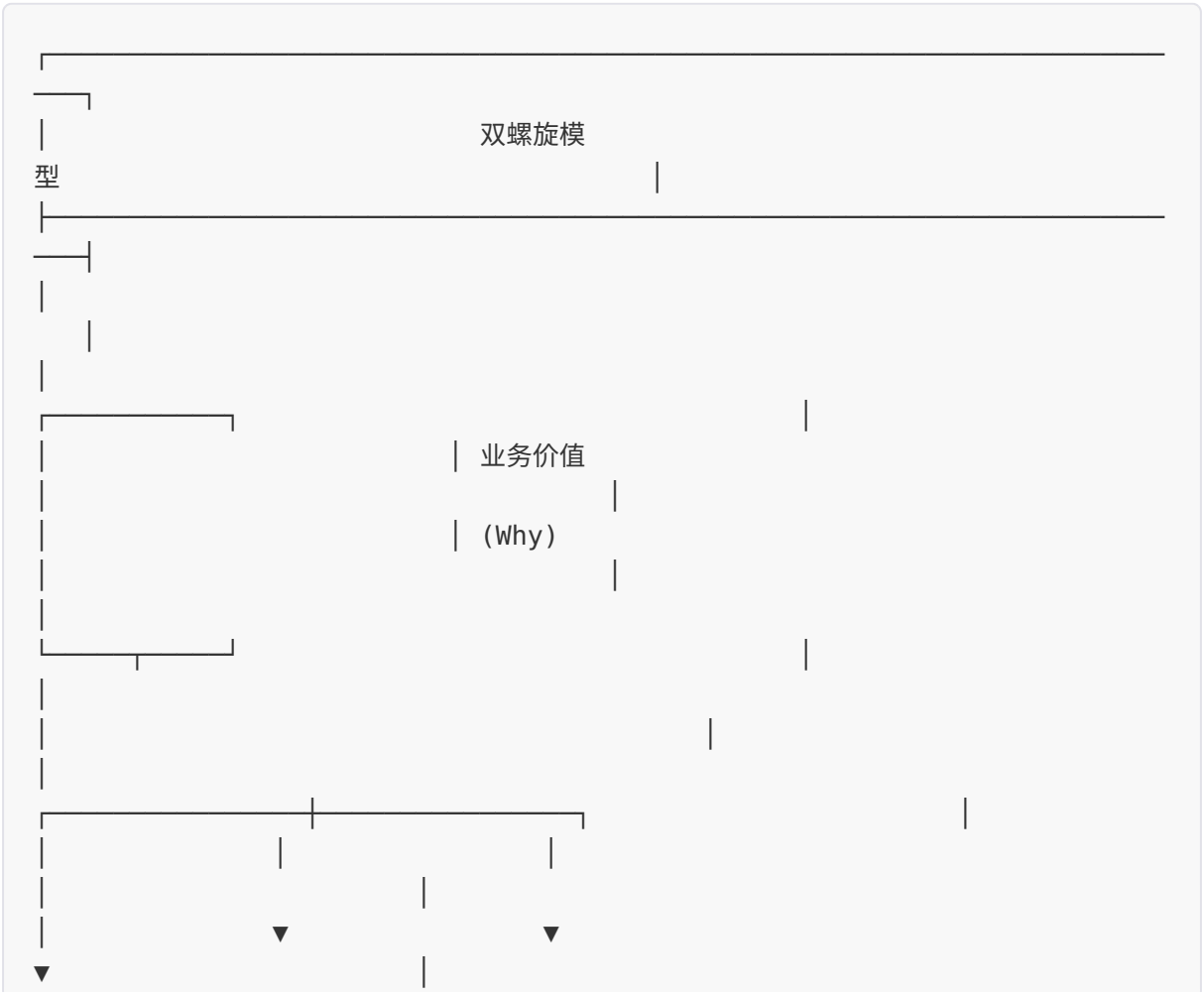


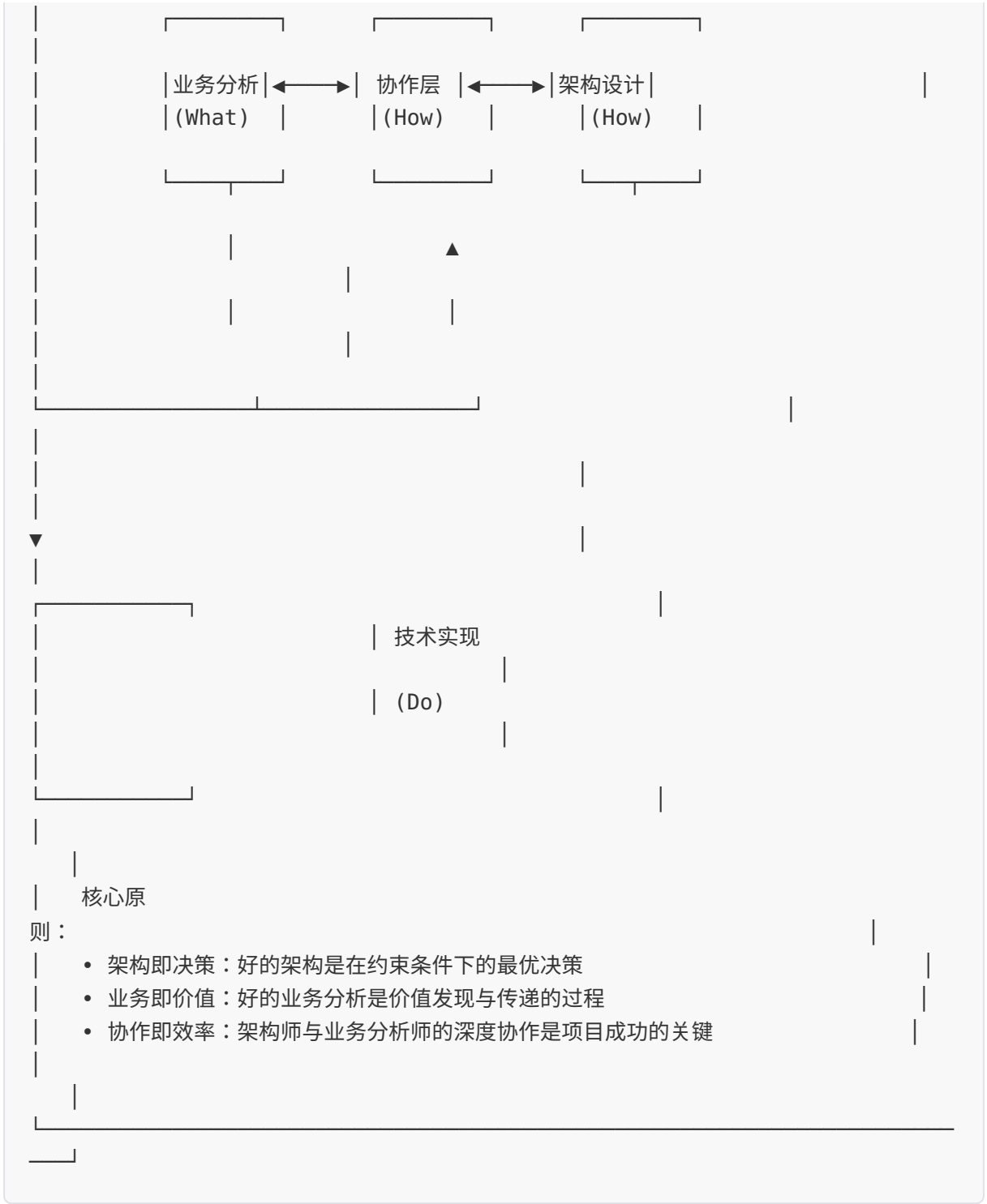
本指南的定位

本指南不是一本简单的技术手册或业务分析教程，而是：

1. **架构师的决策参考**：提供系统化的架构设计方法、决策框架和最佳实践
2. **业务分析师的建模指南**：提供从业务到技术的完整映射方法和价值分析框架
3. **跨角色的沟通桥梁**：建立统一的词汇体系、思维模式和协作机制
4. **实战落地的工具箱**：提供可直接使用的模板、检查清单和工具
5. **持续改进的知识库**：汇集业界最佳实践，支持持续学习和改进

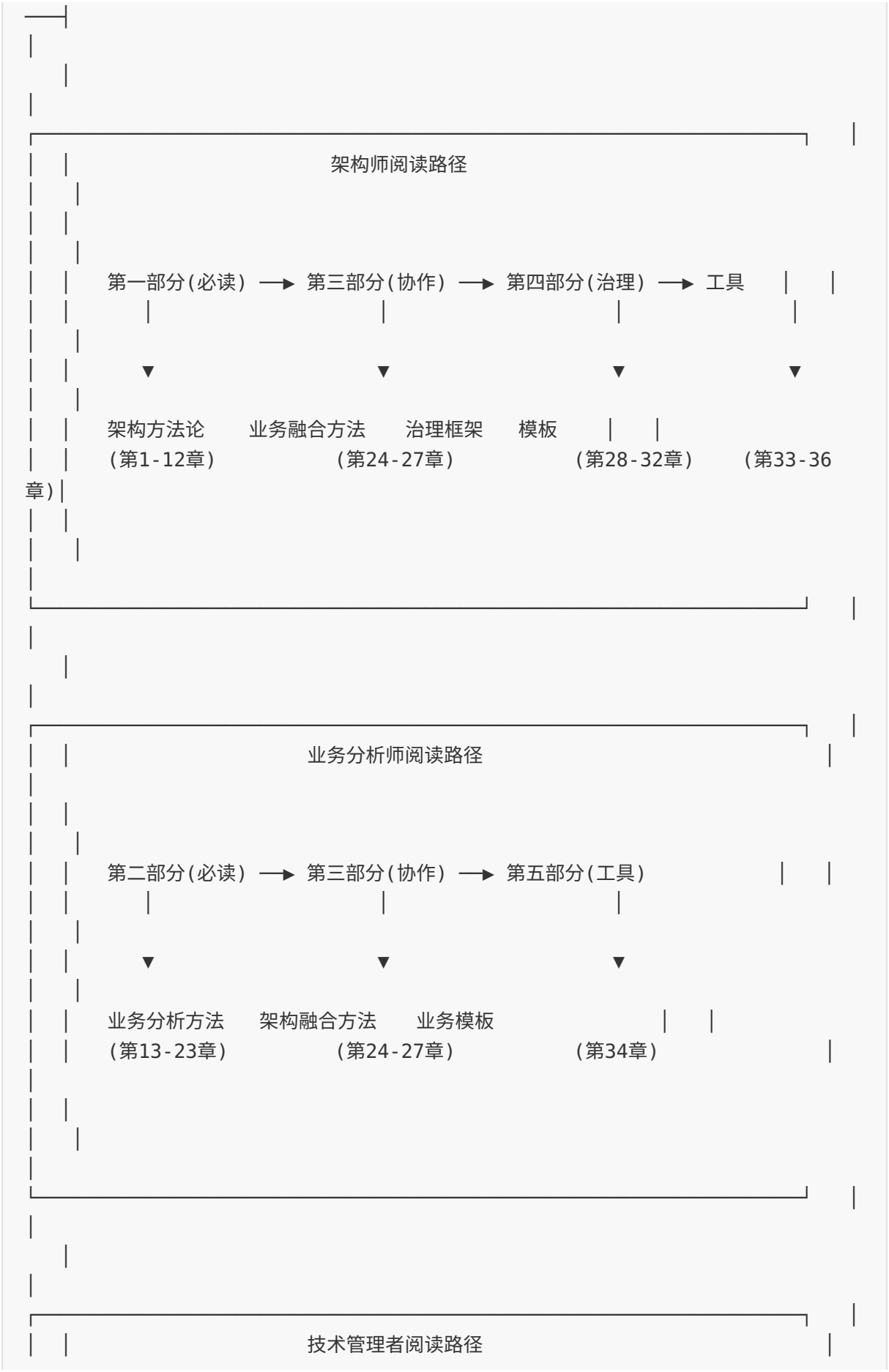
核心理念

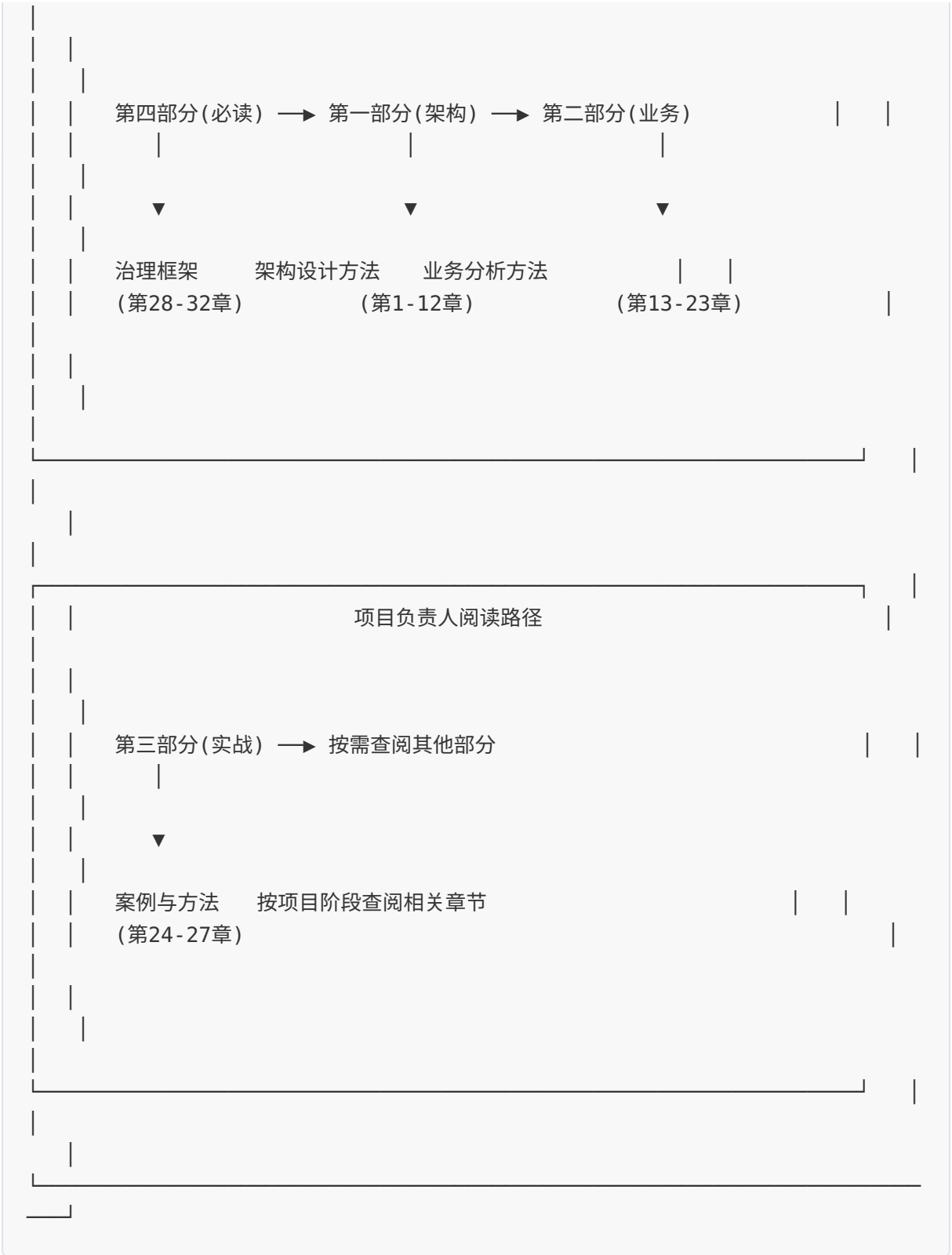



业务分析
(What)
协作层
(How)
架构设计
(How)
技术实现
(Do)
核心原则：
• 架构即决策：好的架构是在约束条件下的最优决策
• 业务即价值：好的业务分析是价值发现与传递的过程
• 协作即效率：架构师与业务分析师的深度协作是项目成功的关键

如何使用本指南


使用路径推荐


架构师阅读路径
第一部分(必读)
第三部分(协作)
第四部分(治理)
工具
架构方法论
(第1-12章)
业务融合方法
(第24-27章)
治理框架
模板
(第28-32章)
(第33-36章)
业务分析师阅读路径
第二部分(必读)
第三部分(协作)
第五部分(工具)
业务分析方法
(第13-23章)
架构融合方法
(第24-27章)
业务模板
(第34章)
技术管理者阅读路径



致谢与声明

本指南综合了以下框架、标准和最佳实践：

架构框架与标准：
- TOGAF 10 - The Open Group Architecture Framework
- ArchiMate 3.2 - Enterprise Architecture Modeling Language
- C4 Model - 软件架构可视化模型
- SABSA - Sherwood Applied Business Security Architecture
- ISO/IEC/IEEE 42010 - 系统和软件工程架构描述

业务分析框架：
- BABOK v3 - Business Analysis Body of Knowledge
- BIZBOK - Business Architecture Body of Knowledge
- BPMN 2.0 - Business Process Model and Notation
- DMN 1.3 - Decision Model and Notation
- CBOK - Common Body of Knowledge for Business Analysis

金融行业标准：
- 等保2.0 - 网络安全等级保护
- JR/T 系列金融行业技术标准
- Basel III - 巴塞尔协议
- PCI DSS - 支付卡行业数据安全标准

工程实践：
- DDD - 领域驱动设计
- DevOps/DevSecOps
- SRE - 站点可靠性工程
- FinOps - 云财务管理

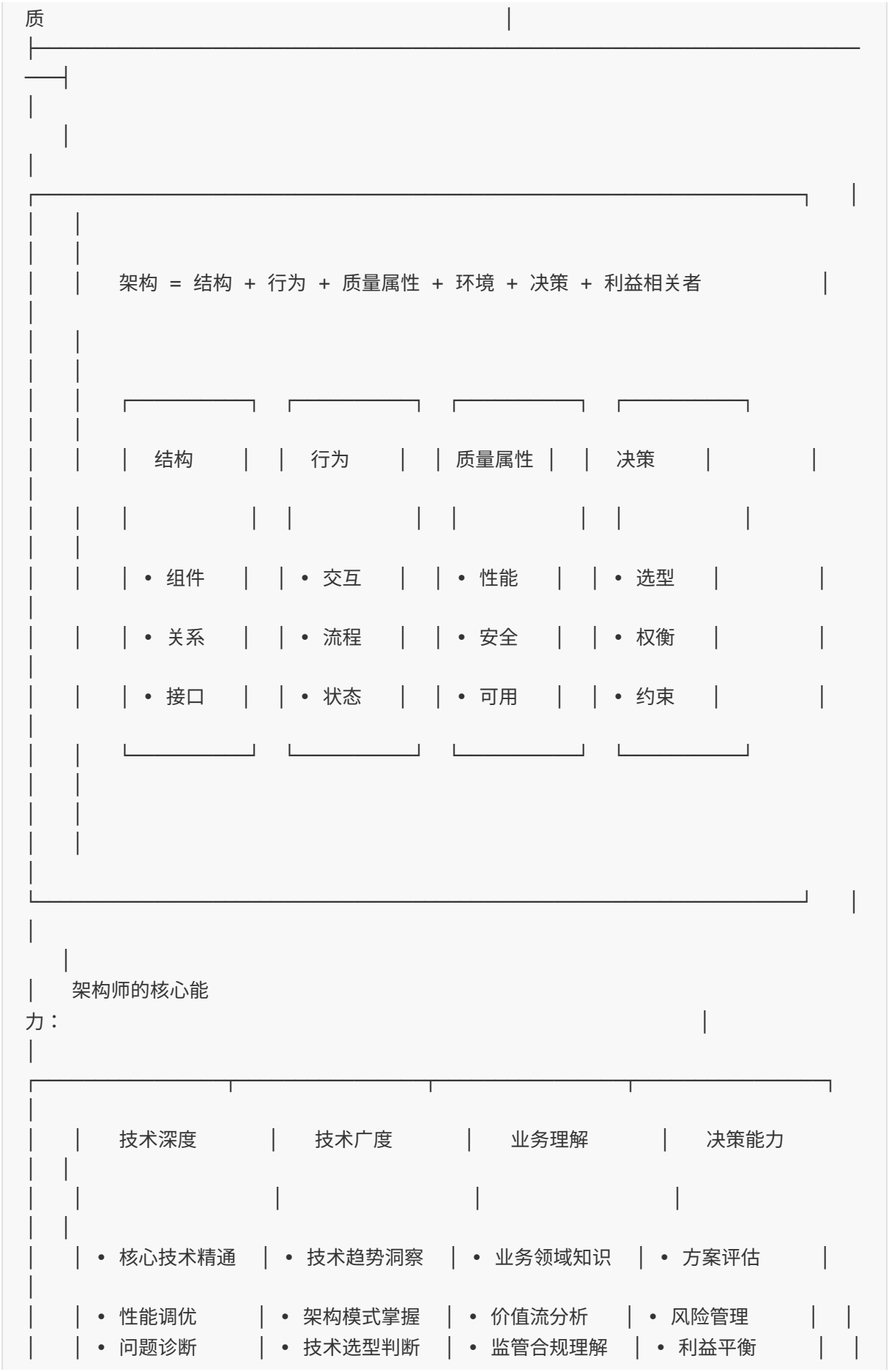
第一部分：架构师篇 - 金融系统架构设计

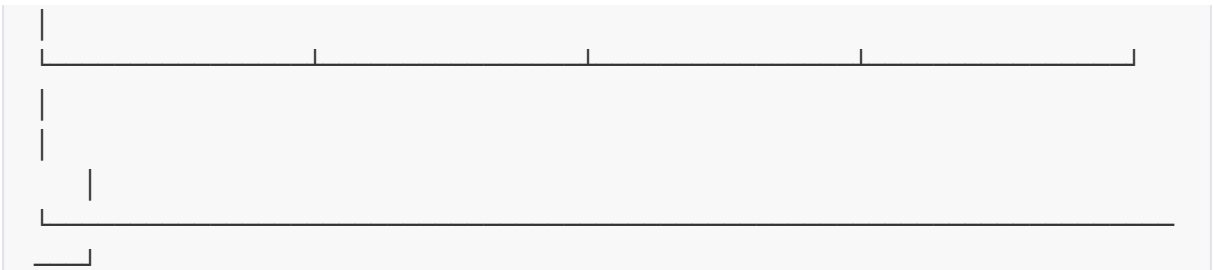
第1章 架构思维与方法论

1.1 什么是架构

架构是对系统关键设计决策的集合，这些决策一旦做出，就很难改变。架构师的核心职责是在各种约束条件下做出最优决策，平衡各方利益相关者的需求。

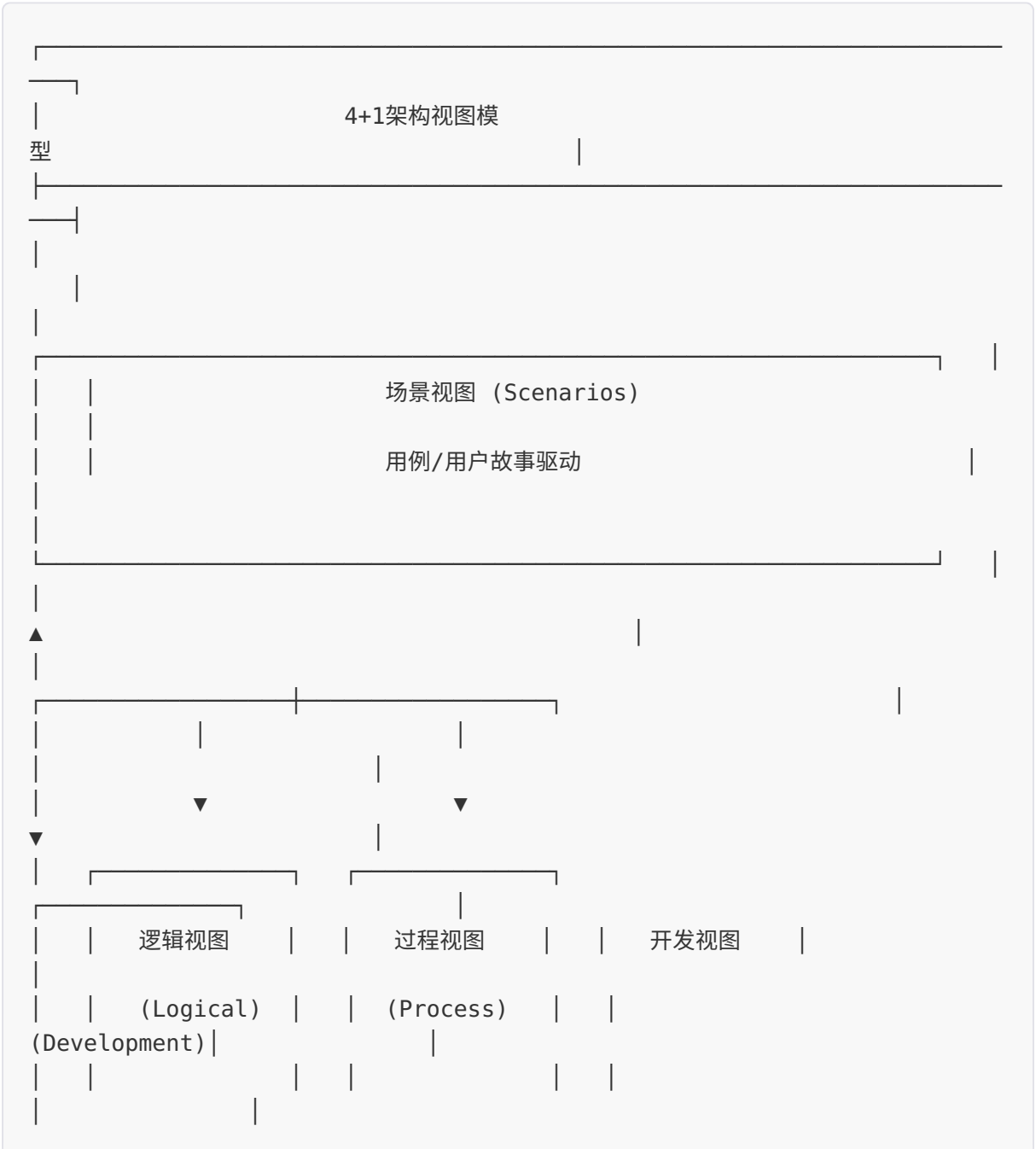
```
┌──────────────────────────────────────────────────────────────────────┐
│                                架构的本                              │
```

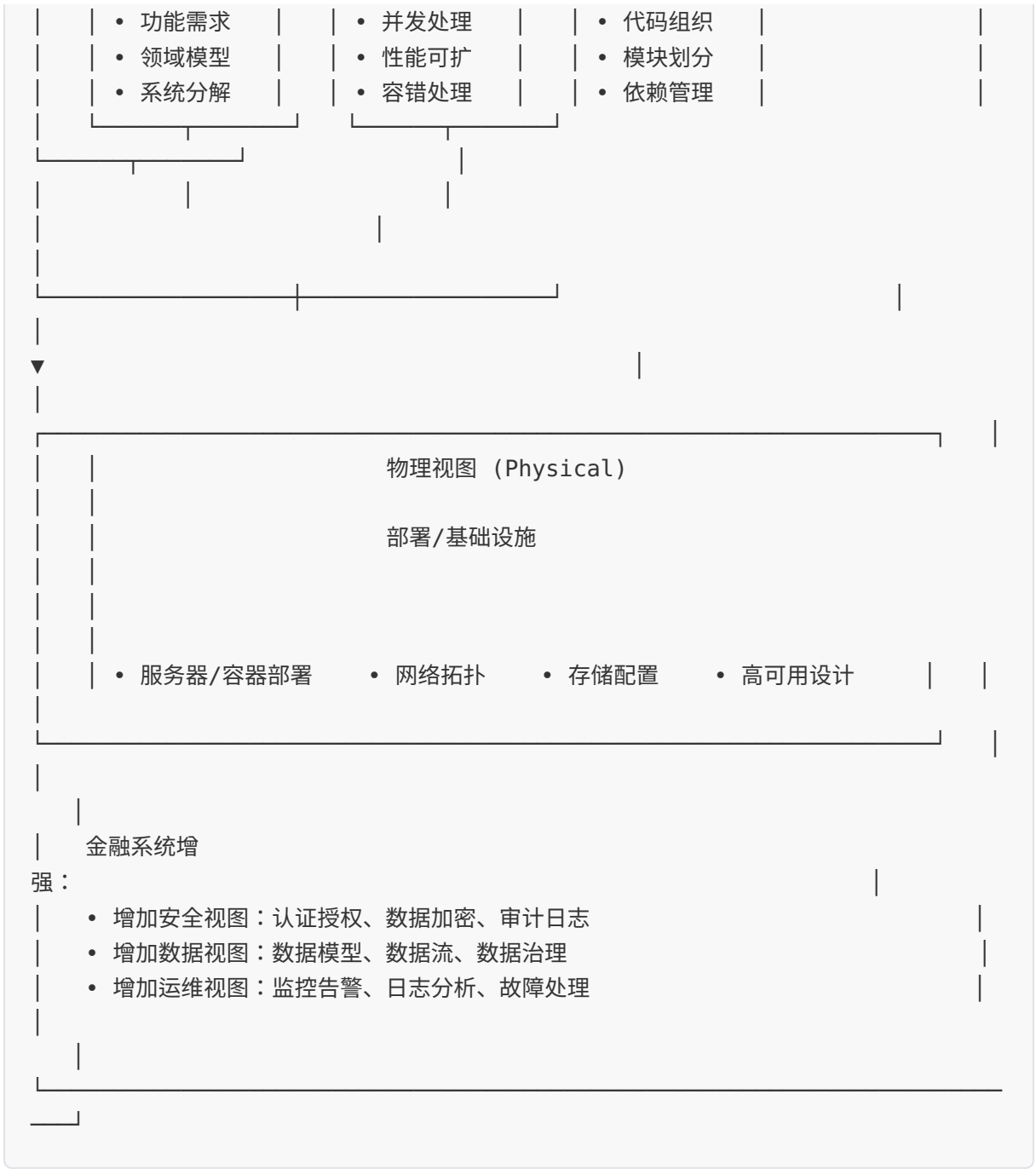

质
架构 = 结构 + 行为 + 质量属性 + 环境 + 决策 + 利益相关者
结构
行为
质量属性
决策
• 组件
• 关系
• 接口
• 交互
• 流程
• 状态
• 性能
• 安全
• 可用
• 选型
• 权衡
• 约束
架构师的核心能力：
技术深度
技术广度
业务理解
决策能力
• 核心技术精通
• 性能调优
• 问题诊断
• 技术趋势洞察
• 架构模式掌握
• 技术选型判断
• 业务领域知识
• 价值流分析
• 监管合规理解
• 方案评估
• 风险管理
• 利益平衡



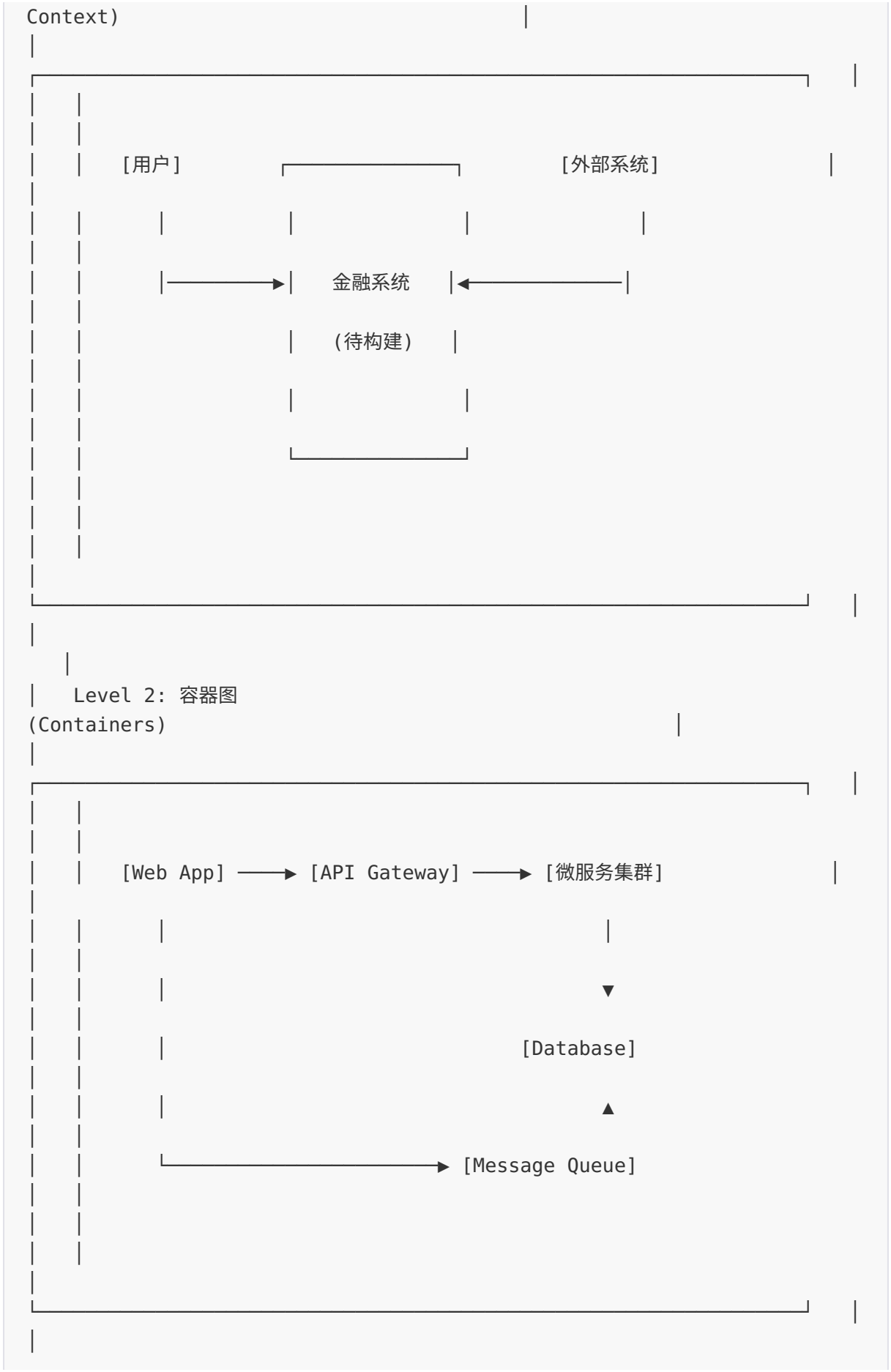
1.2 架构思维框架

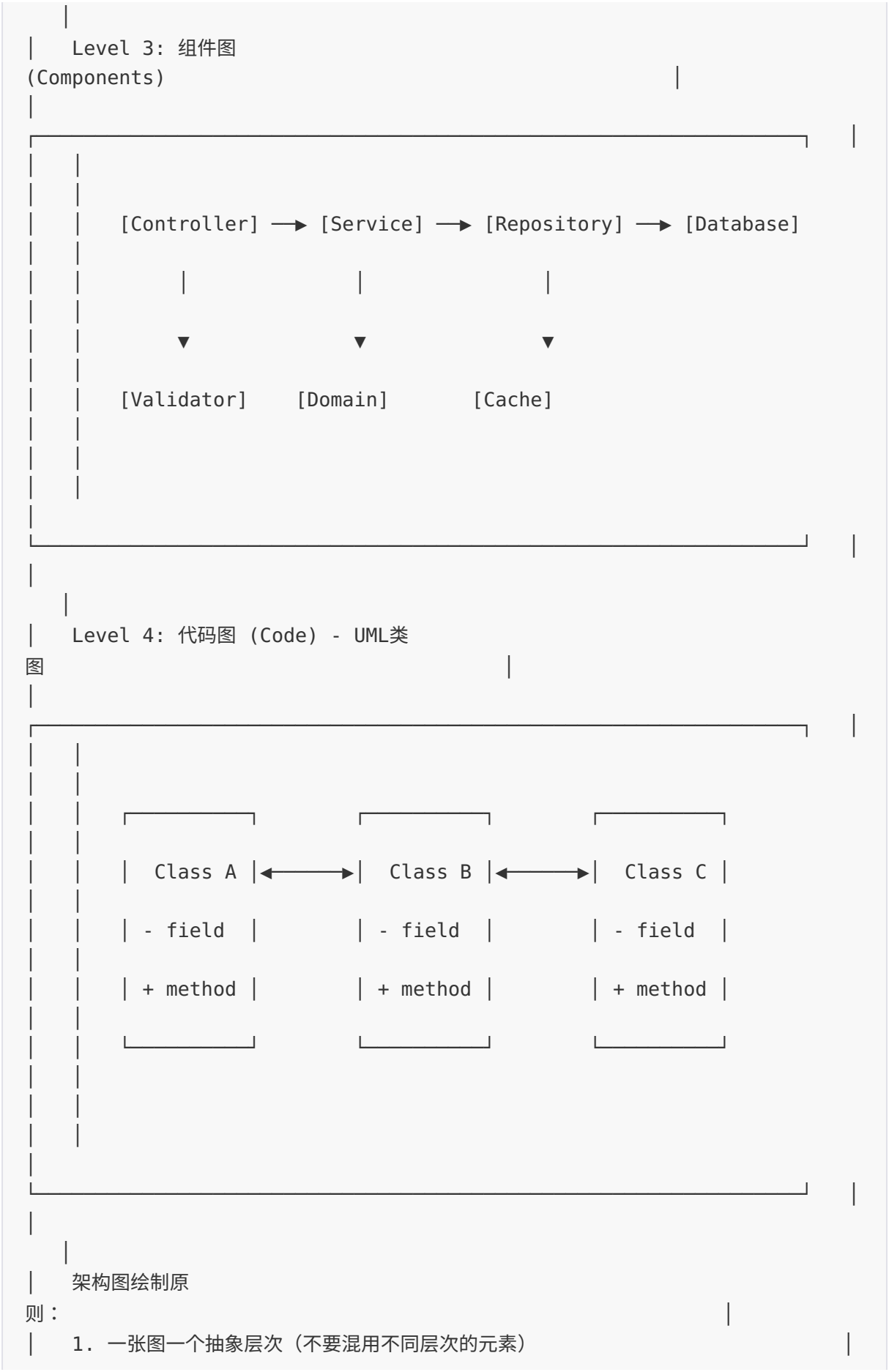
4+1视图模型（Kruchten模型）





C4模型（Simon Brown）

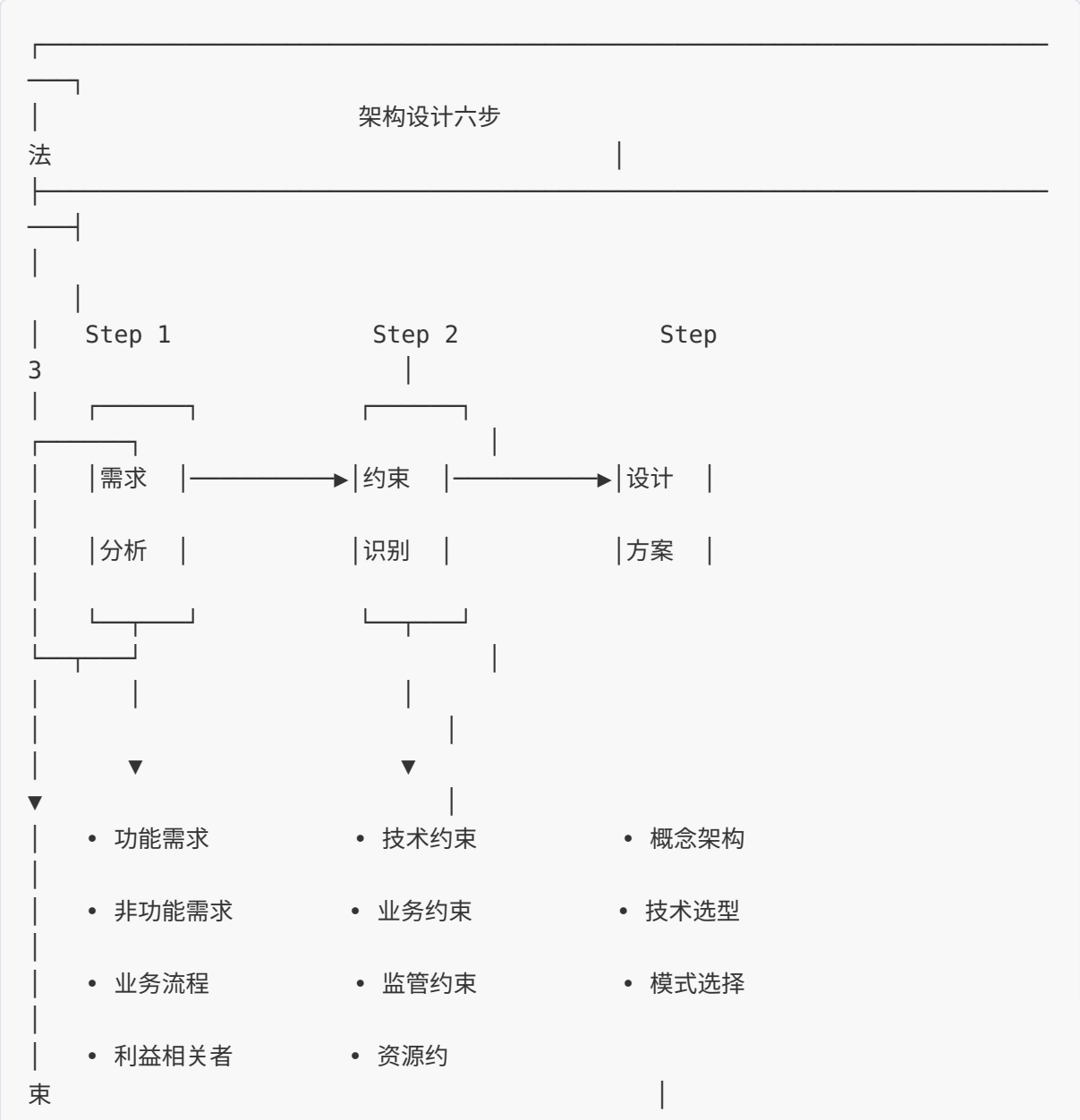

Context)
[用户]
[外部系统]
金融系统
(待构建)
Level 2: 容器图
(Containers)
[Web App]
[API Gateway]
[微服务集群]
[Database]
[Message Queue]

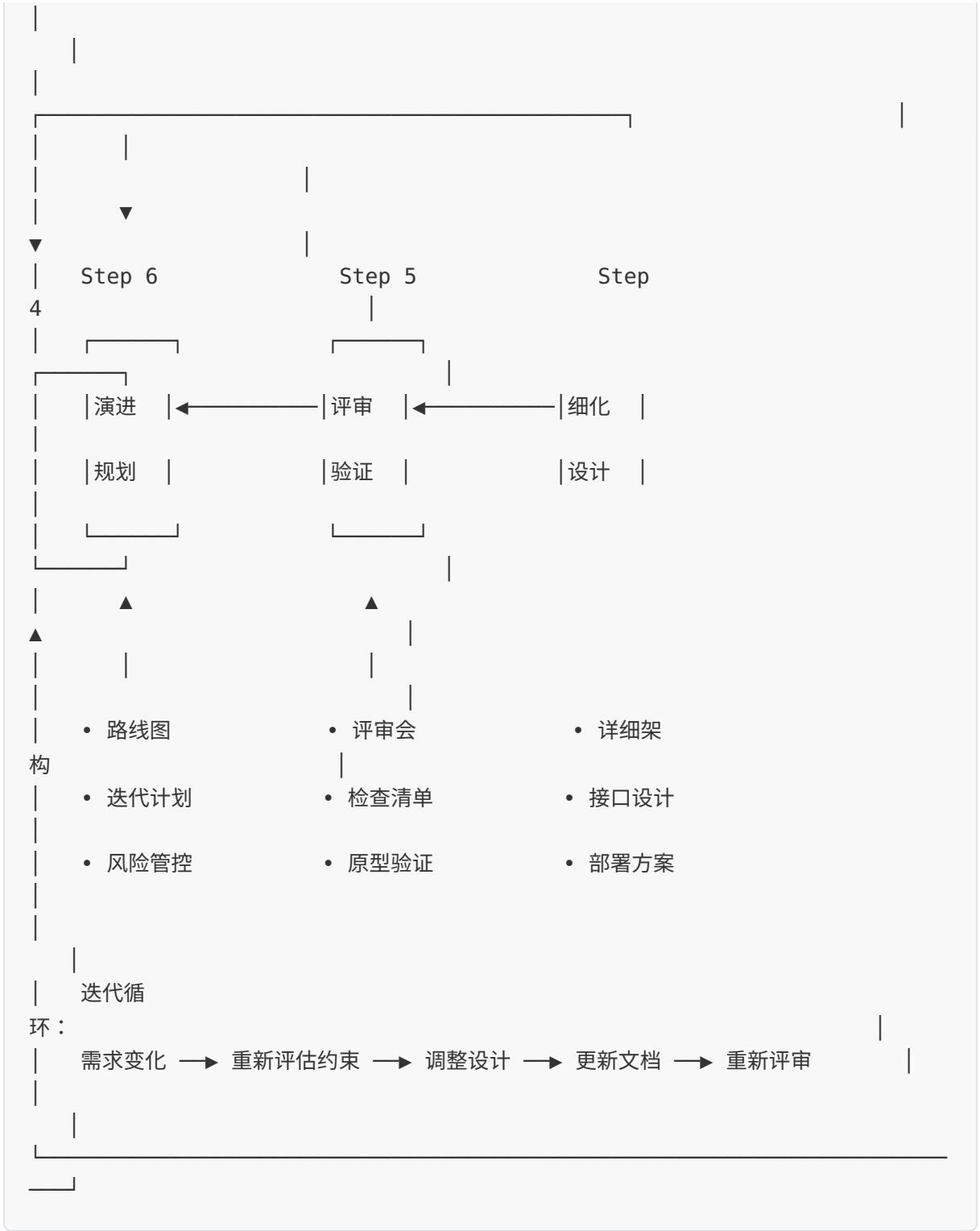

Level 3: 组件图
(Components)
[Controller] ⟶ [Service] ⟶ [Repository] ⟶ [Database]
[Validator]
[Domain]
[Cache]
Level 4: 代码图 (Code) - UML类
图
Class A
- field
+ method
Class B
- field
+ method
Class C
- field
+ method
架构图绘制原
则：
1. 一张图一个抽象层次（不要混用不同层次的元素）

```
│    2. 清晰标注技术选型（不要用通用词汇）
│
│    3. 展示关键架构决策（省略实现细节）
│
│    4. 考虑受众的认知背景（给业务人员看的图vs给开发人员看的图不同）                    │
│
   │
└──────────────────────────────────────────────────────────────────────────────
────┘
```

1.3 架构设计方法论

架构设计六步法




Step 6
Step 5
Step 4
演进
评审
细化
规划
验证
设计
• 路线图
• 评审会
• 详细架构
• 迭代计划
• 检查清单
• 接口设计
• 风险管控
• 原型验证
• 部署方案
迭代循环：
需求变化 → 重新评估约束 → 调整设计 → 更新文档 → 重新评审

架构设计思维模式

思维模式	描述	应用场景	金融示例
抽象思维	从具体中提取共性，形成概念模型	领域建模、组件设计	将各种支付渠道抽象为统一支付接口
分层思维	将系统划分为不同层次的职责	分层架构、关注点分离	表现层/业务层/数据层分离
分治思维	将大问题分解为可管理的小问题	系统拆分、微服务设计	核心系统按域拆分为微服务
演化思维	设计可演进的架构，预留扩展点	架构演进、技术债务管理	绞杀者模式替换遗留系统
权衡思维	在冲突的目标中寻找平衡	CAP权衡、成本效益分析	一致性与可用性的权衡
风险思维	识别风险并设计缓解措施	安全设计、灾备设计	两地三中心架构设计

第2章 金融系统架构基础

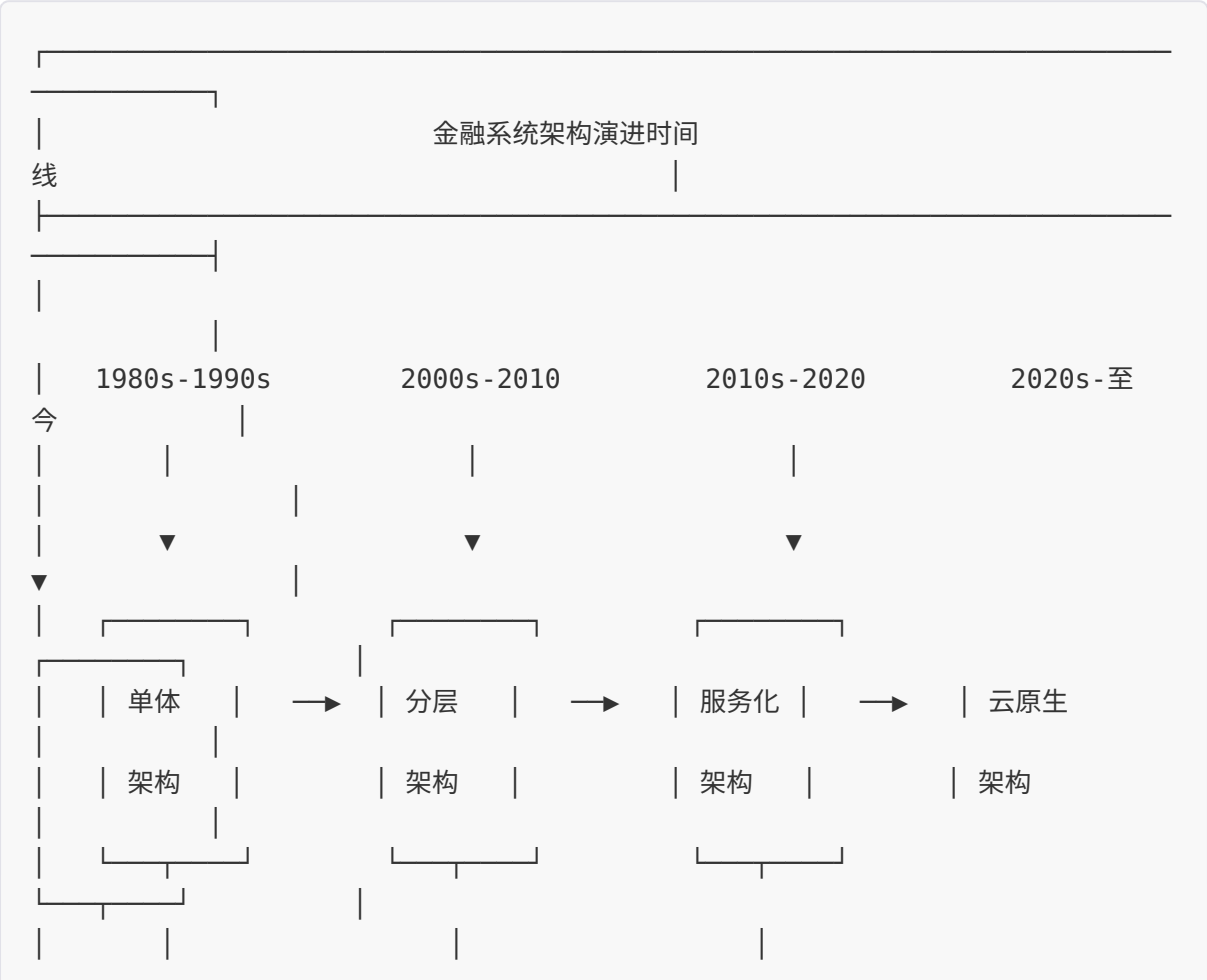
2.1 金融系统架构的特殊性

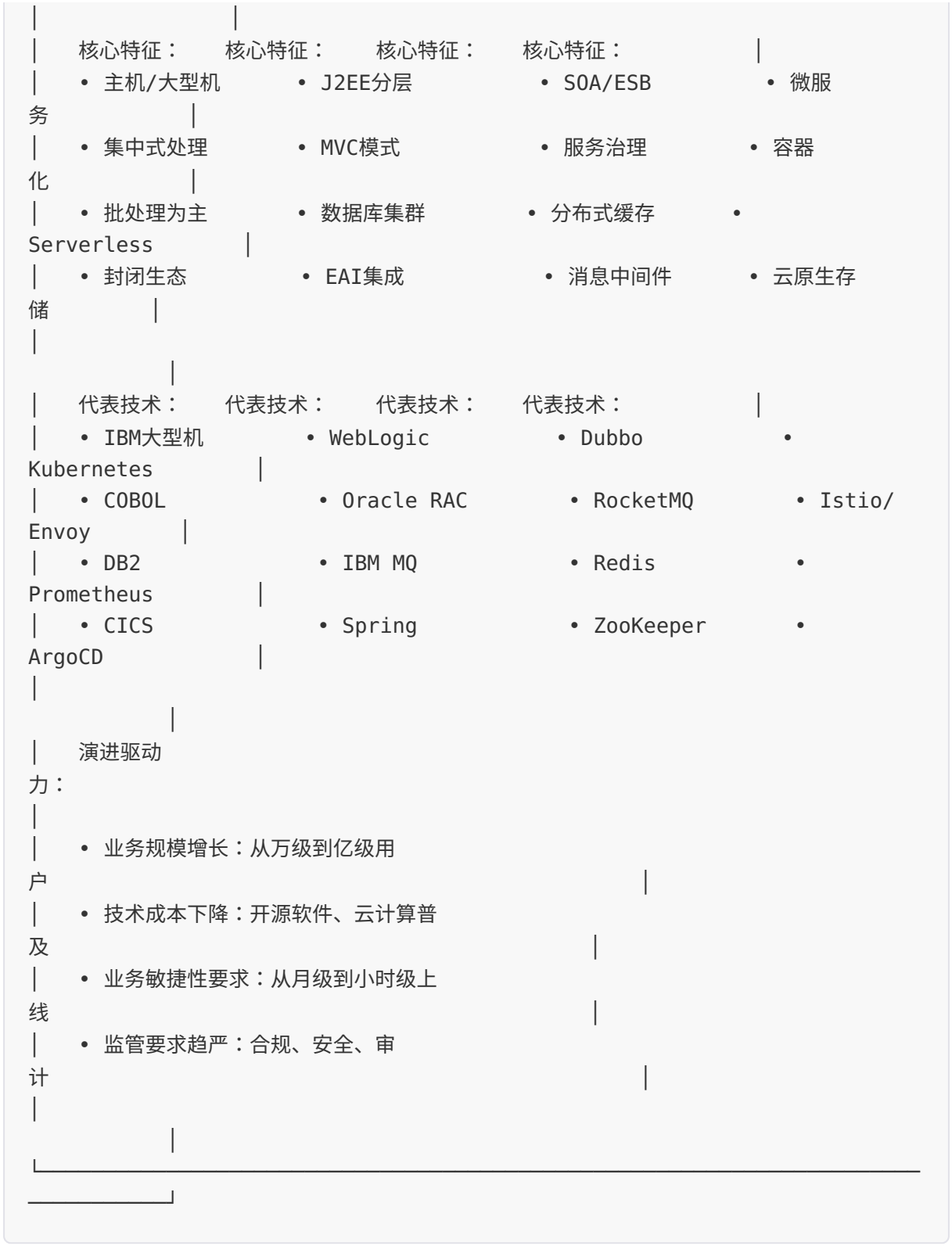
金融系统相比普通企业IT系统有以下显著特点：

维度	普通企业系统	金融系统	架构影响	设计策略
可用性要求	99.9%（年停机8.76小时）	99.999%（年停机5.26分钟）	需要多活架构、故障自动转移	同城双活、异地灾备
数据一致性	最终一致性可接受	强一致性要求（资金账务）	分布式事务、ACID保障	TCC/Saga模式
安全合规	等保2级	等保3-4级、金融监管要求	纵深防御、零信任架构	多层安全、审计追踪

维度	普通企业系统	金融系统	架构影响	设计策略
并发处理	百级TPS	万级-百万级TPS	分布式缓存、异步化、分片	读写分离、分库分表
延迟要求	秒级响应	毫秒级响应（交易）	内存计算、低延迟网络	缓存优先、异步处理
审计要求	基础日志	全流程可追溯、不可篡改	审计日志、区块链存证	日志归集、哈希校验
灾备要求	同城备份	两地三中心、RPO=0	多活架构、数据同步	同步复制、自动切换
监管报送	一般性报表	实时/准实时监管报送	数据标准化、接口规范	统一数据平台

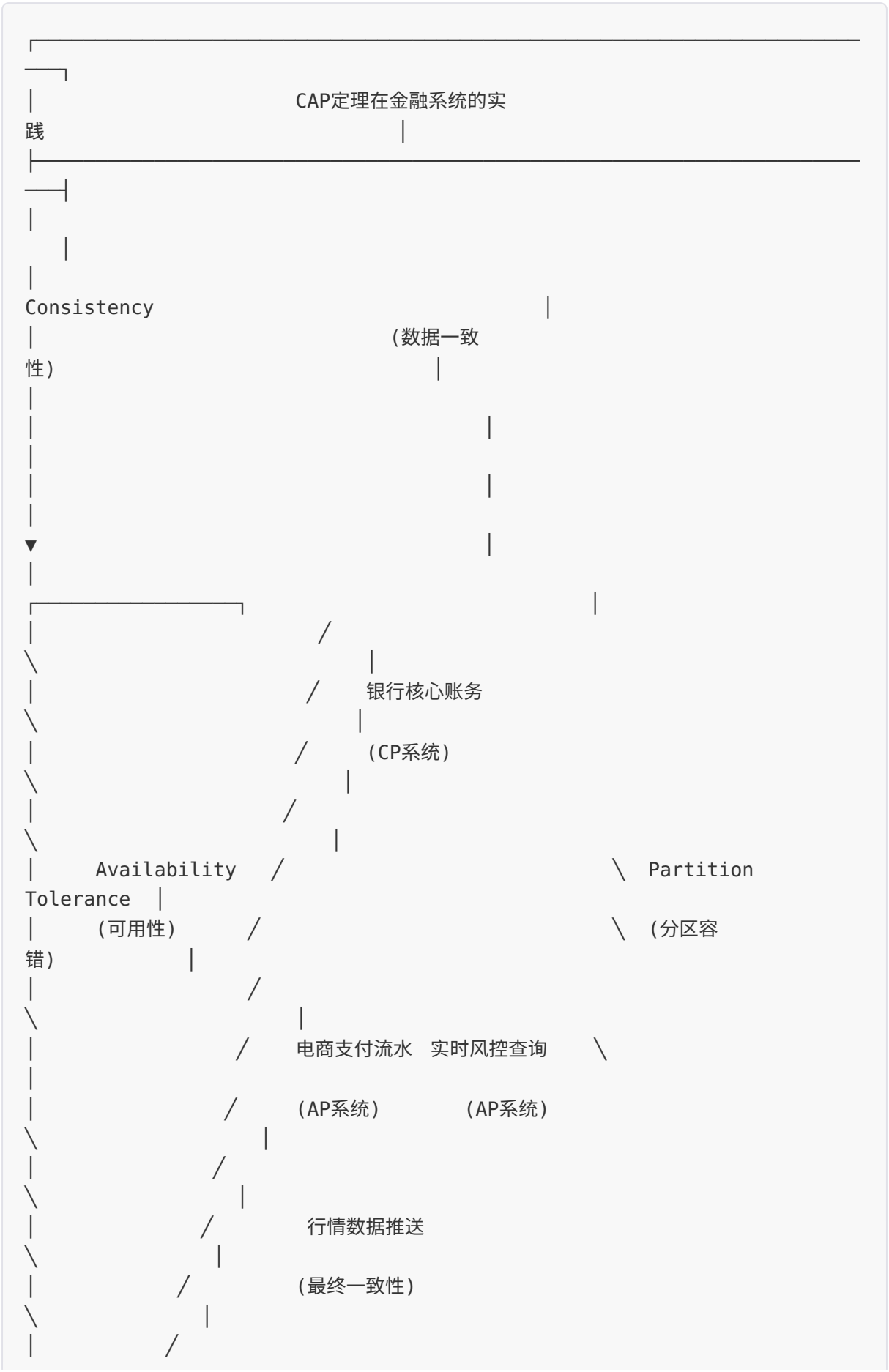
2.2 金融系统架构演进历程

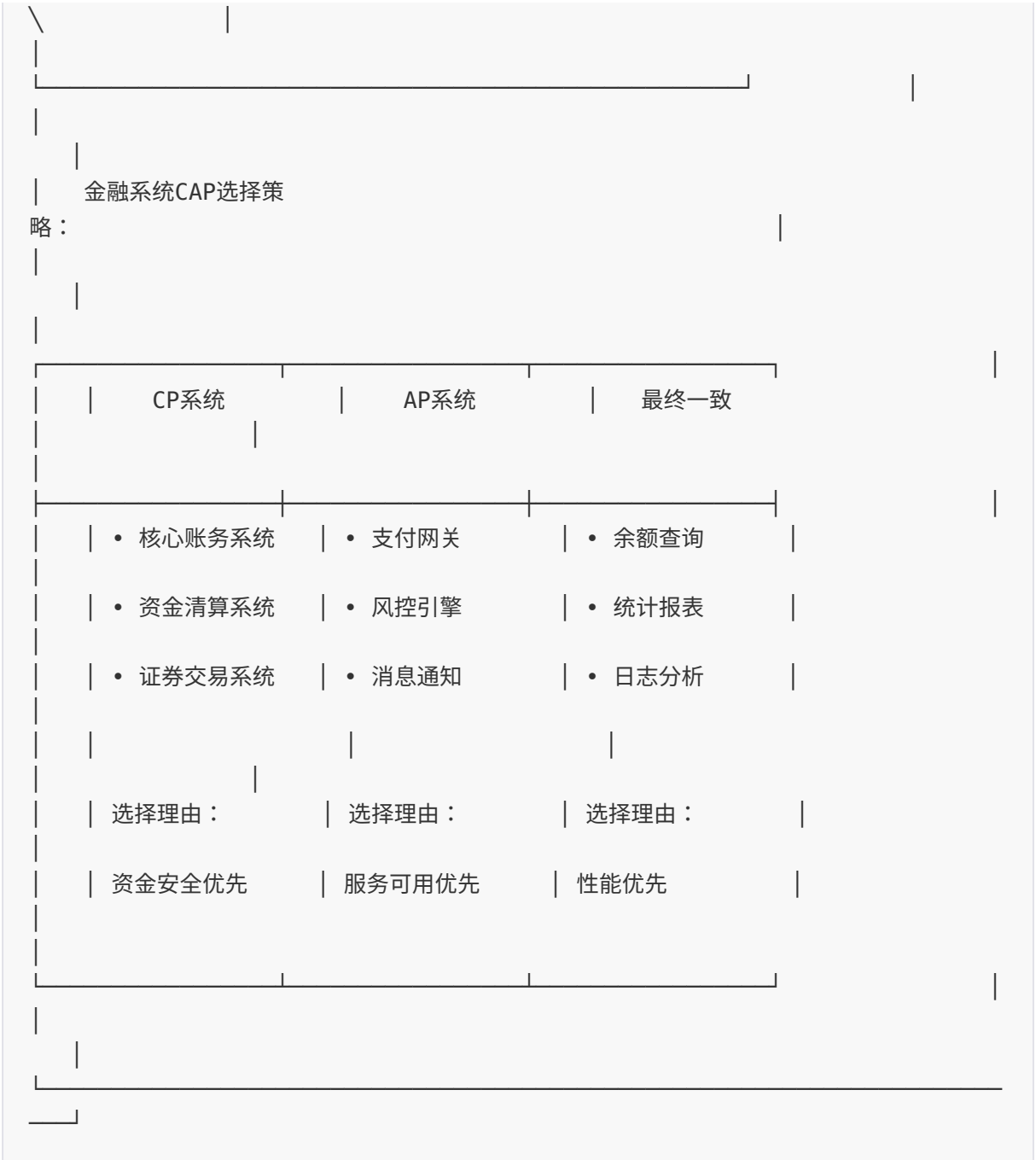




2.3 现代金融系统架构核心原则

CAP定理在金融系统的实践


CAP定理在金融系统的实践
Consistency
(数据一致性)
Availability
(可用性)
Partition Tolerance
(分区容错)
银行核心账务
(CP系统)
电商支付流水
实时风控查询
(AP系统)
(AP系统)
行情数据推送
(最终一致性)



金融架构设计十原则

原则编号	原则名称	核心描述	适用场景	实施要点
FA-01	安全优先原则	安全性设计优先于功能性设计	所有系统设计	安全左移、纵深防御
FA-02	数据主权原则	数据归属清晰，访问可控	客户数据管理	数据分类分级、最小权限

原则编号	原则名称	核心描述	适用场景	实施要点
FA-03	审计可追溯原则	所有操作留痕，可追溯、不可篡改	交易、配置变更	审计日志、时间戳、签名
FA-04	故障隔离原则	故障影响范围可控，单点故障可隔离	微服务设计	舱壁隔离、熔断限流
FA-05	优雅降级原则	部分故障时系统可降级服务	高峰期、故障场景	熔断、限流、兜底策略
FA-06	幂等性原则	重复操作结果一致	支付、转账	幂等键、状态机
FA-07	异步解耦原则	非关键路径异步化处理	通知、对账	消息队列、事件驱动
FA-08	容量预测原则	基于业务增长预测容量需求	扩容规划	容量模型、压测验证
FA-09	多活部署原则	关键系统多活部署	核心交易	同城双活、异地多活
FA-10	左移测试原则	测试左移，生产级质量内建	DevOps实践	自动化测试、混沌工程

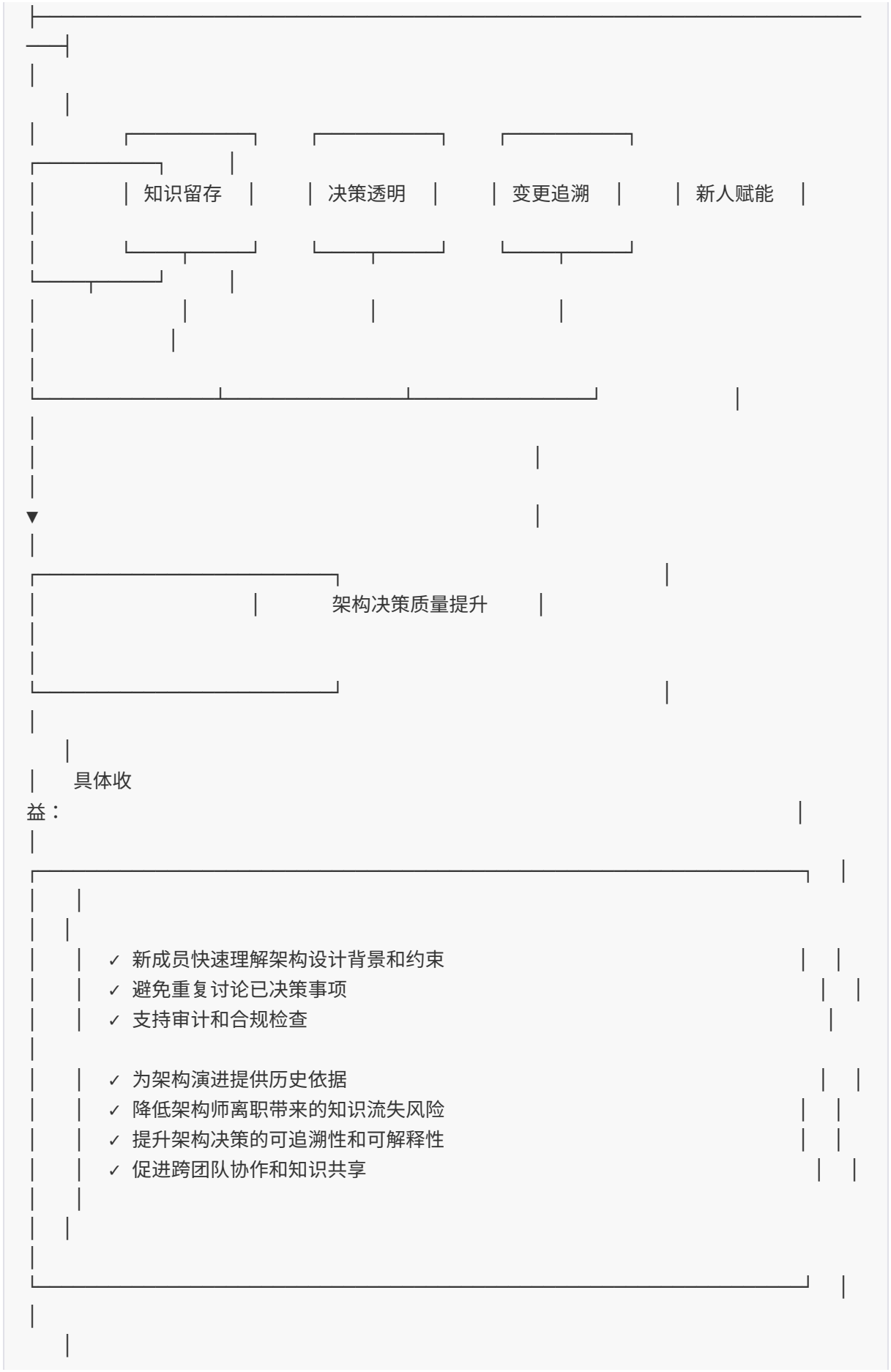
第3章 架构决策记录（ADR）

3.1 ADR概述与价值

什么是ADR

架构决策记录（Architecture Decision Records，ADR）是对架构设计中关键决策的文档化记录。在金融系统中，ADR是架构治理的核心交付物。

```
┌──────────────────────────────────────────────────────────────────┐
│                          ADR的核心价值                           │
```


知识留存
决策透明
变更追溯
新人赋能
架构决策质量提升
具体收益：
✓ 新成员快速理解架构设计背景和约束
✓ 避免重复讨论已决策事项
✓ 支持审计和合规检查
✓ 为架构演进提供历史依据
✓ 降低架构师离职带来的知识流失风险
✓ 提升架构决策的可追溯性和可解释性
✓ 促进跨团队协作和知识共享

```
└──────────────────────────────────────────────────────────────────────────────
────┘
```

3.2 ADR标准化模板

模板一：基础ADR模板（适用于一般决策）

```
# ADR-[编号]: [决策标题]

## 状态
- 建议 / 已接受 / 已拒绝 / 已废弃 / 已替代

## 背景
[描述需要做出决策的问题背景和业务驱动因素]

## 决策
[明确的决策陈述，使用命令式语句，如"我们将采用..."]

## 原因
[详细解释做出此决策的理由，包括业务价值、技术考量等]

## 后果
### 正面后果
-
### 负面后果/权衡
-

## 备选方案
### 备选A: [名称]
- 优点：
- 缺点：
- 未选择原因：

### 备选B: [名称]
- 优点：
- 缺点：
- 未选择原因：

## 相关决策
- 依赖于：ADR-xxx
- 被依赖：ADR-yyy

## 利益相关者
```

```
- [ ] 架构委员会审批
- [ ] 安全团队评审
- [ ] 运维团队确认
- [ ] 业务团队确认

## 决策人
[姓名] @ [日期]

## 修订历史
| 日期 | 版本 | 修改人 | 修改内容 |
|------|------|--------|----------|
|      |      |        |          |
```

模板二：金融增强型ADR模板（适用于重大架构决策）

```
# ADR-[编号]: [决策标题]

## 1. 元信息
| 属性 | 内容 |
|------|------|
| 编号 | ADR-xxx |
| 标题 | [简明扼要的标题] |
| 状态 | □建议 / □已接受 / □已拒绝 / ○已废弃 |
| 分类 | 技术选型 / 架构风格 / 数据架构 / 安全架构 / 集成架构 |
| 优先级 | P0(关键) / P1(重要) / P2(一般) |
| 提出日期 | YYYY-MM-DD |
| 决策日期 | YYYY-MM-DD |
| 复审日期 | YYYY-MM-DD |
| 提出人 | [姓名] |
| 决策人 | [姓名/架构委员会] |

## 2. 业务背景
### 2.1 问题陈述
[清晰描述需要解决的问题]

### 2.2 业务驱动因素
| 因素 | 优先级 | 说明 |
|------|--------|------|
| 监管合规 | 高 | |
| 成本控制 | 中 | |
| 上市时间 | 高 | |
| 技术债务 | 中 | |

### 2.3 成功标准
```

```
- [ ] 标准1
- [ ] 标准2

## 3. 约束条件
### 3.1 技术约束
-

### 3.2 业务约束
-

### 3.3 监管约束
-

### 3.4 资源约束
-

## 4. 决策内容
### 4.1 决策陈述
[使用命令式语句]

### 4.2 决策范围
- 适用：[范围]
- 不适用：[范围]

### 4.3 实施路线图
阶段1（YYYY-MM）：
阶段2（YYYY-MM）：
阶段3（YYYY-MM）：

## 5. 方案对比分析

| 评估维度 | 权重 | 方案A | 方案B | 方案C |
|---------|------|-------|-------|-------|
| 功能性 | 20% | 8/10 | 9/10 | 7/10 |
| 性能 | 20% | 7/10 | 8/10 | 9/10 |
| 安全性 | 20% | 9/10 | 8/10 | 8/10 |
| 可维护性 | 15% | 6/10 | 8/10 | 7/10 |
| 成本 | 15% | 8/10 | 6/10 | 9/10 |
| 上市时间 | 10% | 7/10 | 9/10 | 6/10 |
| **加权总分** | 100% | **7.5** | **8.1** | **7.6** |

## 6. 风险评估

| 风险 | 概率 | 影响 | 缓解措施 | 责任人 |
|------|------|------|---------|--------|
```

```
| | 高/中/低 | 高/中/低 | | |

## 7. 合规检查

| 合规要求 | 适用 | 满足方式 | 验证方法 |
|---------|------|---------|---------|
| 等保三级 | 是 | | |
| 数据安全法 | 是 | | |
| 个人隐私保护 | 是 | | |
| 行业监管规定 | 是 | | |

## 8. 影响分析

### 8.1 系统影响
- 影响系统：[列表]
- 改造工作量：[人天]

### 8.2 团队影响
- 技能要求：[新技能]
- 培训计划：[计划]

### 8.3 运营影响
- 监控需求：
- 变更流程：

## 9. 相关决策

| 关系 | ADR编号 | 说明 |
|------|---------|------|
| 依赖 | ADR-xxx | |
| 被依赖 | ADR-yyy | |
| 替代 | ADR-zzz | |

## 10. 审批记录

| 角色 | 姓名 | 日期 | 意见 | 签名 |
|------|------|------|------|------|
| 首席架构师 | | | | |
| 安全负责人 | | | | |
| 运维负责人 | | | | |
| 合规负责人 | | | | |

## 11. 附录

### 11.1 参考资料
```

```
-

### 11.2 术语表
-

### 11.3 修订历史
| 版本 | 日期 | 修改人 | 修改内容 |
|------|------|--------|----------|
| 1.0 | | | 初始版本 |
```

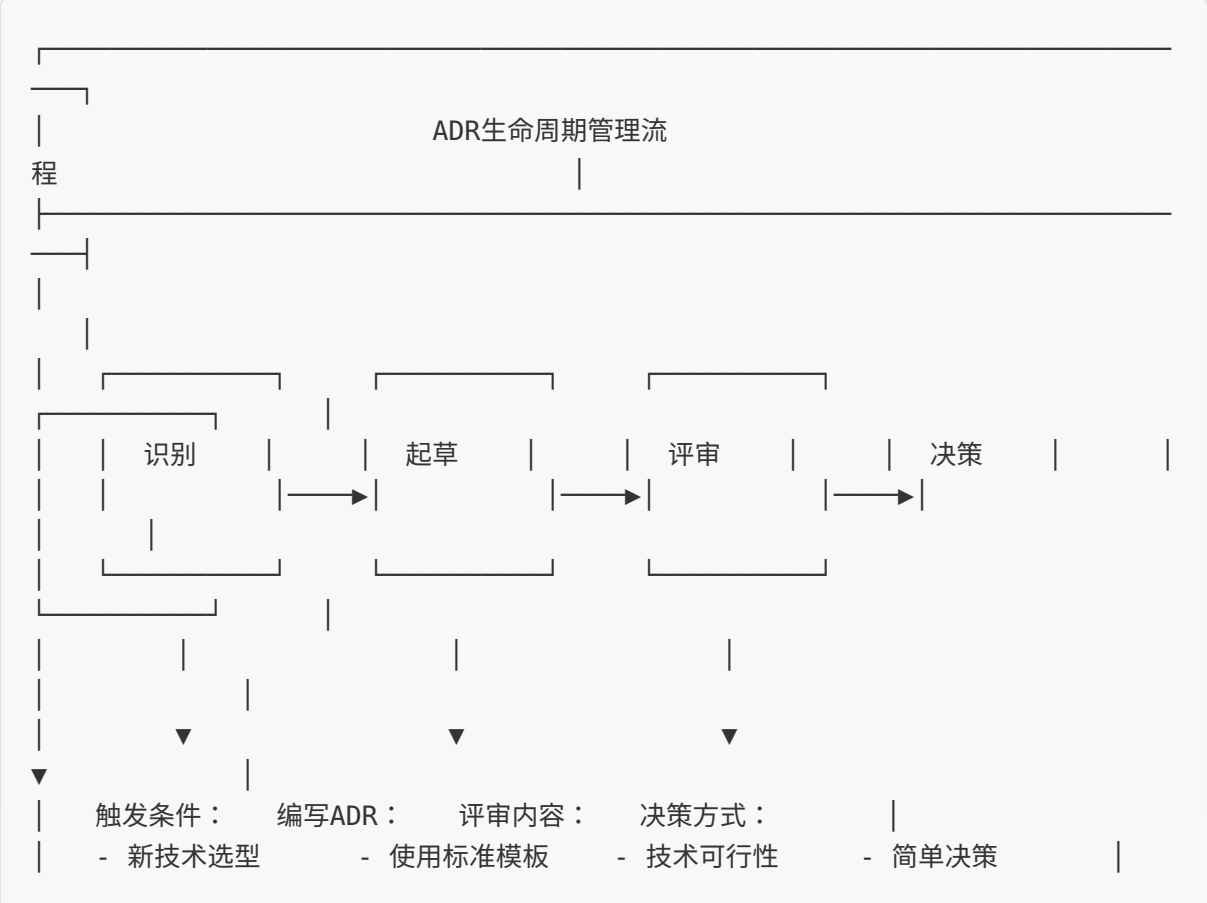
3.3 ADR示例：核心系统数据库选型

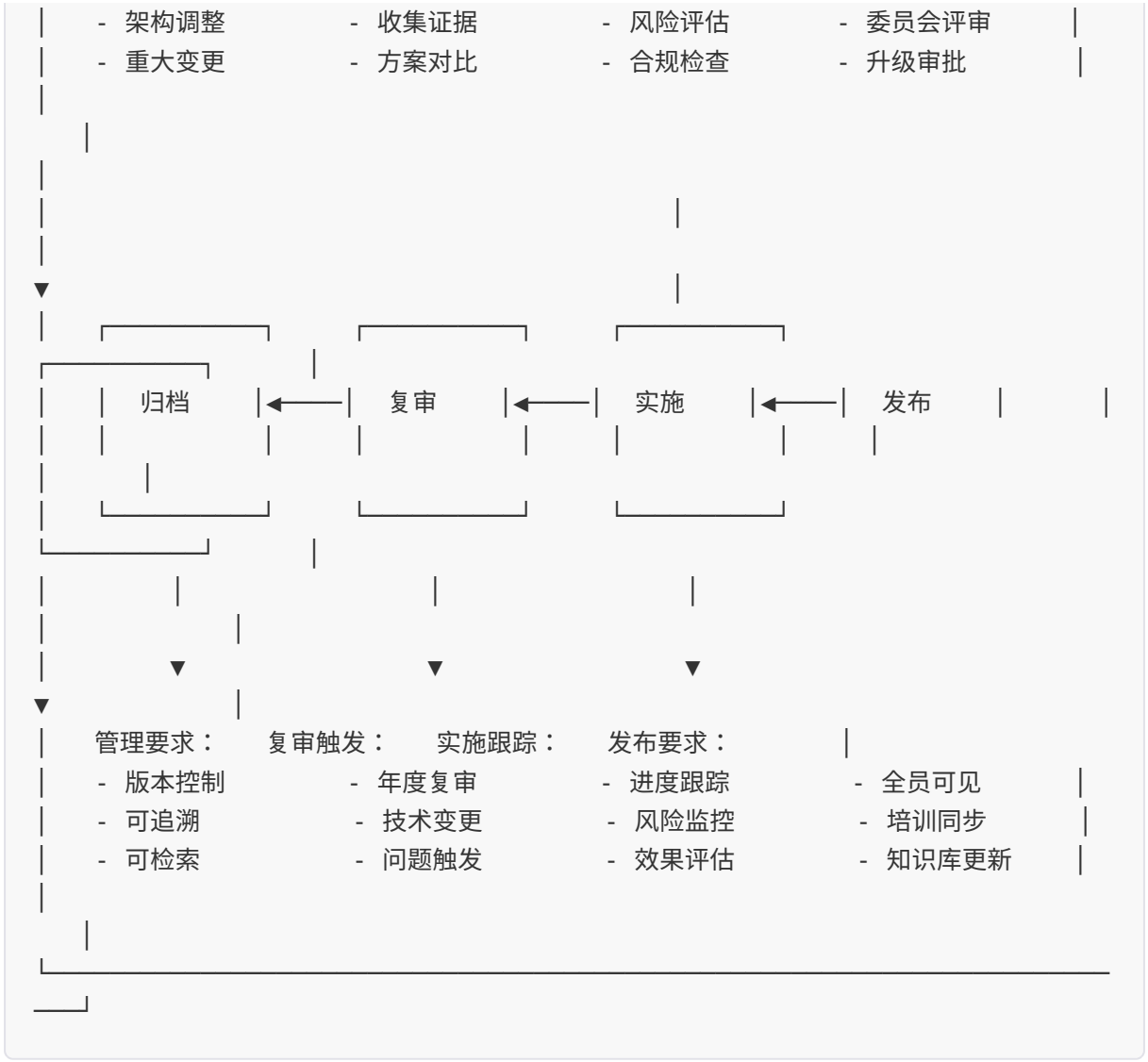
```
# ADR-042: 核心账务系统数据库选型决策

## 1. 元信息
| 属性 | 内容 |
|------|------|
| 编号 | ADR-042 |
| 标题 | 核心账务系统数据库选型决策 |
| 状态 | □已接受 |
| 分类 | 技术选型 > 数据存储 |
| 优先级 | P0(关键) |
| 提出日期 | 2024-01-15 |
| 决策日期 | 2024-02-20 |
| 复审日期 | 2025-02-20 |
| 提出人 | 张三（应用架构师） |
| 决策人 | 架构委员会 |

## 2. 业务背景
### 2.1 问题陈述
当前核心账务系统基于Oracle数据库，面临以下挑战：
1. license成本持续上升，年费用超过800万
2. 集中式架构无法满足业务扩展需求
3. 自主可控要求下需要国产替代方案

### 2.2 业务驱动因素
| 因素 | 优先级 | 说明 |
|------|--------|------|
| 成本控制 | 高 | 未来3年降低数据库成本50% |
| 自主可控 | 高 | 符合金融监管国产化要求 |
| 性能扩展 | 高 | 支持日均交易量增长300% |
| 技术演进 | 中 | 向云原生架构演进 |
```

```
## 3. 约束条件
### 3.1 技术约束
- 必须支持ACID事务
- 必须支持强一致性
- 必须支持两地三中心部署
- RTO < 5分钟，RPO = 0

### 3.2 监管约束
- 符合等保三级要求
- 符合金融业信息系统国产化要求
- 支持审计追踪

## 4. 决策内容
### 4.1 决策陈述
我们将采用分布式关系型数据库TiDB作为核心账务系统主数据库，Oracle作为历史查询备库。

### 4.2 实施路线图
阶段1（2024Q2）：完成POC验证，确认技术可行性
阶段2（2024Q3）：完成非核心模块迁移
阶段3（2024Q4）：核心账务模块迁移
阶段4（2025Q1）：完成双轨运行，Oracle转备库

## 5. 方案对比分析

| 评估维度 | 权重 | Oracle RAC | TiDB | OceanBase | GoldenDB |
|---------|------|------------|------|-----------|----------|
| 事务一致性 | 25% | 10/10 | 9/10 | 9/10 | 9/10 |
| 性能扩展 | 20% | 6/10 | 9/10 | 9/10 | 8/10 |
| 成本效益 | 20% | 4/10 | 9/10 | 8/10 | 7/10 |
| 运维复杂度 | 15% | 7/10 | 7/10 | 7/10 | 6/10 |
| 生态成熟度 | 10% | 10/10 | 8/10 | 7/10 | 7/10 |
| 国产化要求 | 10% | 0/10 | 10/10 | 10/10 | 10/10 |
| **加权总分** | 100% | **6.1** | **8.6** | **8.2** | **7.7** |

## 6. 风险评估

| 风险 | 概率 | 影响 | 缓解措施 | 责任人 |
|------|------|------|---------|--------|
| 分布式事务性能不达预期 | 中 | 高 | 充分POC测试；预留回退方案 | 李四 |
| 团队技能不足 | 中 | 中 | 提前培训；引入外部专家 | 王五 |
| 数据迁移失败 | 低 | 高 | 双轨并行；逐步切流 | 赵六 |

## 7. 合规检查

| 合规要求 | 适用 | 满足方式 | 验证方法 |
```

```
|---------|------|---------|---------|
| 等保三级 | 是 | TiDB支持国密算法、审计日志 | 第三方测评 |
| 数据安全法 | 是 | 数据分类分级、加密存储 | 合规检查 |
| 金融业国产化 | 是 | 国产数据库 | 资质审核 |

## 8. 审批记录

| 角色 | 姓名 | 日期 | 意见 | 签名 |
|------|------|------|------|------|
| 首席架构师 | 陈总 | 2024-02-18 | 同意 | ✓ |
| 安全负责人 | 刘总 | 2024-02-19 | 同意，建议加强审计 | ✓ |
| 运维负责人 | 周总 | 2024-02-19 | 同意 | ✓ |

## 9. 修订历史
| 版本 | 日期 | 修改人 | 修改内容 |
|------|------|--------|----------|
| 1.0 | 2024-01-15 | 张三 | 初始版本 |
| 1.1 | 2024-02-10 | 张三 | 补充POC结果 |
| 1.2 | 2024-02-20 | 张三 | 更新审批状态 |
```

3.4 ADR管理流程

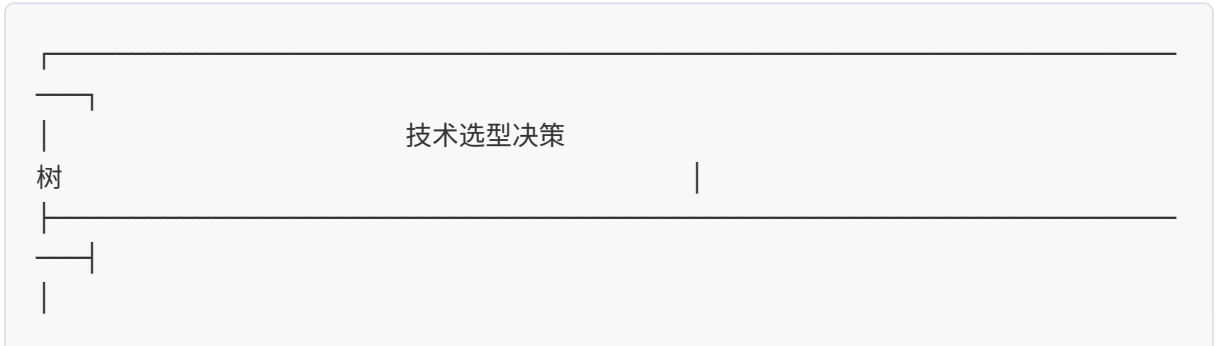


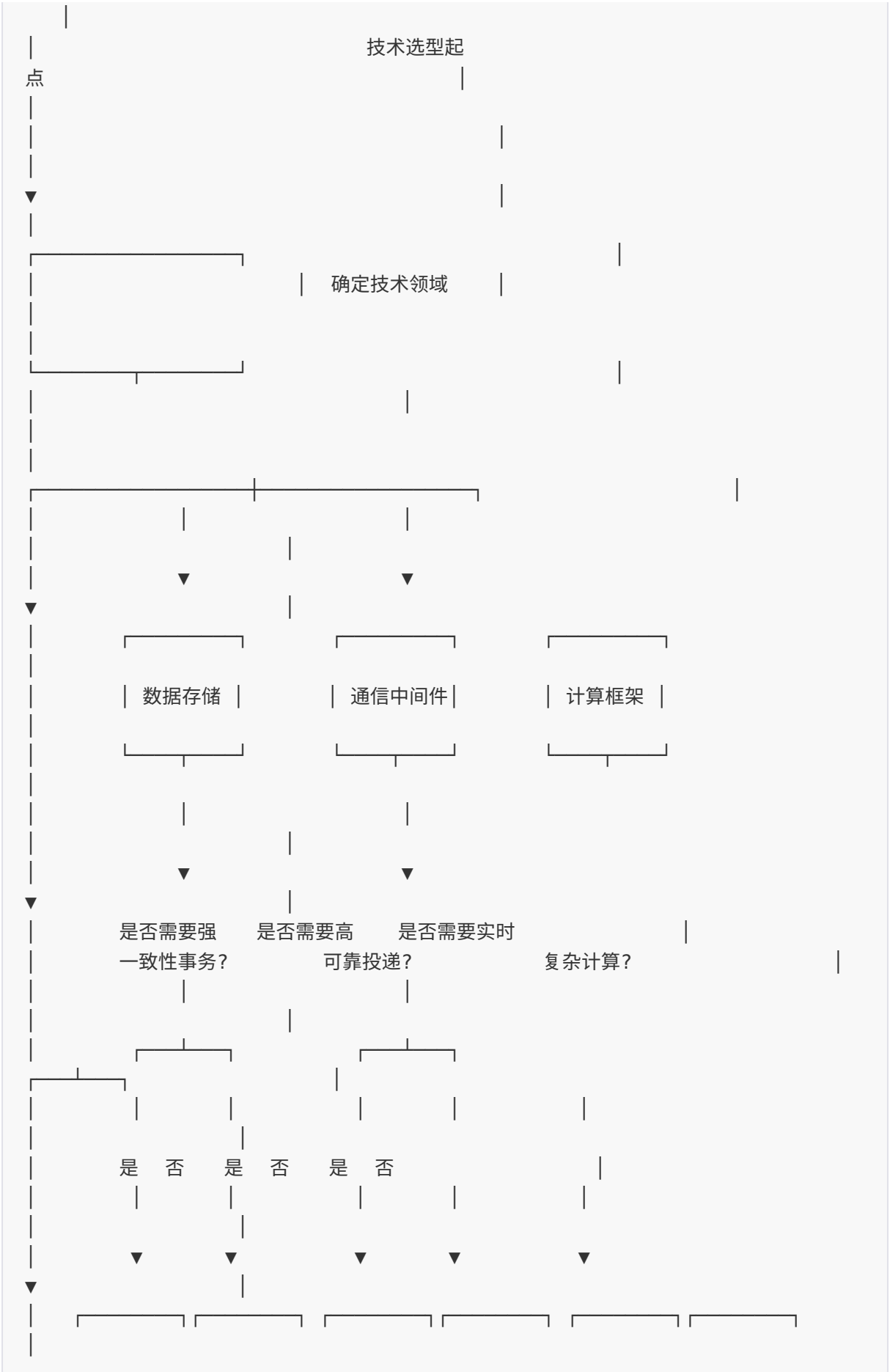


第4章 技术选型决策框架

4.1 技术选型决策树

金融系统技术选型通用决策树

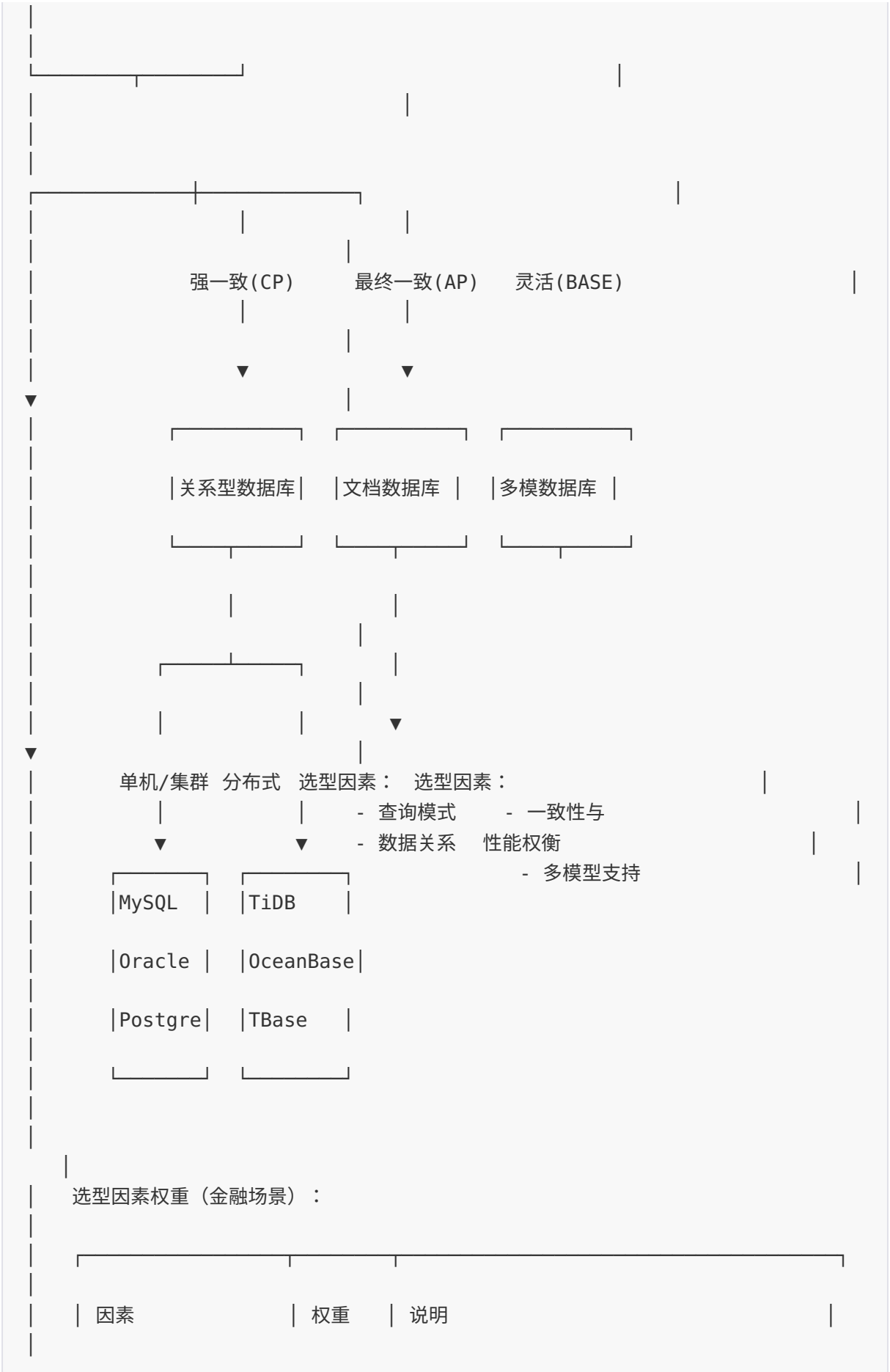


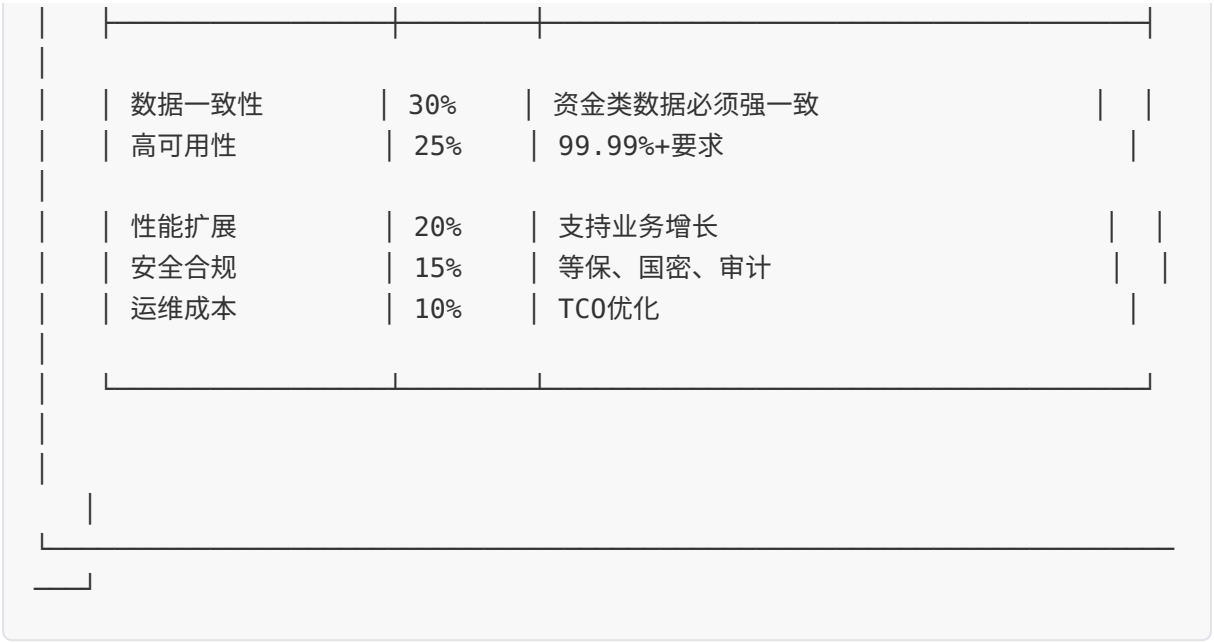

技术选型起点
确定技术领域
数据存储
通信中间件
计算框架
是否需要强一致性事务?
是否需要高可靠投递?
是否需要实时复杂计算?
是 否 是 否 是 否

```
|    |关系型DB||NoSQL    | |消息队列||HTTP/RPC| |流计算   ||批处理   |       |
|    |- TiDB  ||- Redis | |- Kafka ||- gRPC  | |- Flink ||- Spark |
|
|    |- OB    ||- Mongo | |- Rocket||- REST  | |- Storm ||- Hadoop|
|
|    └─────────┘└─────────┘ └─────────┘└─────────┘ └─────────┘└─────────┘
|
|
    |
|    选型检查清
单：                                                                  |
|    □ 是否满足功能性需求
|
|    □ 是否满足非功能性需求（性能、可用性、安全）                                  |
|    □ 是否符合技术约束（已有技术栈、团队技能）                                    |
|    □ 是否满足合规要求（等保、国产化）
|
|    □ 成本是否在预算范围
内                                                  |
|    □ 是否有足够的社区/厂商支持
|
|
    |
└──────────────────────────────────────────────────────────────────────────
──┘
```

数据库选型决策树（详细版）

```
┌──────────────────────────────────────────────────────────────────────────
──┐
|                          数据库选型决策流
程                                    |
├──────────────────────────────────────────────────────────────────────────
──┤
|
    |
|                              数据库选型决
策                                    |
|
|                                        |
|
▼                                        |
|
┌──────────────────┐                                |
|                       |  数据一致性要求   |
```


强一致(CP)
最终一致(AP)
灵活(BASE)
关系型数据库
文档数据库
多模数据库
单机/集群
分布式
选型因素：
- 查询模式
- 数据关系
选型因素：
- 一致性与
性能权衡
- 多模型支持
MySQL
TiDB
Oracle
OceanBase
Postgre
TBase
选型因素权重（金融场景）：
因素
权重
说明



4.2 技术评估矩阵模板

综合技术评估矩阵

评估维度	权重	评分标准	候选技术A	候选技术B	候选技术C
功能性	15%				
- 功能完备性		是否满足全部功能需求			
- 扩展性		是否支持未来功能扩展			
非功能性	35%				
- 性能(吞吐/延迟)		TPS/QPS、响应时间			
- 可用性		SLA、故障恢复时间			
- 可扩展性		水平/垂直扩展能力			
- 安全性		认证、加密、审计			

评估维度	权重	评分标准	候选技术A	候选技术B	候选技术C
运维性	20%				
- 可监控性		监控指标丰富度			
- 可维护性		运维复杂度、文档			
- 可移植性		云原生支持、跨平台			
生态与社区	15%				
- 社区活跃度		GitHub stars、贡献者			
- 企业支持		商业支持、案例			
- 集成生态		与现有系统集成度			
成本	15%				
- 许可成本		License费用			
- 运维成本		人力、硬件			
- 迁移成本		改造成本、风险			
加权总分	100%				

金融特性专项评估矩阵

评估项	重要性	评估内容	合规要求	验证方法
数据安全	关键	加密算法支持（国密/国际）、密钥管理	等保3级、密评	安全测试
审计日志	关键	操作日志完整性、不可篡改、保留期限	金融监管要求	日志审计
高可用	关键	RTO/RPO指标、灾备架构	业务连续性管理	灾备演练

评估项	重要性	评估内容	合规要求	验证方法
数据一致性	关键	事务支持级别、分布式一致性	资金安全	一致性测试
监管报送	重要	数据格式、接口标准	央行/银保监会	联调测试
国产化	重要	自主可控、供应链安全	金融信创要求	资质审核

4.3 技术选型案例：实时风控引擎技术栈

业务需求背景

- 需要支持每秒10万+事件处理
- 规则响应延迟<50ms（P99）
- 支持复杂规则（CEP）和机器学习模型
- 7×24小时不间断服务
- 支持规则热更新

候选方案对比

维度	权重	Apache Flink + Drools	Spark Streaming + 自研	商业方案（SAS）
处理延迟	25%	9/10（毫秒级）	7/10（秒级）	8/10（毫秒级）
吞吐能力	20%	9/10（百万级）	8/10（十万级）	7/10（十万级）
规则复杂度	20%	8/10（支持CEP）	7/10	9/10（完善）
机器学习	15%	8/10（集成良好）	9/10	9/10（完善）
运维成本	10%	7/10（需专业团队）	6/10	9/10（厂商支持）
采购成本	10%	9/10（开源）	10/10（自研）	4/10（昂贵）
加权总分	100%	**8.4**	**7.5**	**7.7**

决策结论

选择Apache Flink + Drools方案，理由：

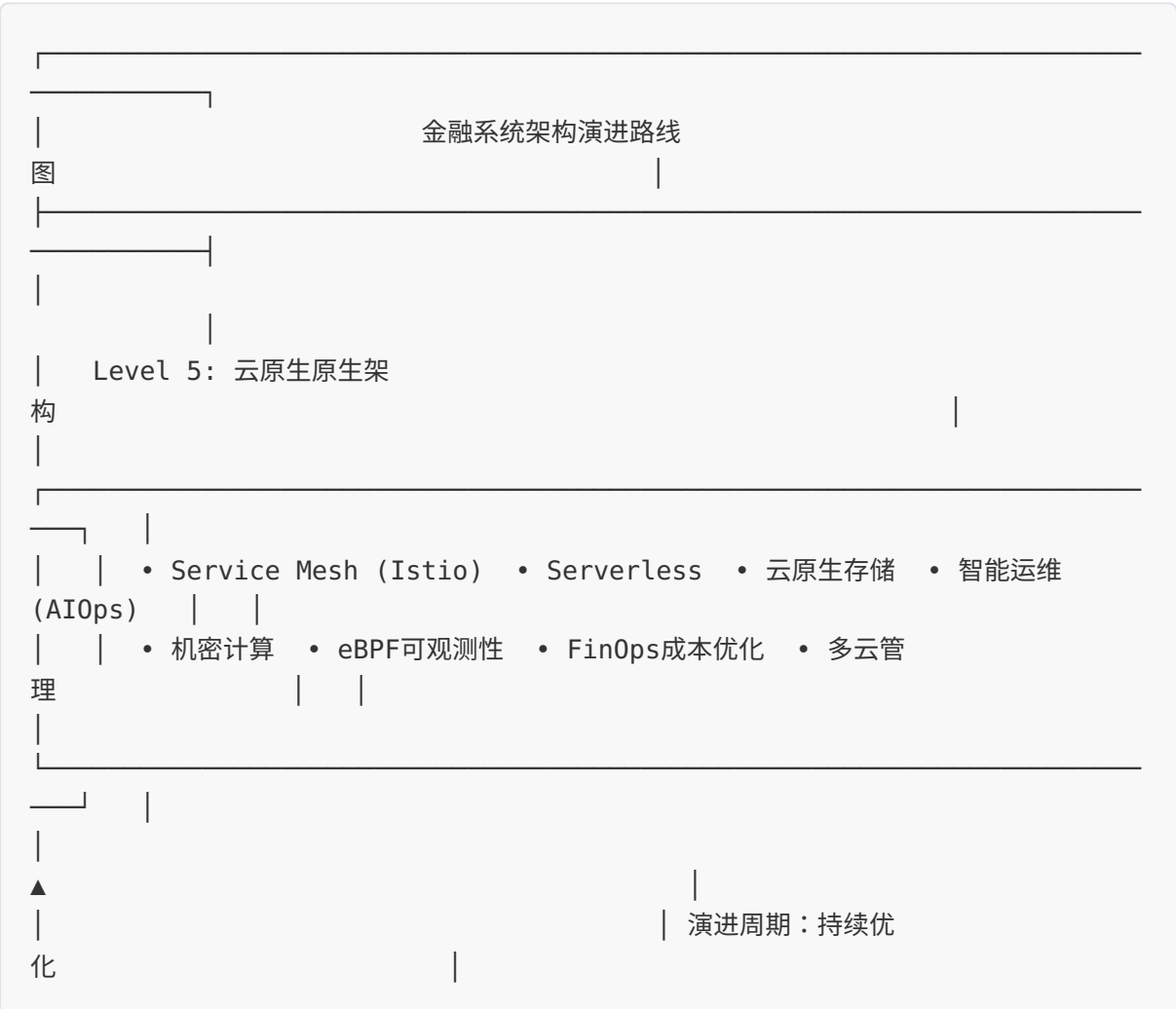
1. 满足严格的延迟和吞吐要求
2. 开源可控，长期TCO更低
3. 团队已有Flink使用经验
4. 社区活跃，生态完善

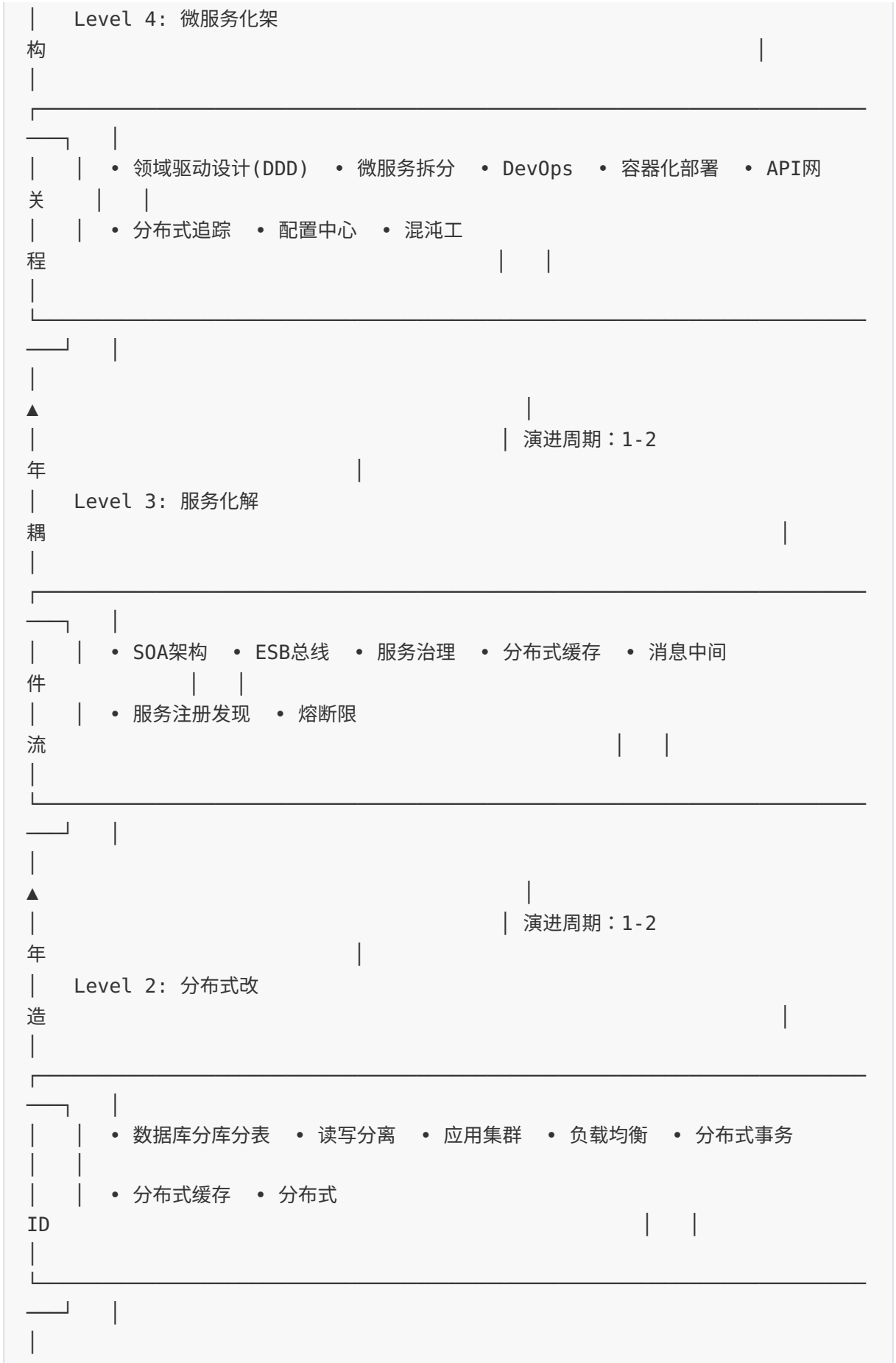
风险缓解措施

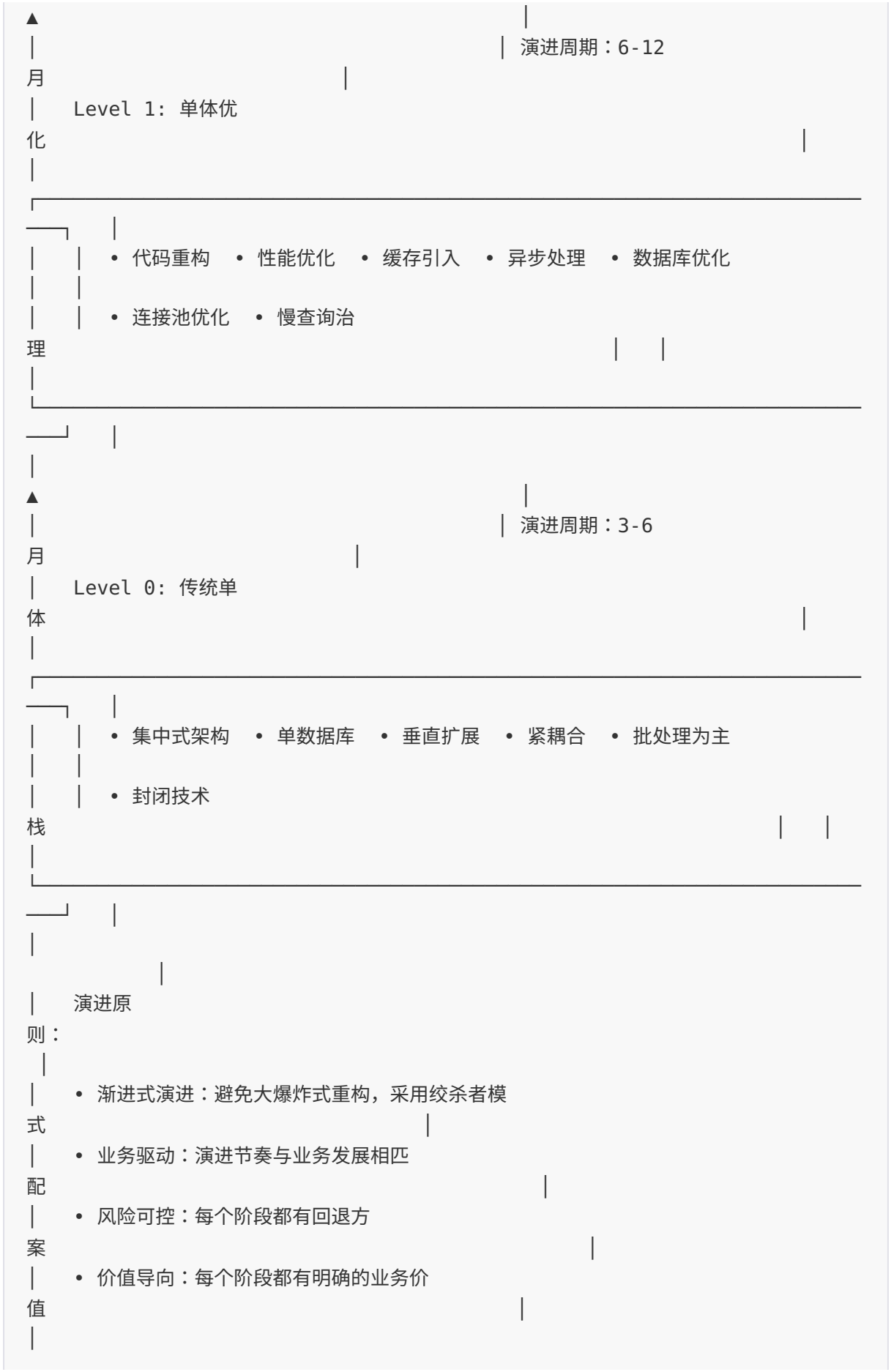
- 与Flink商业公司Ververica建立技术支持关系
- 关键人员培训计划
- 降级方案：高峰期可切换至简化规则集

第5章 架构演进路线图

5.1 从单体到云原生的演进路径




Level 4: 微服务化架构
• 领域驱动设计(DDD) • 微服务拆分 • DevOps • 容器化部署 • API网关
• 分布式追踪 • 配置中心 • 混沌工程
演进周期：1-2年
Level 3: 服务化解耦
• SOA架构 • ESB总线 • 服务治理 • 分布式缓存 • 消息中间件
• 服务注册发现 • 熔断限流
演进周期：1-2年
Level 2: 分布式改造
• 数据库分库分表 • 读写分离 • 应用集群 • 负载均衡 • 分布式事务
• 分布式缓存 • 分布式ID

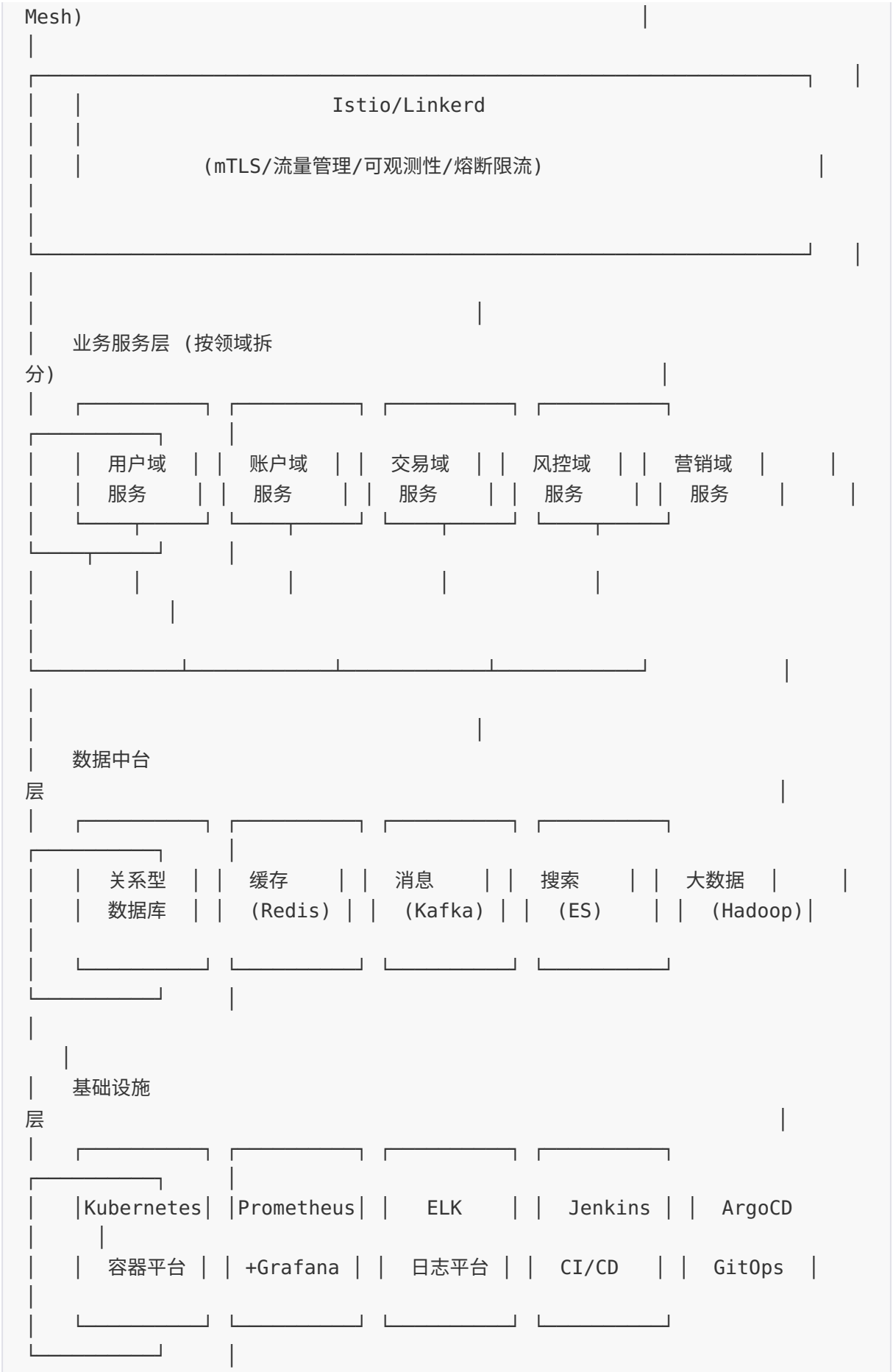

演进周期：6-12月
Level 1: 单体优化
• 代码重构 • 性能优化 • 缓存引入 • 异步处理 • 数据库优化
• 连接池优化 • 慢查询治理
演进周期：3-6月
Level 0: 传统单体
• 集中式架构 • 单数据库 • 垂直扩展 • 紧耦合 • 批处理为主
• 封闭技术栈
演进原则：
• 渐进式演进：避免大爆炸式重构，采用绞杀者模式
• 业务驱动：演进节奏与业务发展相匹配
• 风险可控：每个阶段都有回退方案
• 价值导向：每个阶段都有明确的业务价值

5.2 各阶段详细设计

Level 1-2: 单体优化与分布式改造

优化领域	具体措施	预期效果	实施难度	关键指标
代码层面	模块化重构、消除循环依赖	可维护性↑30%	中	代码复杂度
数据库	索引优化、SQL调优、慢查询治理	查询性能↑50%	低	响应时间
缓存	引入Redis缓存热点数据	响应时间↓60%	低	缓存命中率
异步	非关键路径异步化（MQ）	吞吐↑40%	中	队列积压
分库分表	水平/垂直拆分	容量扩展10倍	高	分片均衡度

Level 3-4: 服务化与微服务化


Mesh)
Istio/Linkerd
(mTLS/流量管理/可观测性/熔断限流)
业务服务层 (按领域拆
分)
用户域 服务
账户域 服务
交易域 服务
风控域 服务
营销域 服务
数据中台
层
关系型 数据库
缓存 (Redis)
消息 (Kafka)
搜索 (ES)
大数据 (Hadoop)
基础设施
层
Kubernetes
Prometheus +Grafana
ELK
Jenkins
ArgoCD
容器平台
日志平台
CI/CD
GitOps

5.3 演进策略与风险控制

演进策略矩阵

策略	适用场景	优点	风险	案例
绞杀者模式	遗留系统复杂、无法直接替换	风险可控、渐进式迁移	过渡期维护成本高	核心系统改造
大爆炸模式	系统相对简单、有停机窗口	一次性完成、无过渡期	风险集中、回滚困难	外围系统升级
并行运行	关键系统、零容忍故障	随时回退、风险最低	资源消耗大、数据一致性挑战	支付系统升级
微前端模式	前端系统现代化	渐进式用户体验升级	集成复杂度	网银系统重构
数据同步模式	数据层改造	数据一致性好	同步延迟	数仓建设

第6章 量化架构设计

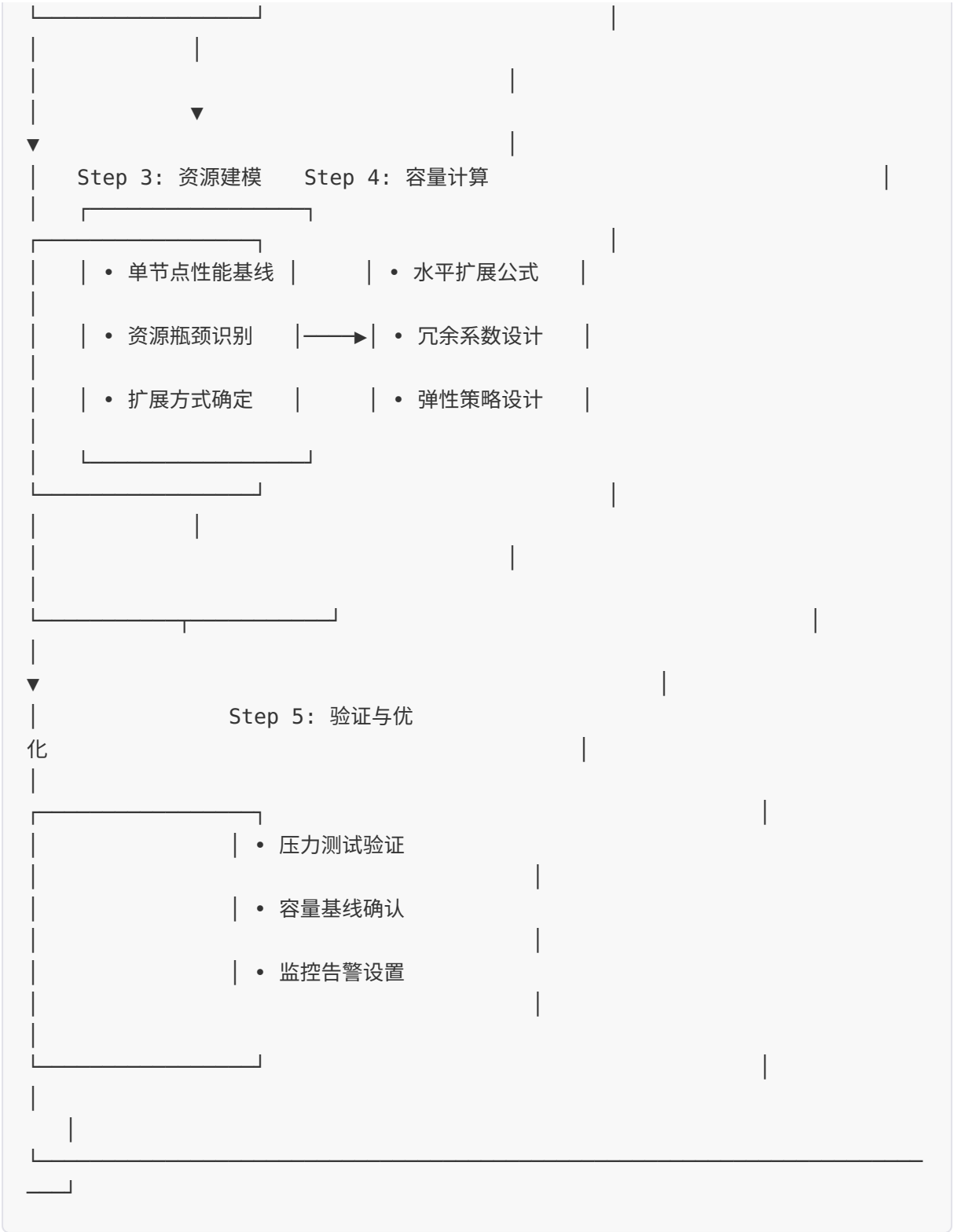
6.1 性能架构量化设计

性能指标体系

指标类型	指标名称	定义	金融行业标准	测量方法
吞吐量	TPS	每秒事务处理数	支付: 10000+	压测工具
	QPS	每秒查询处理数	查询: 50000+	压测工具

指标类型	指标名称	定义	金融行业标准	测量方法
延迟	平均响应时间	请求平均耗时	< 200ms	APM工具
	P95延迟	95分位响应时间	< 500ms	APM工具
	P99延迟	99分位响应时间	< 1000ms	APM工具
	P999延迟	99.9分位响应时间	< 2000ms	APM工具
并发	并发用户数	同时在线用户数	设计容量的200%	压力测试
	并发连接数	保持的连接数	设计容量的150%	监控指标
资源	CPU利用率	CPU使用百分比	< 70%（常态）	系统监控
	内存利用率	内存使用百分比	< 80%	系统监控
	磁盘I/O	磁盘读写速率	< 70%带宽	系统监控
网络	带宽利用率	网络带宽使用	< 60%	网络监控
	网络延迟	网络传输延迟	< 5ms（内网）	网络探测

性能容量规划模型



容量计算公式

```
所需服务器数量 = 峰值TPS / (单服务器TPS × 冗余系数)

其中：
- 单服务器TPS = 基准TPS × 目标CPU利用率
```

```
- 冗余系数 = 1.3 ~ 1.5 (建议值)

示例：
- 目标峰值TPS: 100,000
- 单服务器基准TPS: 5,000
- 目标CPU利用率: 70%
- 冗余系数: 1.3

计算：
单服务器实际TPS = 5000 × 0.7 = 3,500
所需服务器 = 100,000 / (3,500 × 1.3) ≈ 22台
```

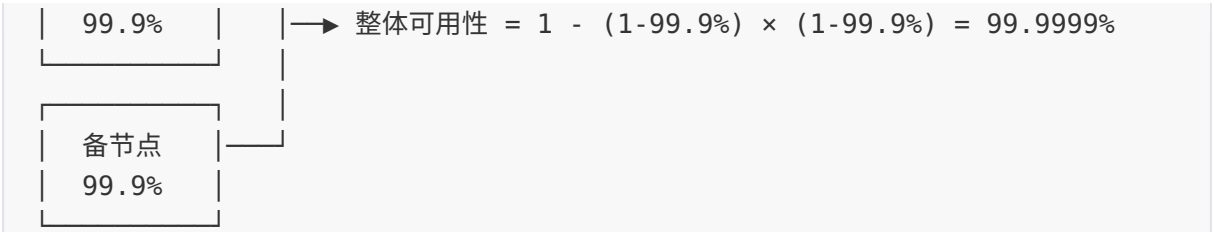
6.2 可用性架构量化设计

可用性等级定义

等级	可用性	年停机时间	适用场景	架构要求	金融示例
可用1级	99.9%	8.76小时	内部管理系统	单节点+备份	OA系统
可用2级	99.95%	4.38小时	一般业务系统	主备架构	培训系统
可用3级	99.99%	52.6分钟	重要业务系统	主备+自动切换	营销系统
可用4级	99.999%	5.26分钟	核心交易系统	同城双活	网银系统
可用5级	99.9999%	31.5秒	支付清算系统	异地多活	支付系统

可用性计算模型

```
系统整体可用性 = 各组件可用性的乘积

示例（串联组件）：
┌──────────┐    ┌──────────┐    ┌──────────┐
│ 负载均衡 │──→│ 应用服务 │──→│ 数据库   │
│ 99.99%   │    │ 99.95%   │    │ 99.99%   │
└──────────┘    └──────────┘    └──────────┘

系统可用性 = 99.99% × 99.95% × 99.99% ≈ 99.93%

并联组件可用性计算：
┌──────────┐
│ 主节点   │──┐
```



RTO/RPO设计矩阵

系统等级	RTO要求	RPO要求	实现架构	技术方案
核心业务	< 5分钟	0	同城双活+异地灾备	数据库同步复制、应用双活
重要业务	< 30分钟	< 5分钟	主备+异地备份	异步复制、快照备份
一般业务	< 4小时	< 1小时	主备	定时备份、日志复制
内部系统	< 24小时	< 24小时	冷备	定期全量备份

第7章 架构评审与治理

7.1 架构评审检查清单

通用架构评审检查清单

检查项	检查内容	检查标准	结果	备注
功能性				
FUNC-01	功能覆盖度	100%覆盖需求	□通过 □不通过	
FUNC-02	正确性保证	有验证机制	□通过 □不通过	
FUNC-03	互操作性	符合接口标准	□通过 □不通过	
性能效率				
PERF-01	响应时间	满足SLA要求	□通过 □不通过	
PERF-02	吞吐量	满足峰值需求	□通过 □不通过	

检查项	检查内容	检查标准	结果	备注
PERF-03	资源利用率	< 70%目标	□通过 □不通过	
兼容性				
COMP-01	向后兼容	兼容现有版本	□通过 □不通过	
COMP-02	集成兼容	符合企业标准	□通过 □不通过	
可用性				
AVAI-01	服务可用性	满足SLA要求	□通过 □不通过	
AVAI-02	故障恢复	RTO/RPO达标	□通过 □不通过	
AVAI-03	降级策略	有降级方案	□通过 □不通过	
可靠性				
RELI-01	容错设计	单点故障可隔离	□通过 □不通过	
RELI-02	数据一致性	符合业务要求	□通过 □不通过	
RELI-03	幂等性	关键操作幂等	□通过 □不通过	
安全性				
SECU-01	认证授权	统一身份认证	□通过 □不通过	
SECU-02	数据加密	传输和存储加密	□通过 □不通过	
SECU-03	审计日志	操作可追溯	□通过 □不通过	
SECU-04	输入校验	防注入/防篡改	□通过 □不通过	
可维护性				
MAINT-01	模块化	低耦合高内聚	□通过 □不通过	
MAINT-02	可测试性	单元测试覆盖>80%	□通过 □不通过	
MAINT-03	可观测性	日志/指标/追踪	□通过 □不通过	

检查项	检查内容	检查标准	结果	备注
可移植性				
PORT-01	云原生	支持容器化部署	□通过 □不通过	
PORT-02	配置外置	配置与代码分离	□通过 □不通过	

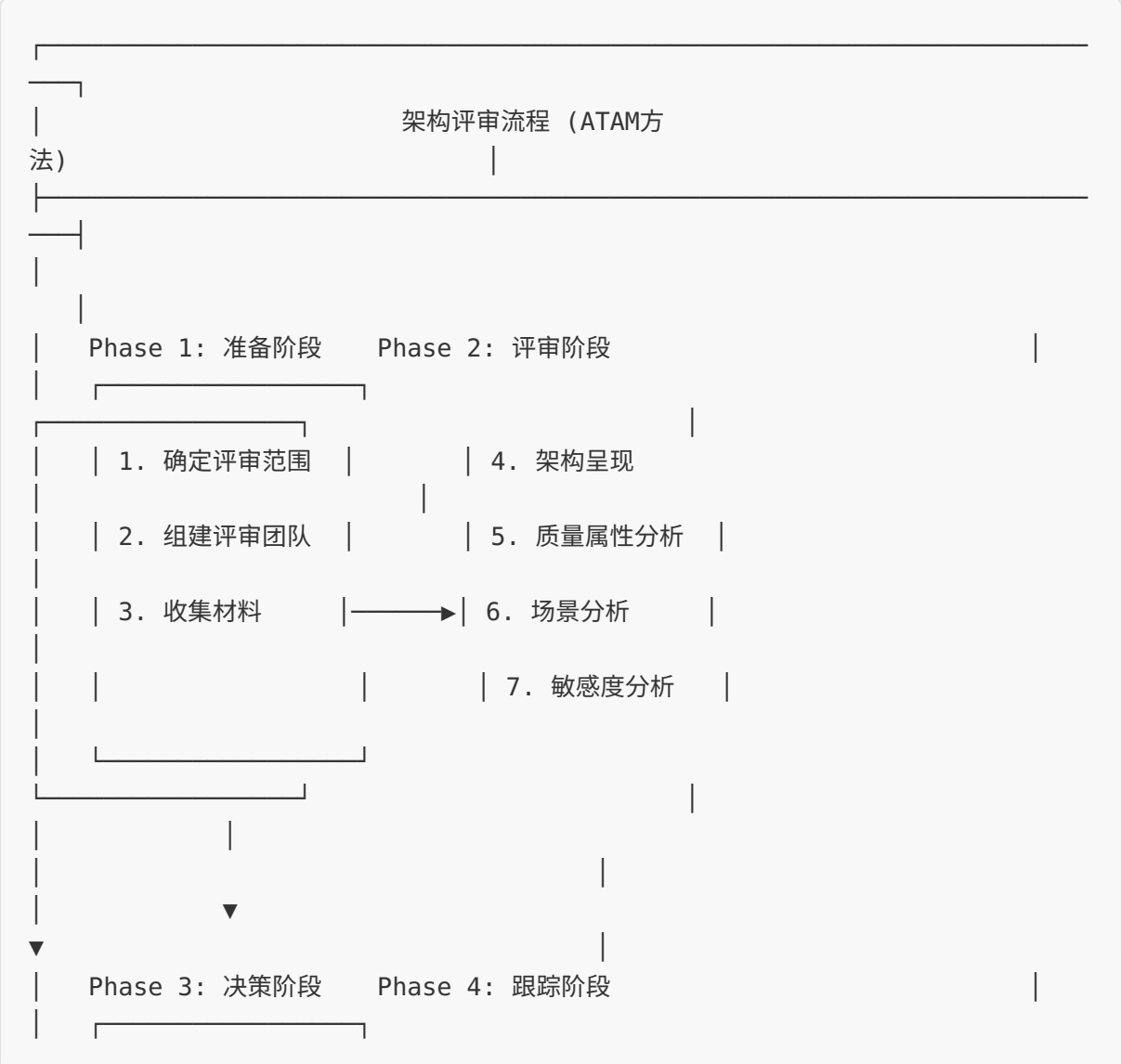
7.2 金融专项架构评审检查清单

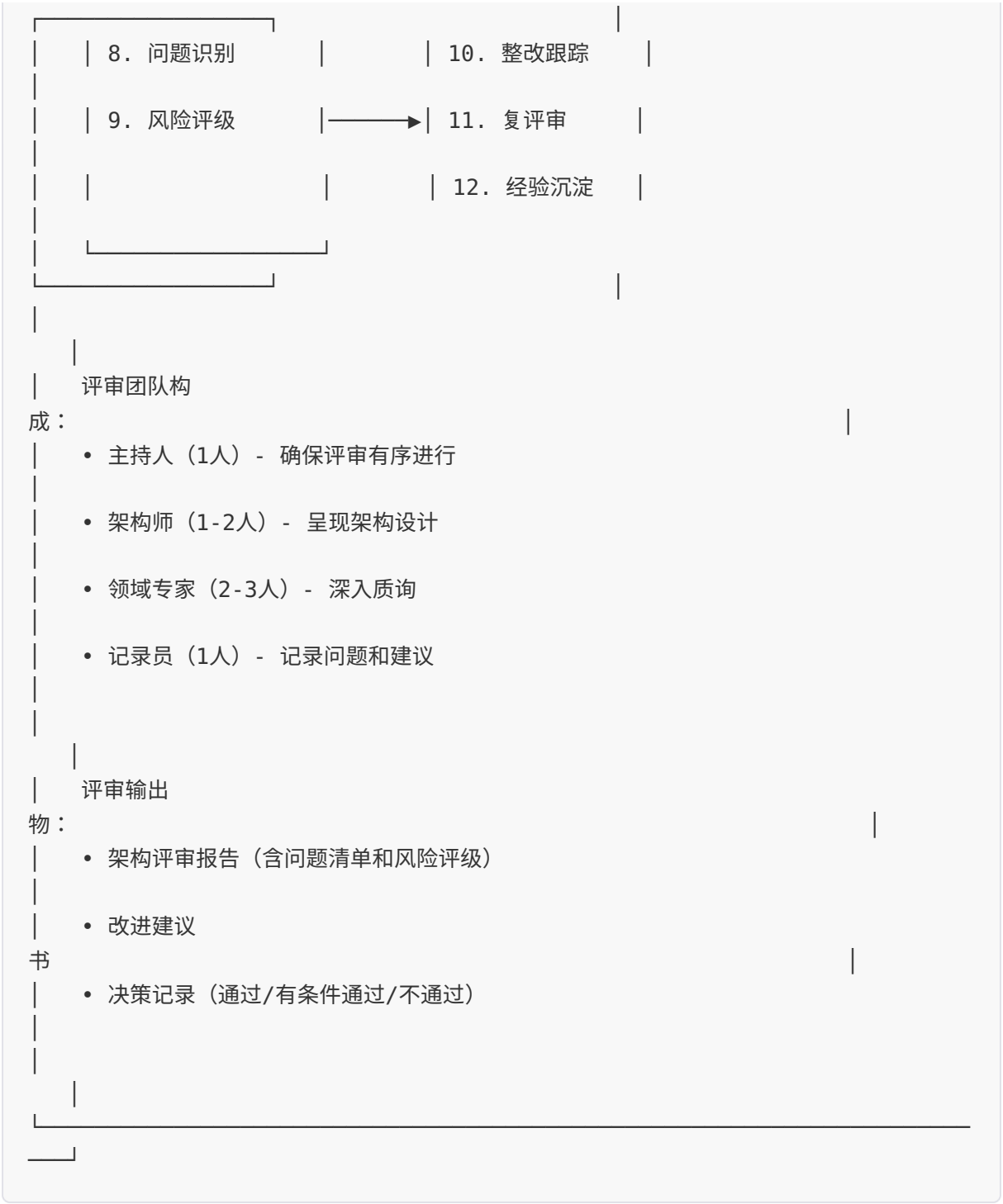
金融安全合规检查清单

检查项	检查内容	合规标准	验证方法	结果
等保合规				
CPL-001	等级保护定级	系统定级准确	定级报告	□是 □否
CPL-002	安全物理环境	机房符合要求	现场检查	□是 □否
CPL-003	安全通信网络	网络分区隔离	网络拓扑	□是 □否
CPL-004	安全区域边界	边界防护措施	配置检查	□是 □否
CPL-005	安全计算环境	主机安全加固	基线检查	□是 □否
CPL-006	安全管理中心	集中安全管理	平台验证	□是 □否
数据安全				
DSE-001	数据分类分级	数据分类准确	分类清单	□是 □否
DSE-002	数据加密	敏感数据加密	加密配置	□是 □否
DSE-003	数据脱敏	脱敏策略生效	脱敏验证	□是 □否
DSE-004	数据备份	备份策略有效	恢复演练	□是 □否
DSE-005	数据访问控制	最小权限原则	权限审计	□是 □否
业务连续性				
BCP-001	灾备架构	两地三中心	架构评审	□是 □否

检查项	检查内容	合规标准	验证方法	结果
BCP-002	RTO/RPO	满足业务要求	演练验证	□是 □否
BCP-003	应急预案	预案完备有效	预案评审	□是 □否
金融监管				
REG-001	交易记录	记录完整准确	日志审计	□是 □否
REG-002	数据报送	接口符合规范	联调测试	□是 □否
REG-003	风险监控	监控规则有效	规则验证	□是 □否

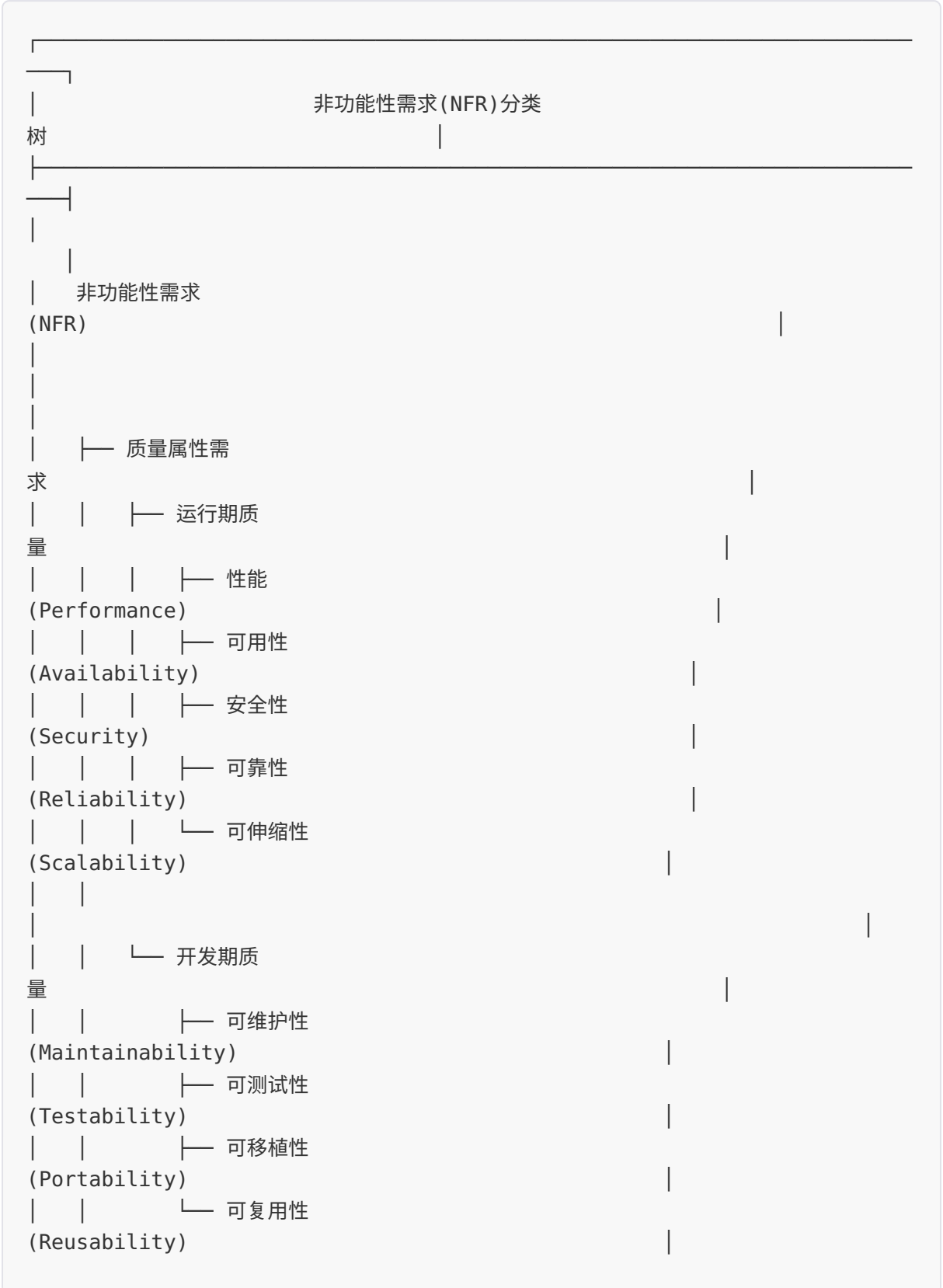
7.3 架构评审流程

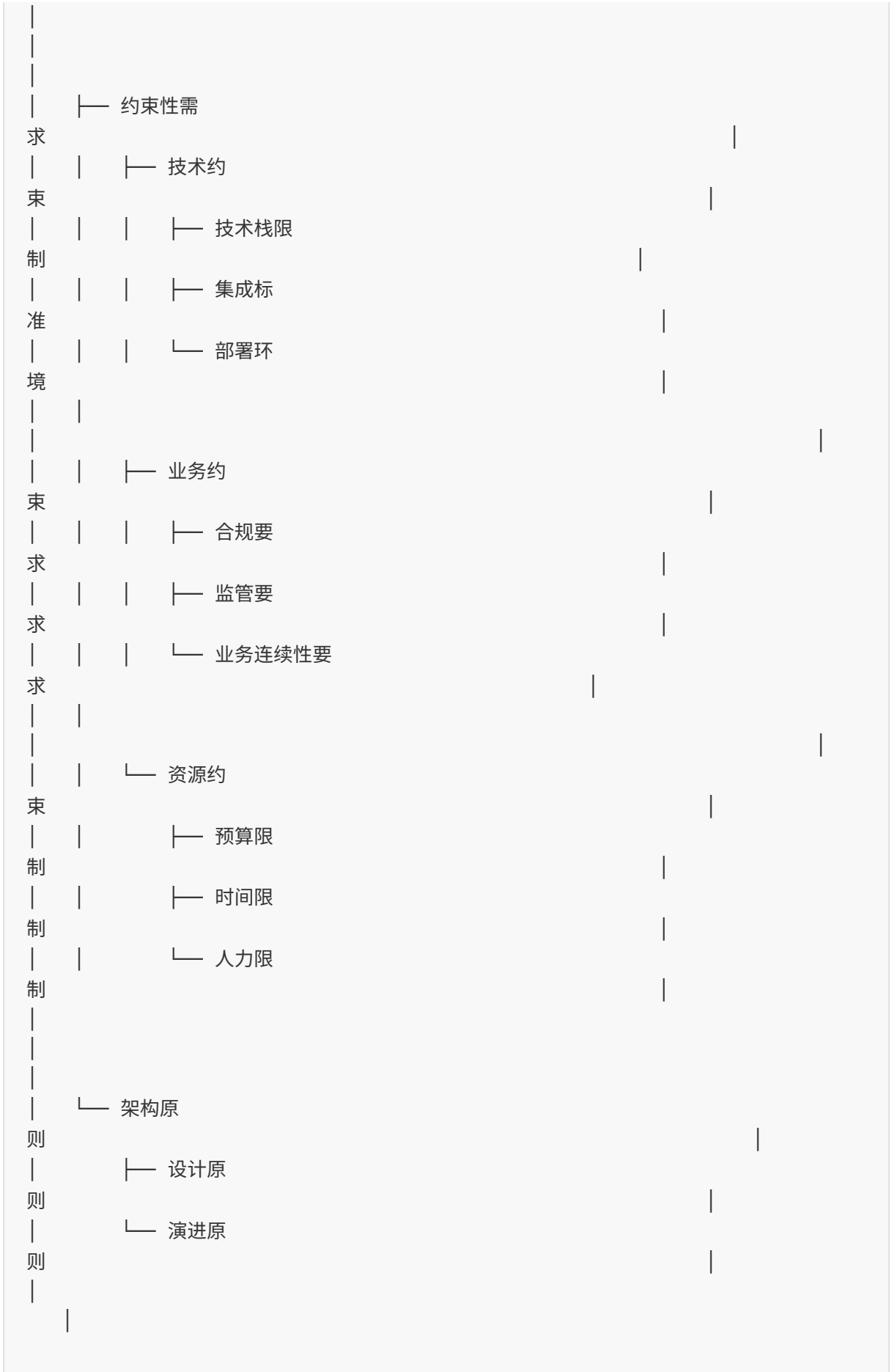



8. 问题识别
9. 风险评级
10. 整改跟踪
11. 复评审
12. 经验沉淀
评审团队构成：
• 主持人（1人）- 确保评审有序进行
• 架构师（1-2人）- 呈现架构设计
• 领域专家（2-3人）- 深入质询
• 记录员（1人）- 记录问题和建议
评审输出物：
• 架构评审报告（含问题清单和风险评级）
• 改进建议书
• 决策记录（通过/有条件通过/不通过）

第8章 非功能性需求框架

8.1 NFR分类体系

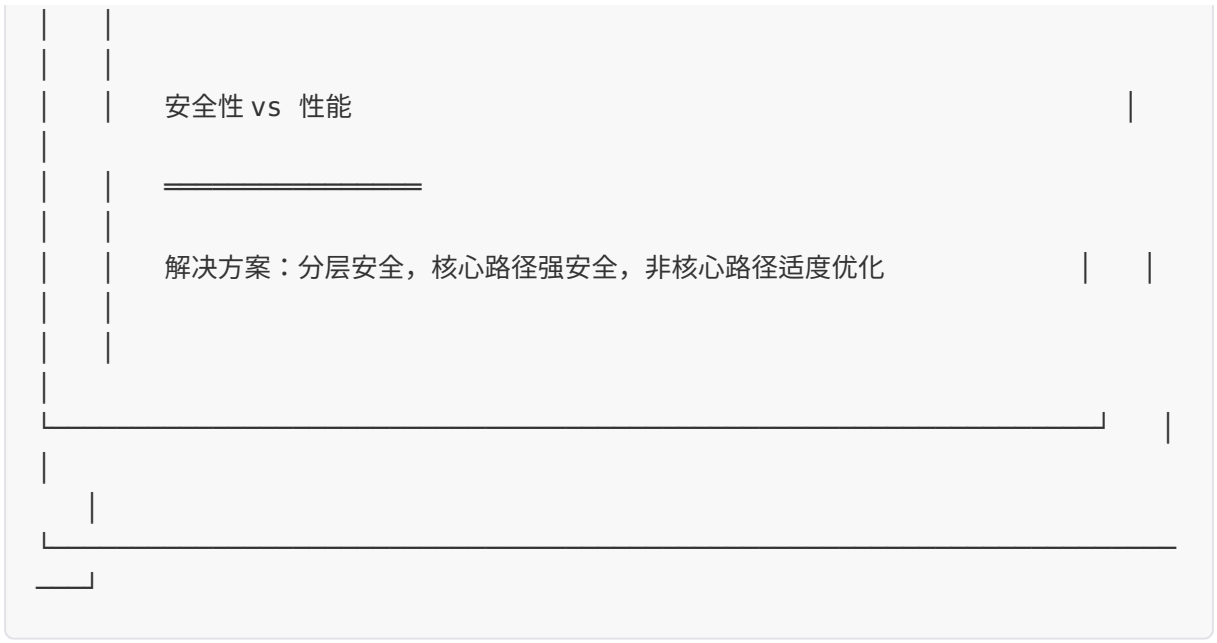



├── 约束性需
求
├── 技术约
束
├── 技术栈限
制
├── 集成标
准
└── 部署环
境
├── 业务约
束
├── 合规要
求
├── 监管要
求
└── 业务连续性要
求
└── 资源约
束
├── 预算限
制
├── 时间限
制
└── 人力限
制
└── 架构原
则
├── 设计原
则
└── 演进原
则

```
└──────────────────────────────────────────────────────────────────────────────
────┘
```

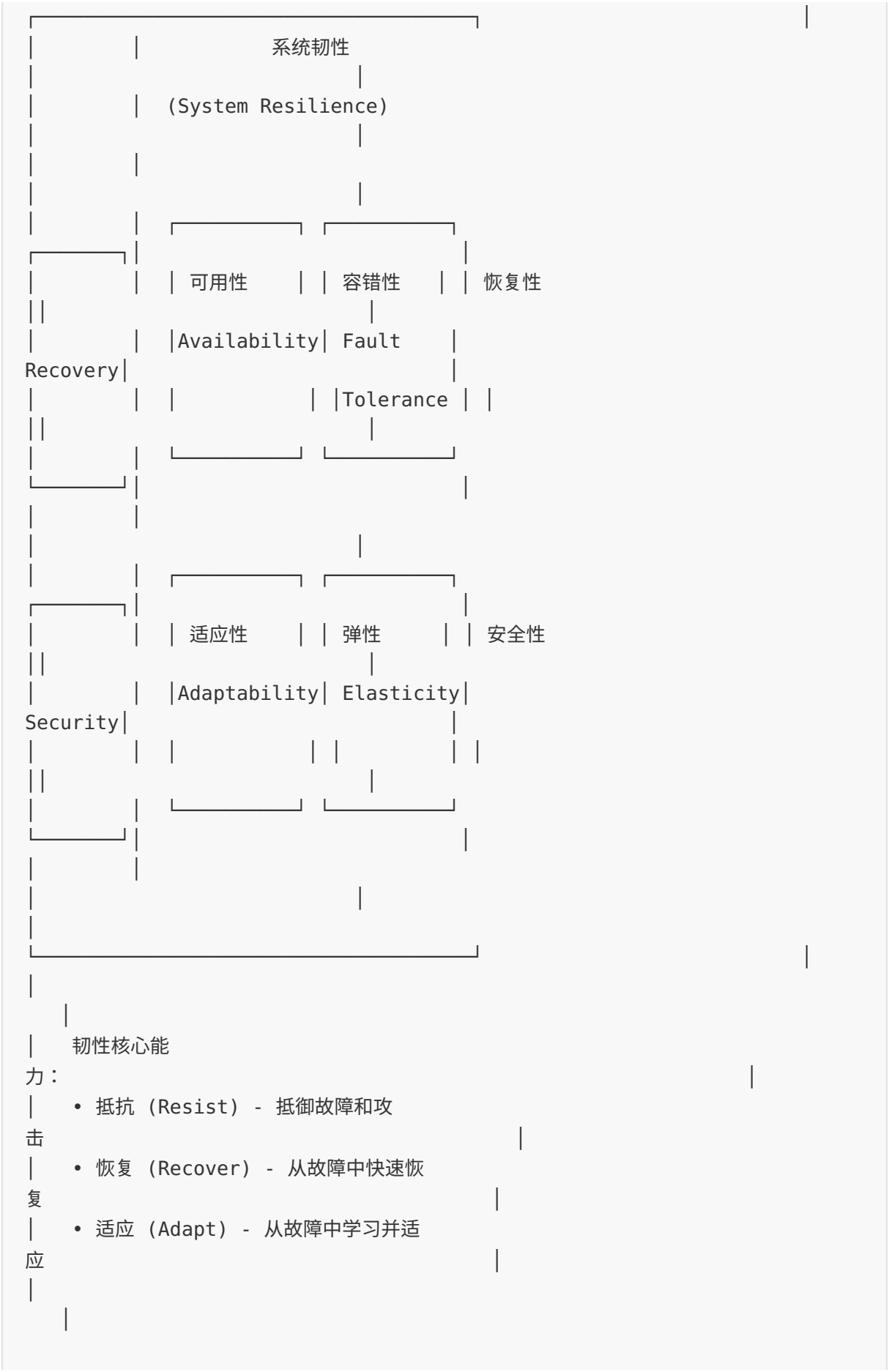
8.2 NFR与架构决策映射

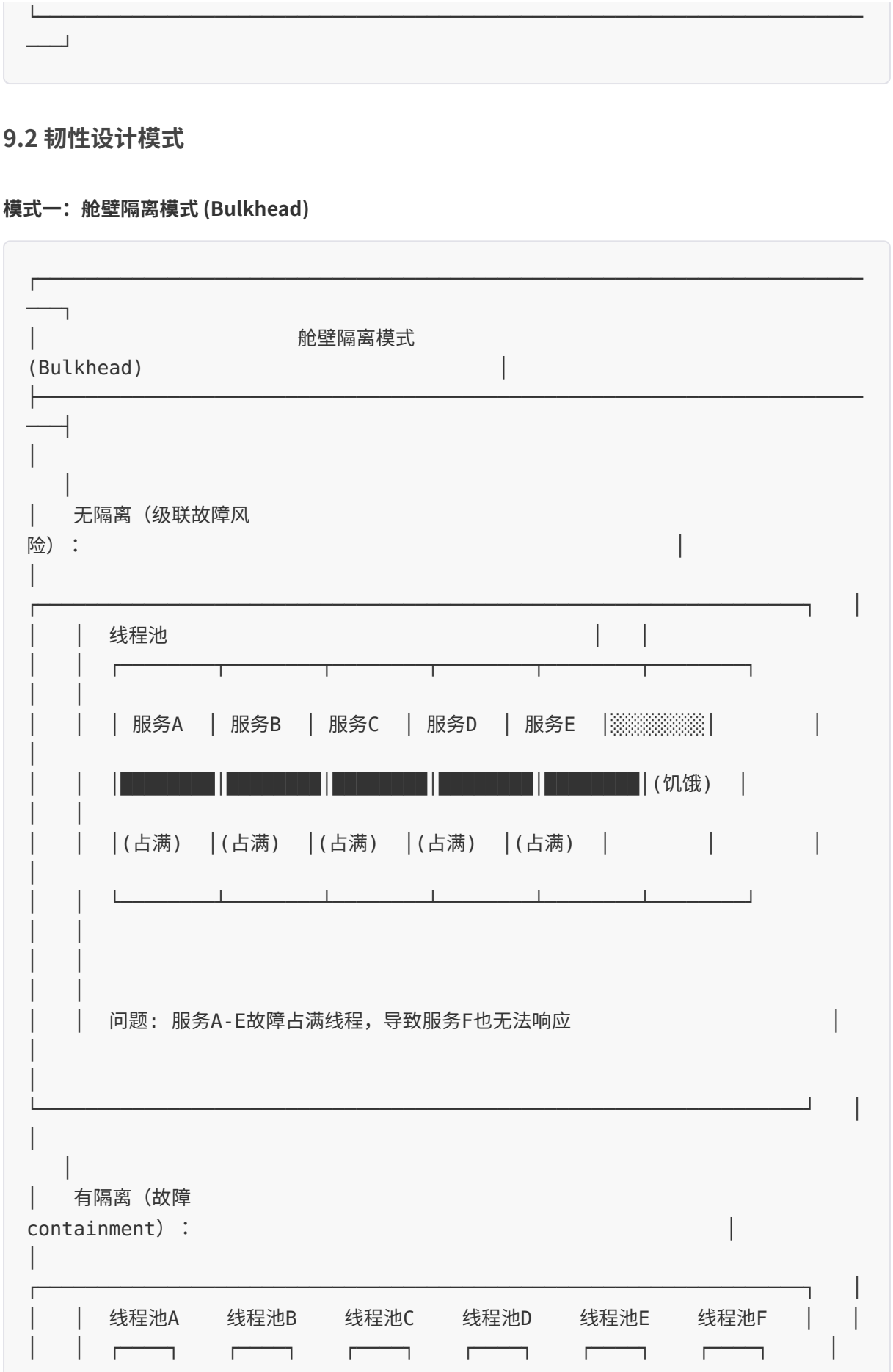
```
┌──────────────────────────────────────────────────────────────────────────────
────┐
│                          NFR与架构决策映射矩
阵                                    │
├──────────────────────────────────────────────────────────────────────────────
────┤
│
   │
│   NFR类别   具体需求     架构决策       技术选型              │
│
──────────────────────────────────────────────────────────────────────────────

│
│   性能         <100ms响应    缓存架构      Redis Cluster         │
│   可用性       99.999%           多活架构      同城双活+异地灾备     │
│   安全    等保三级     纵深防御      WAF+IDS+SIEM          │
│   可伸缩       10倍弹性     微服务+容器化    Kubernetes           │
│   可维护   MTTR<30分钟   可观测性架构    Prometheus+ELK        │
│   一致性   强一致      分布式事务      TCC/Saga              │
│
   │
│   NFR冲突解决策
略：                                                    │
│
┌──────────────────────────────────────────────────────────────────────┐   │
│   │
│   │
│   │   性能 vs 一致性                                                  │
│
│   │   ═══════════════
│   │
│   │   解决方案：区分场景，读操作最终一致，写操作强一致                    │   │
│   │
│   │
│   │   可用性 vs 一致性                                                 │
│
│   │   ═══════════════
│   │
│   │   解决方案：CAP权衡，核心业务CP，非核心AP                          │   │
```



第9章 系统韧性设计模式

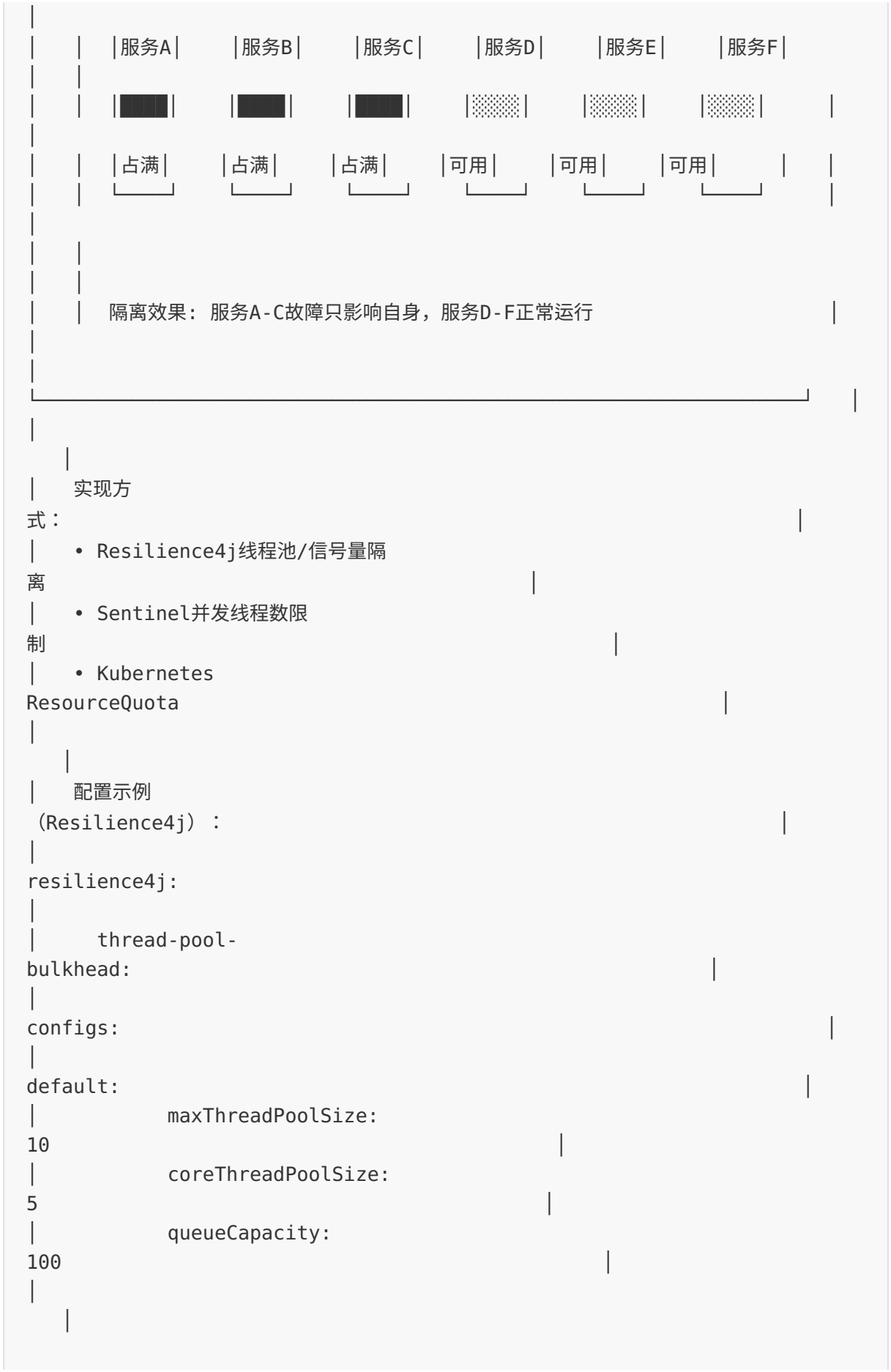
9.1 韧性设计原则


系统韧性
(System Resilience)
可用性
Availability
容错性
Fault Tolerance
恢复性
Recovery
适应性
Adaptability
弹性
Elasticity
安全性
Security
韧性核心能力：
• 抵抗 (Resist) - 抵御故障和攻击
• 恢复 (Recover) - 从故障中快速恢复
• 适应 (Adapt) - 从故障中学习并适应



9.2 韧性设计模式

模式一：舱壁隔离模式 (Bulkhead)


服务A
服务B
服务C
服务D
服务E
服务F
占满
占满
占满
可用
可用
可用
隔离效果: 服务A-C故障只影响自身，服务D-F正常运行
实现方式：
• Resilience4j线程池/信号量隔离
• Sentinel并发线程数限制
• Kubernetes ResourceQuota
配置示例（Resilience4j）：
resilience4j:
thread-pool-bulkhead:
configs:
default:
maxThreadPoolSize: 10
coreThreadPoolSize: 5
queueCapacity: 100

模式二：熔断器模式 (Circuit Breaker)

熔断器有三种状态：
- **CLOSED（闭合）**：正常状态，请求正常通过
- **OPEN（断开）**：故障状态，请求快速失败
- **HALF-OPEN（半开）**：探测状态，尝试恢复

模式三：限流模式 (Rate Limiting)

限流算法	特点	适用场景	金融适用度
计数器算法	实现简单，但有边界突发问题	简单限流	★★★
滑动窗口算法	平滑，无边界问题，但内存消耗大	精度要求高	★★★★
令牌桶算法	允许突发，长期速率恒定，推荐	API限流	★★★★★
漏桶算法	强制恒定速率，适合流量整形	流量整形	★★★

模式四：重试与退避模式

退避策略	特点	适用场景	金融建议
固定间隔	简单，可能有惊群效应	非关键操作	不推荐
线性退避	间隔逐渐增加	一般重试	可用
指数退避	快速增加间隔，有效分散压力	网络超时	推荐
指数退避+抖动	加随机避免同步重试	关键服务	金融推荐

第10章 数据一致性模式

10.1 金融数据一致性要求

一致性级别	定义	适用场景	实现技术	性能影响
强一致性	所有节点数据实时一致	核心账务、资金划转	2PC、Paxos/Raft	高
会话一致性	同一会话内数据一致	用户查询、交易明细	会话粘滞、读写同库	中
最终一致性	数据最终达到一致	余额查询、统计报表	异步复制、补偿机制	低
弱一致性	允许短暂不一致	日志记录、非关键数据	异步写入、批量同步	最低

10.2 分布式事务解决方案

分布式事务模式对比

模式	一致性	性能	复杂度	适用场景	金融示例
2PC	强一致	低	低	短事务、低并发	账户余额更新
TCC	最终一致	高	高	高并发、长事务	电商订单、支付
Saga	最终一致	高	中	长流程业务	贷款审批、理赔
本地消息表	最终一致	中	低	异步场景	通知、对账

10.3 金融场景一致性方案

场景一：跨行转账（Saga + TCC混合）

业务流程：

```
  (TCC)      (Saga)      (Saga)      (异步)
```

一致性保证：
1. 本行扣款：TCC模式，保证资金不丢失
   - Try：冻结客户资金
   - Confirm：实际扣款，记录转出流水
   - Cancel：解冻资金

2. 人行转账：调用超级网银接口，同步等待结果
   - 成功：继续他行入账
   - 失败：Saga补偿，调用本行冲正接口

3. 他行入账：异步确认
   - 定时查询入账结果
   - 超时未成功：人工介入处理

4. 通知客户：本地消息表，最终一致性

第11章 企业集成模式

11.1 集成架构演进

阶段	架构模式	特点	优缺点	适用场景
阶段1	点对点集成	直接连接，复杂度O(n^2)	简单但难以管理	系统数量<5
阶段2	ESB集中式	统一总线，松耦合	集中式风险	传统企业
阶段3	**API网关+事件驱动**	同步异步分离	云原生友好	**现代金融推荐**

11.2 消息队列选型

金融级消息队列对比

特性	Apache Kafka	RocketMQ	RabbitMQ	Pulsar
吞吐量	极高(百万级)	高(十万级)	中(万级)	高(十万级)

特性	Apache Kafka	RocketMQ	RabbitMQ	Pulsar
延迟	毫秒级	毫秒级	微秒级	毫秒级
可靠性	高	极高(金融级)	中	高
事务消息	支持	原生支持	有限支持	支持
顺序消息	分区有序	全局/分区有序	单队列有序	全局有序
延时消息	不支持	原生支持	插件支持	原生支持
金融推荐度	★★★	★★★★★	★★	★★★★

第12章 遗留系统现代化策略

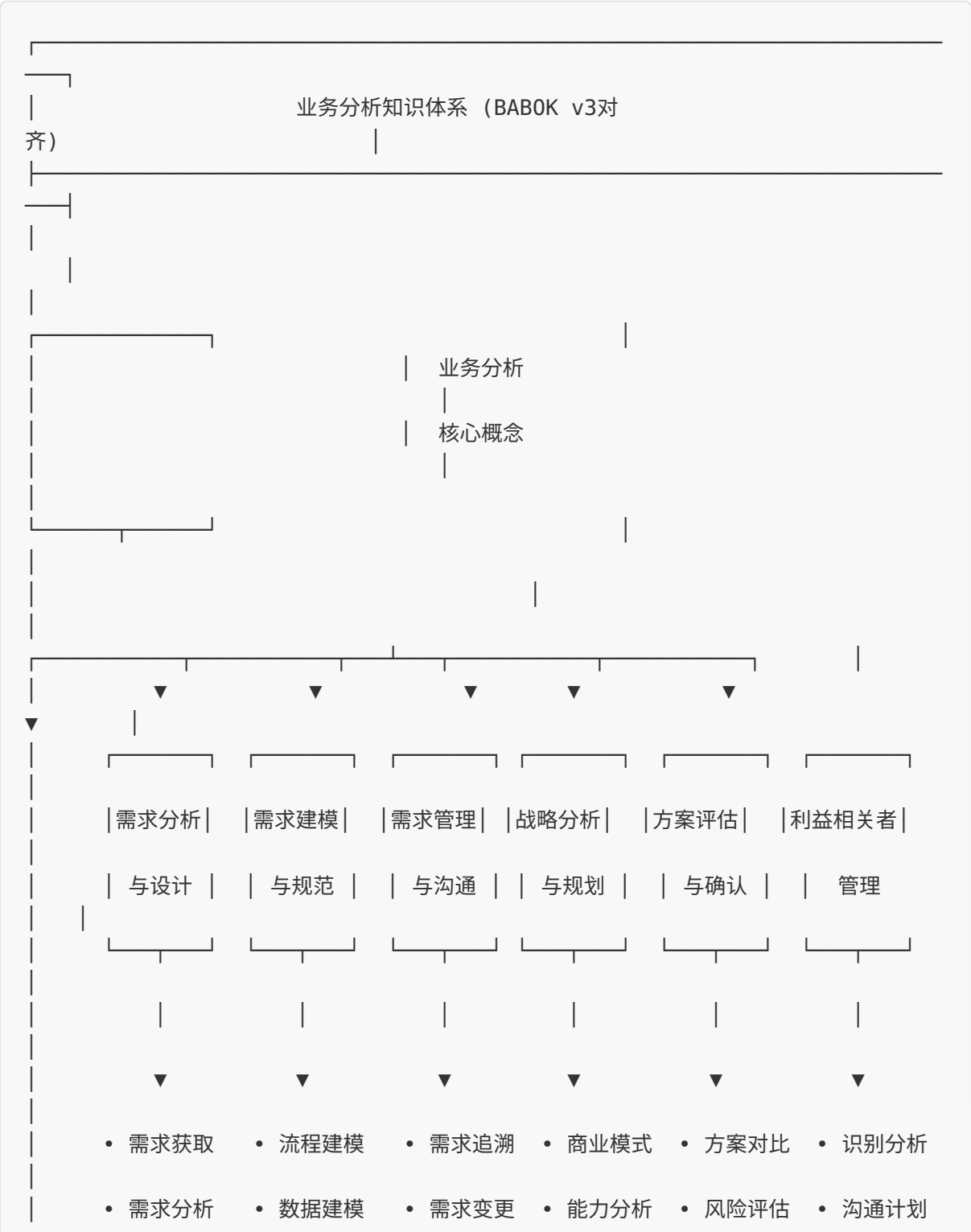
12.1 现代化策略矩阵

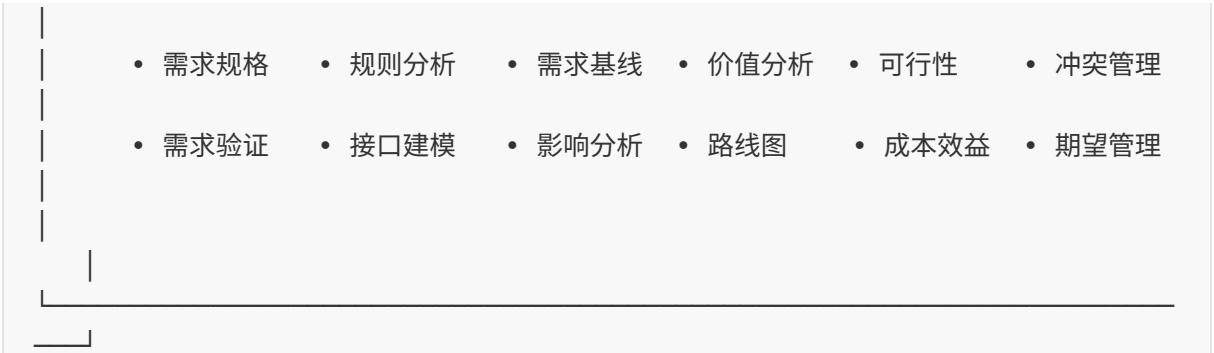
策略	业务价值	系统复杂度	风险	成本	周期
重构(Rebuild)	高	高	高	高	长
绞杀者(Strangler)	高	高	中	中	长
修缮者(Refactor)	中	中	低	低	中
封装(Encapsulate)	低	高	低	低	短
退役(Retire)	低	低	低	低	短

第二部分：业务分析师篇 - 金融业务分析与建模

第13章 业务分析思维框架

13.1 业务分析核心领域



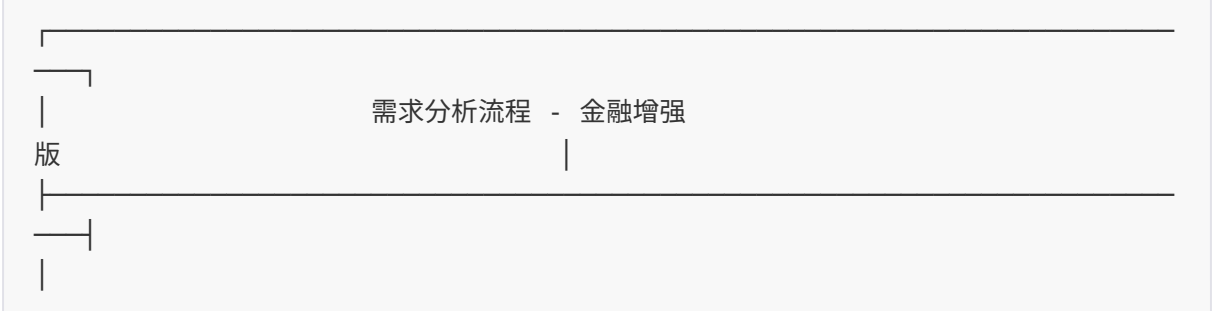


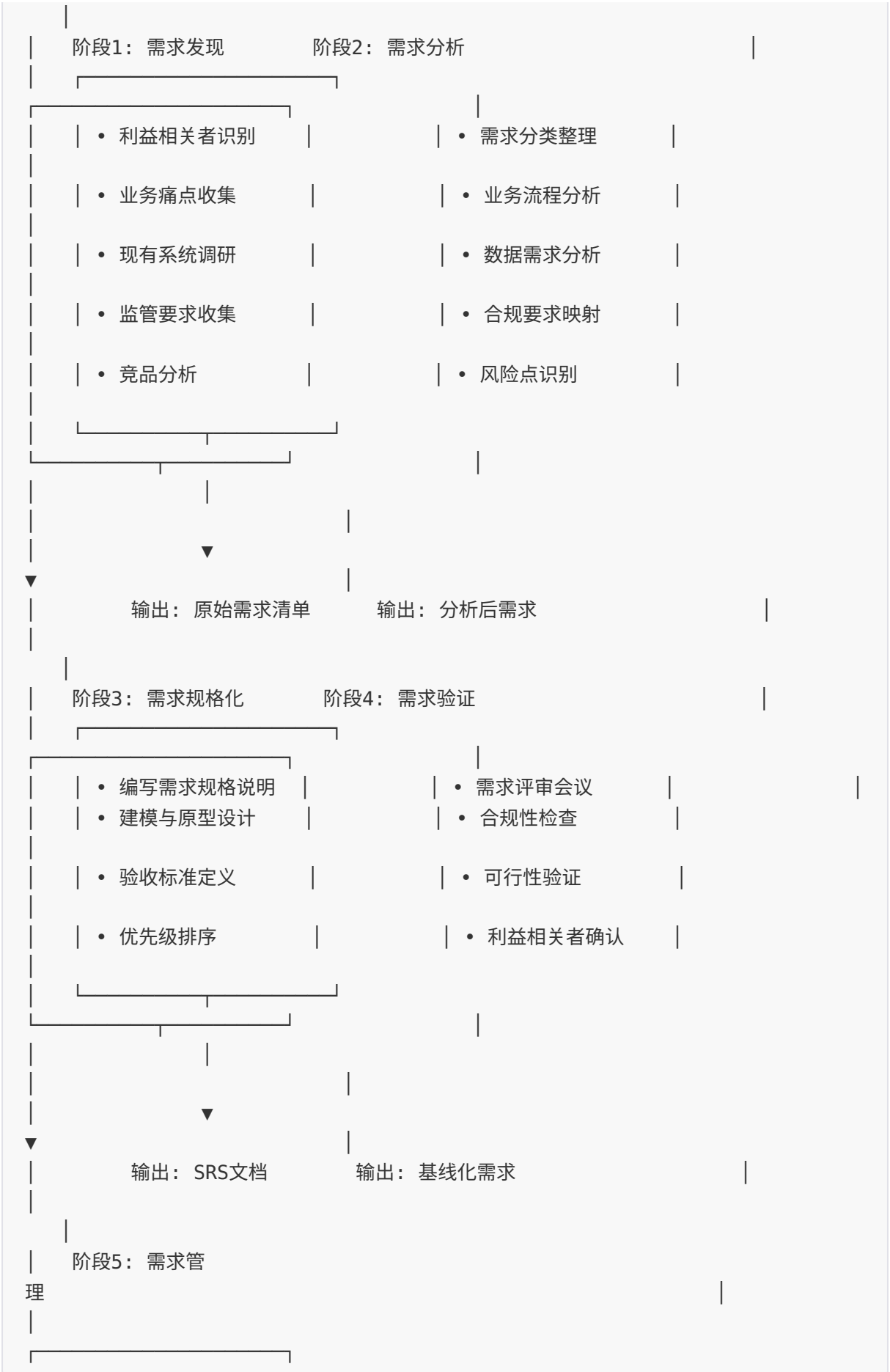
13.2 金融系统业务分析特殊性

维度	一般业务分析	金融系统业务分析	分析要点
合规要求	一般关注	高度监管敏感	法规遵循、监管报送
风险管理	常规评估	核心关注点	信用风险、市场风险、操作风险
数据精度	容忍一定误差	零容忍（资金）	精确计算、四舍五入规则
审计要求	基础记录	全流程可追溯	审计追踪、数据血缘
安全等级	标准安全	极高安全	数据分级、访问控制
时效要求	一般时效	实时/准实时	交易时效、SLA要求
复杂性	相对简单	高度复杂	产品多样性、业务规则

第14章 业务需求分析方法论

14.1 需求分析流程



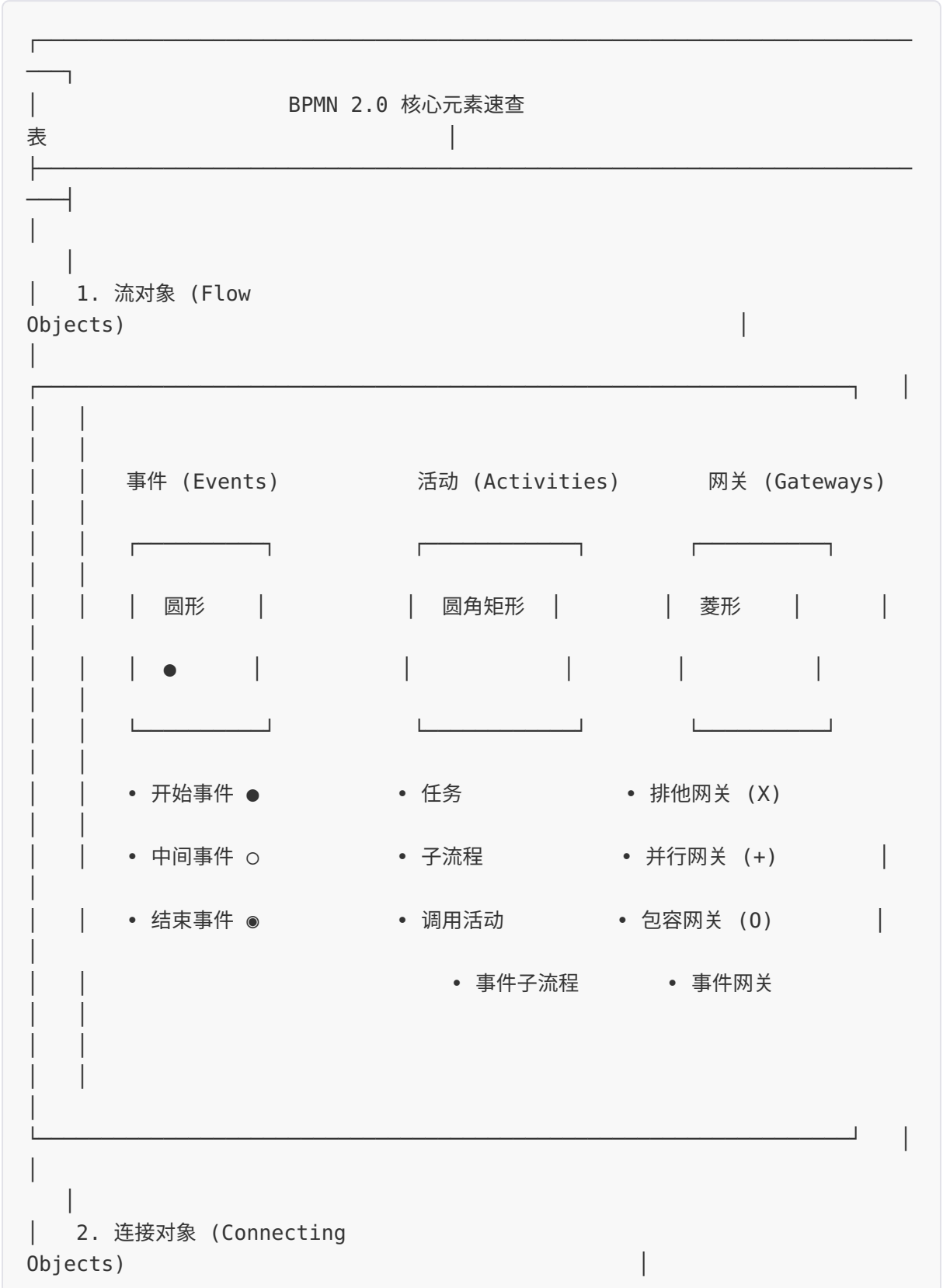

阶段1：需求发现
阶段2：需求分析
• 利益相关者识别
• 业务痛点收集
• 现有系统调研
• 监管要求收集
• 竞品分析
• 需求分类整理
• 业务流程分析
• 数据需求分析
• 合规要求映射
• 风险点识别
输出：原始需求清单
输出：分析后需求
阶段3：需求规格化
阶段4：需求验证
• 编写需求规格说明
• 建模与原型设计
• 验收标准定义
• 优先级排序
• 需求评审会议
• 合规性检查
• 可行性验证
• 利益相关者确认
输出：SRS文档
输出：基线化需求
阶段5：需求管理

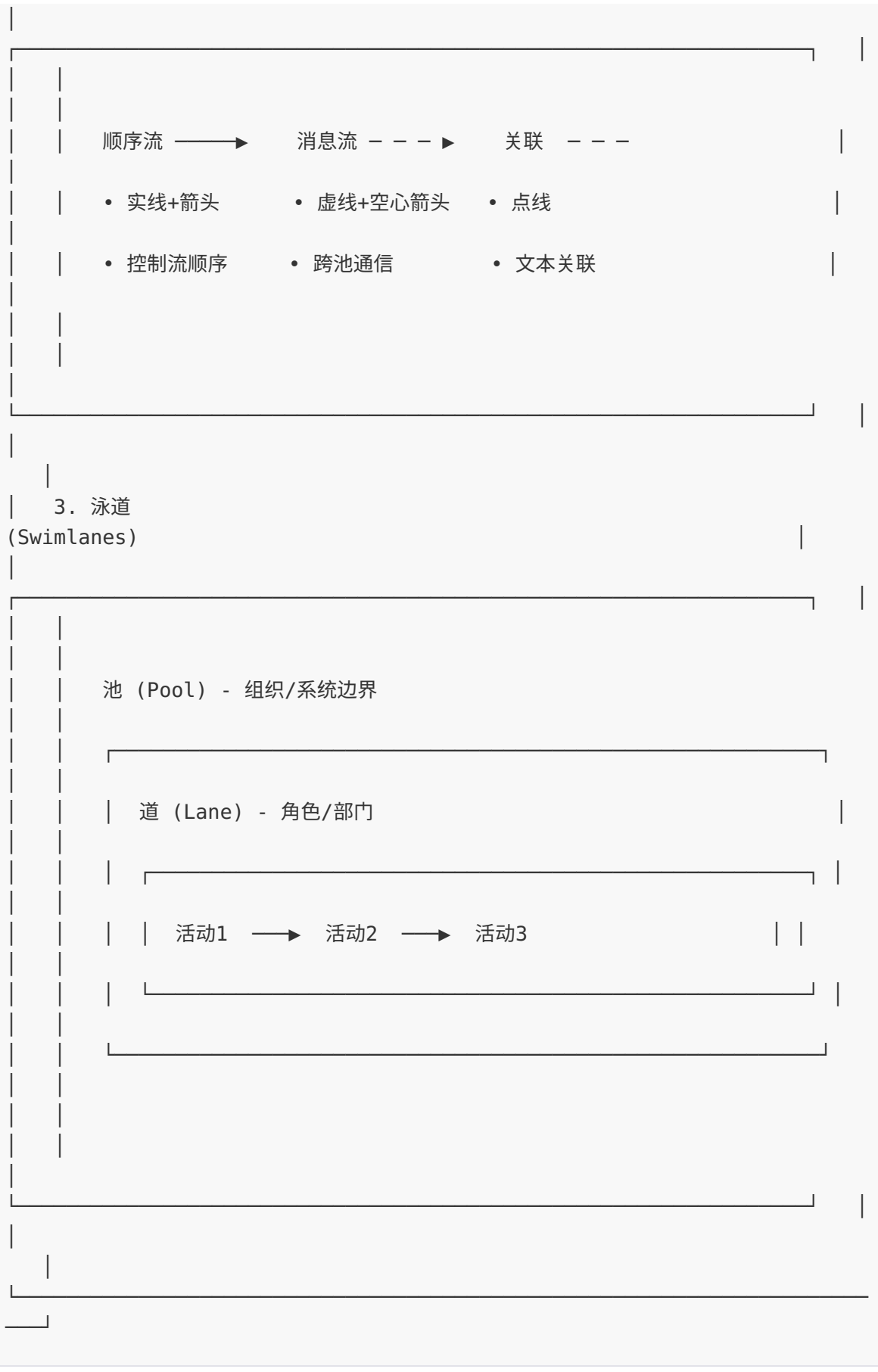
14.2 需求分类框架

需求类型	定义	示例	优先级方法
业务需求	业务目标和价值	提高客户开户效率50%	价值驱动
用户需求	用户需要完成的任务	客户可以在APP上完成开户	用户研究
功能需求	系统必须提供的功能	系统应支持OCR识别身份证	MoSCoW
非功能需求	系统质量属性	系统响应时间<1秒	技术标准
合规需求	法规监管要求	符合反洗钱法规	强制要求
接口需求	系统间集成需求	与核心系统实时对接	依赖分析
数据需求	数据处理需求	交易数据保存10年	监管要求
安全需求	安全控制需求	双因素认证	风险评估

第15章 业务流程建模（BPMN）

15.1 BPMN核心元素




顺序流
消息流
关联
• 实线+箭头
• 虚线+空心箭头
• 点线
• 控制流顺序
• 跨池通信
• 文本关联
3. 泳道
(Swimlanes)
池 (Pool) - 组织/系统边界
道 (Lane) - 角色/部门
活动1
活动2
活动3

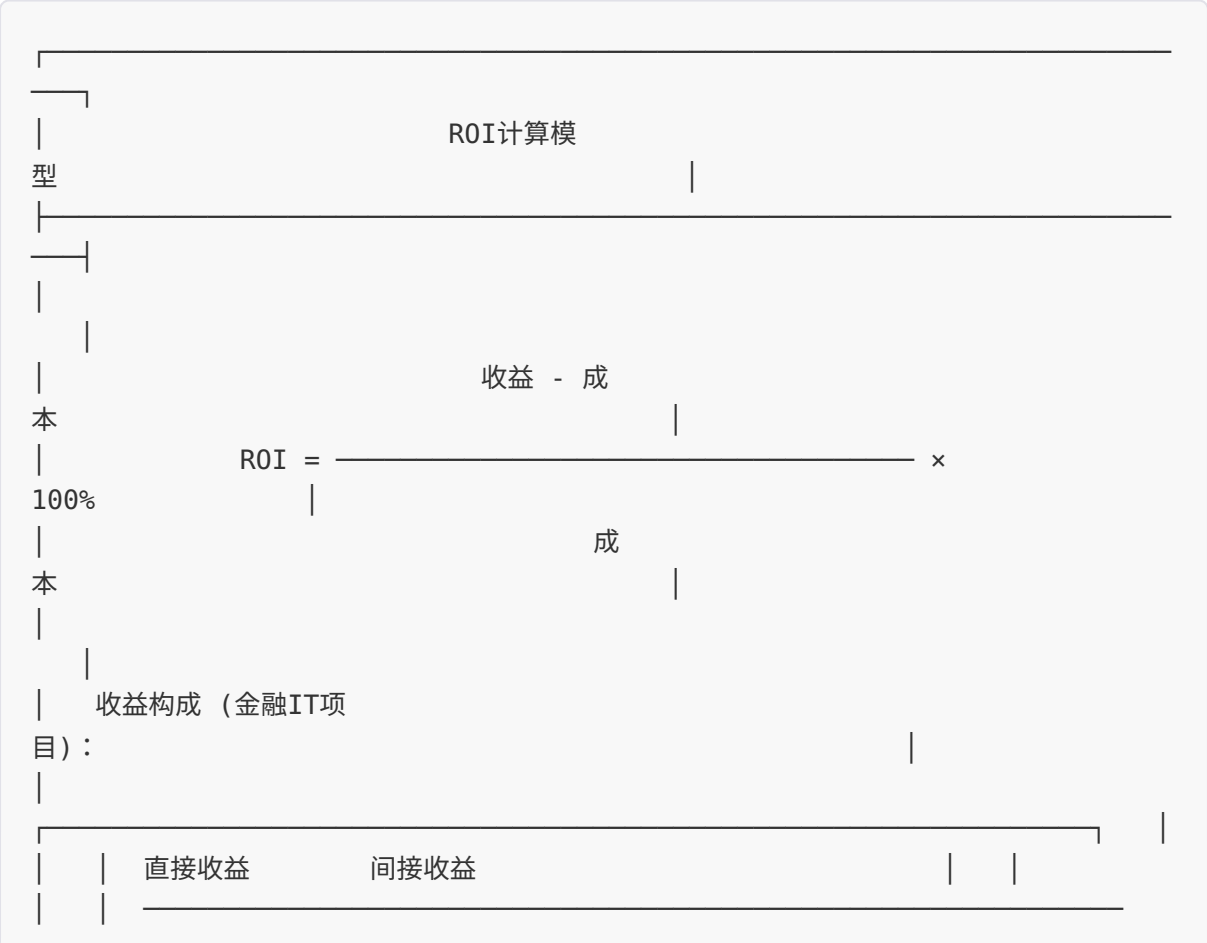
15.2 业务流程优化方法

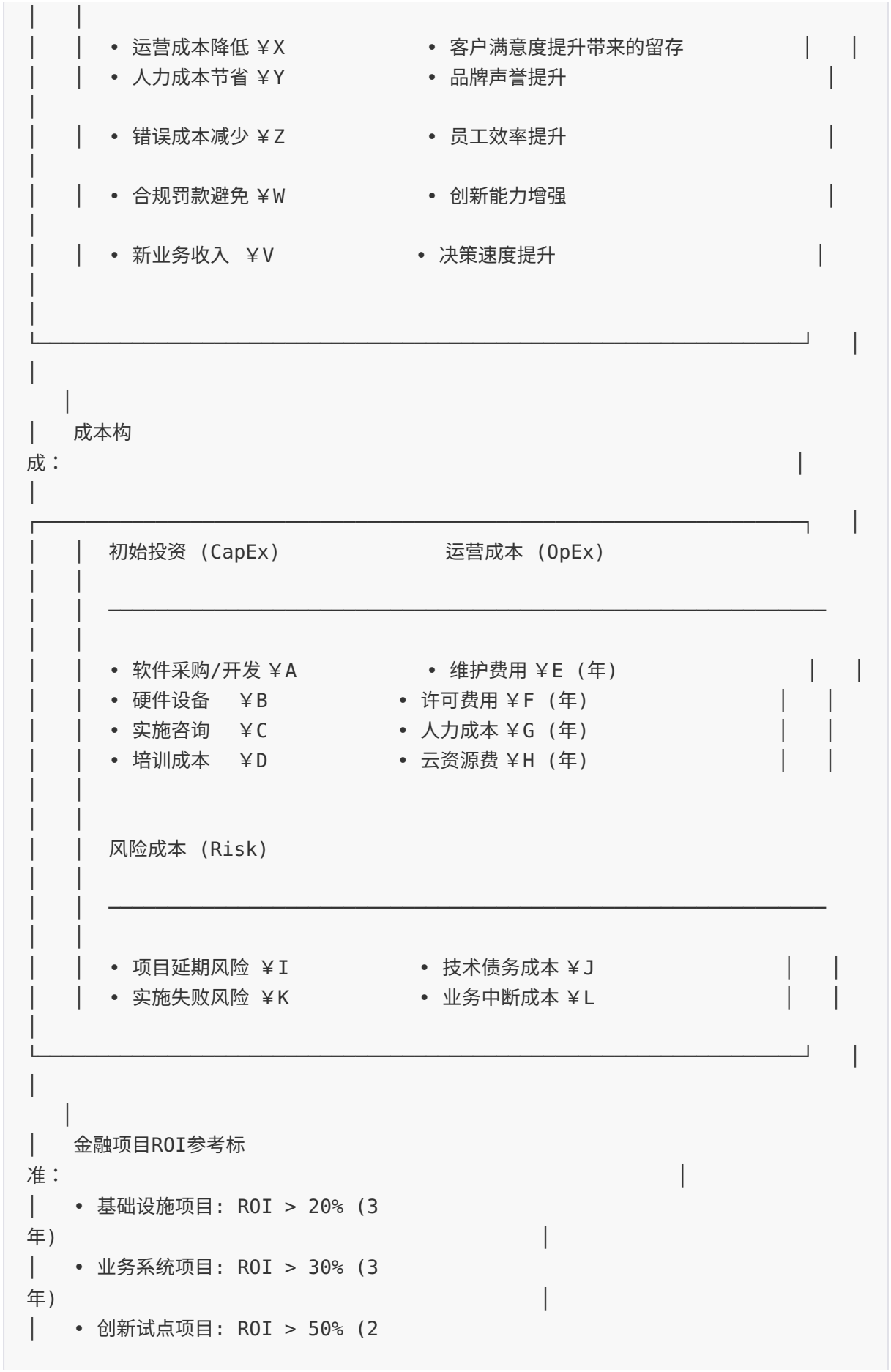
ESIA优化法

方法	说明	金融示例
E-消除	消除非增值活动	重复录入、不必要的审批
S-简化	简化复杂流程	智能表单、自动计算
I-整合	整合分散流程	统一工作界面、一站式服务
A-自动化	自动化手工操作	RPA、OCR、规则引擎

第16章 业务价值分析

16.1 ROI与TCO分析



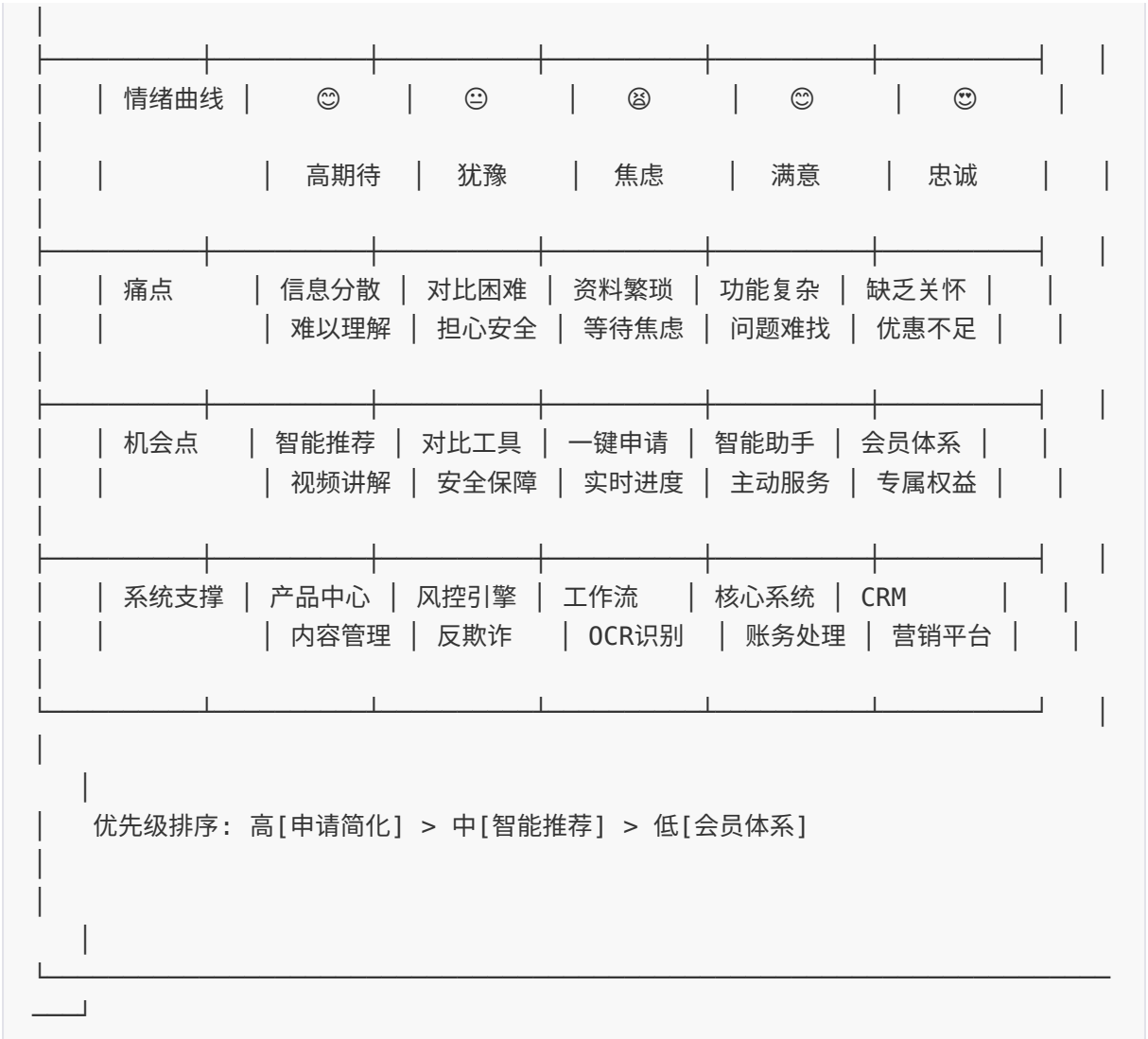

• 运营成本降低 ￥X
• 人力成本节省 ￥Y
• 错误成本减少 ￥Z
• 合规罚款避免 ￥W
• 新业务收入 ￥V
• 客户满意度提升带来的留存
• 品牌声誉提升
• 员工效率提升
• 创新能力增强
• 决策速度提升
成本构成：
初始投资 (CapEx)
运营成本 (OpEx)
• 软件采购/开发 ￥A
• 硬件设备 ￥B
• 实施咨询 ￥C
• 培训成本 ￥D
• 维护费用 ￥E (年)
• 许可费用 ￥F (年)
• 人力成本 ￥G (年)
• 云资源费 ￥H (年)
风险成本 (Risk)
• 项目延期风险 ￥I
• 实施失败风险 ￥K
• 技术债务成本 ￥J
• 业务中断成本 ￥L
金融项目ROI参考标准：
• 基础设施项目: ROI > 20% (3年)
• 业务系统项目: ROI > 30% (3年)
• 创新试点项目: ROI > 50% (2

```
年)                                                    |
|
   |
└──────────────────────────────────────────────────────────────────────────
──┘
```

第17章 用户研究与体验分析

17.1 用户旅程地图

```
┌──────────────────────────────────────────────────────────────────────────
──┐
|                    用户旅程地图 (User Journey
Map)                          |
├──────────────────────────────────────────────────────────────────────────
──┤
|
   |
|   用户画像: [典型用户描
述]                                                    |
|   场景: [用户要完成的目
标]                                                    |
|   时间跨度: [旅程开始-结
束]                                                   |
|
   |
|
┌──────┬──────┬──────┬──────┬──────┬──────┐    |
|   |   阶段   |   认知   |   考虑   |   申请   |   使用   |   忠诚   |    |
|
├──────┼──────┼──────┼──────┼──────┼──────┤    |
|   | 用户行为 | 搜索产品 | 对比方案 | 填写资料 | 使用服务 | 推荐他人 |    |
|   |          | 阅读介绍 | 计算利率 | 提交申请 | 管理账户 | 复购产品 |    |
|
├──────┼──────┼──────┼──────┼──────┼──────┤    |
|   | 触点渠道 | 搜索引擎 | 官网/App | App/Web  | App/Web  | App/客服 |    |
|   |          | 社交媒体 | 客服咨询 | 线下网点 | 短信通知 | 社群     |    |
|
├──────┼──────┼──────┼──────┼──────┼──────┤    |
|   | 用户想法 | 利率如何 | 安全吗   | 流程复杂 | 到账速度 | 服务如何 |    |
|   |          | 额度多少 | 快不快   | 资料多吗 | 使用方便 | 值得信任 |    |
```



第18章 需求优先级排序方法

18.1 MoSCoW方法

优先级	定义	比例	金融示例
M-Must	必须有，否则项目失败	60%	资金转账、反洗钱检查
S-Should	应该有，短期有替代方案	20%	批量转账、多币种支持
C-Could	可以有，锦上添花	15%	语音搜索、暗黑模式

优先级	定义	比例	金融示例
W-Won't	不会有/暂不	5%	区块链支付（本期）

18.2 Kano模型

需求类型	特点	金融示例	策略
基本需求	没有不满意，有了也不满意	资金安全、交易准确	必须100%满足
期望需求	做得越好越满意	操作便捷、响应快速	持续优化
魅力需求	意想不到，高度满意	智能推荐、场景服务	差异化竞争

18.3 WSJF方法

$$WSJF = \frac{\text{业务价值} + \text{时间紧迫性} + \text{风险降低/机会启用}}{\text{作业规模}}$$

第19章 领域分析方法（DDD）

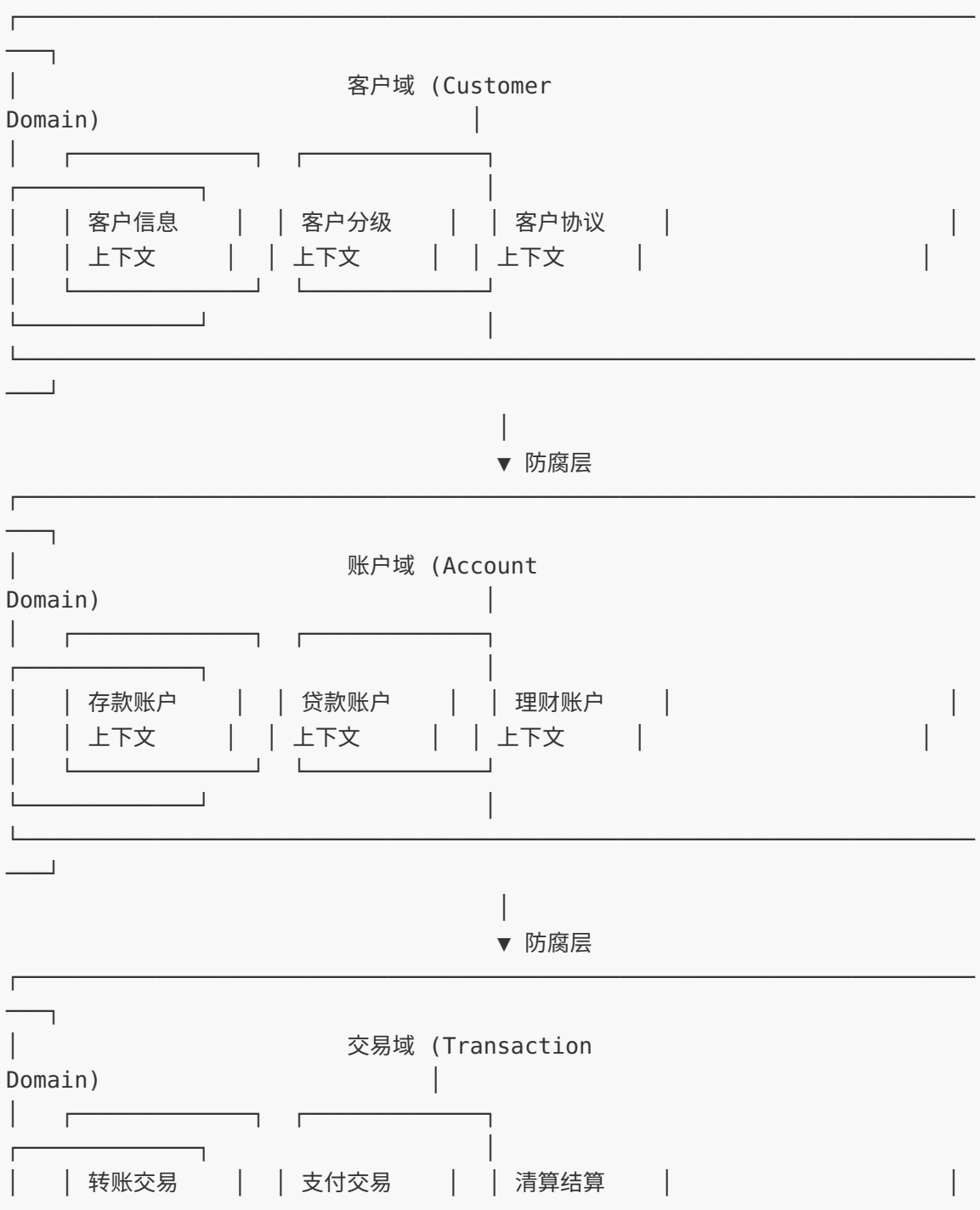
19.1 DDD核心概念

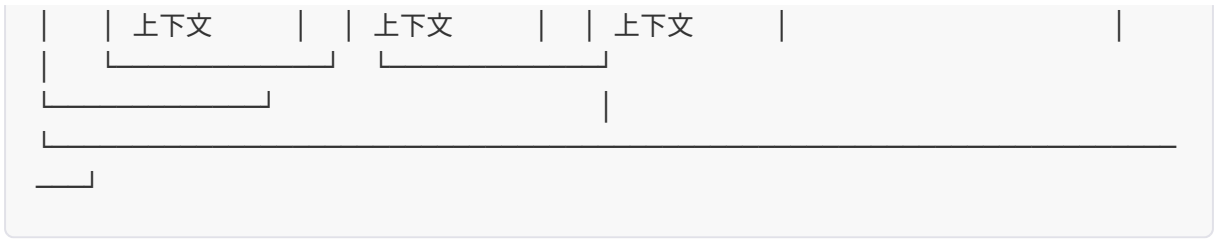
概念	定义	金融示例
限界上下文	领域边界，统一语言范围	客户域、账户域、交易域
实体	有唯一标识的对象	客户、账户、交易订单
值对象	无标识，可替换的对象	地址、金额、身份证号
聚合	一致性的边界	订单聚合（订单+订单项）
领域事件	领域内发生的事	账户已创建、资金已划转

概念	定义	金融示例
仓储	持久化抽象	账户仓储、交易仓储

19.2 金融限界上下文划分

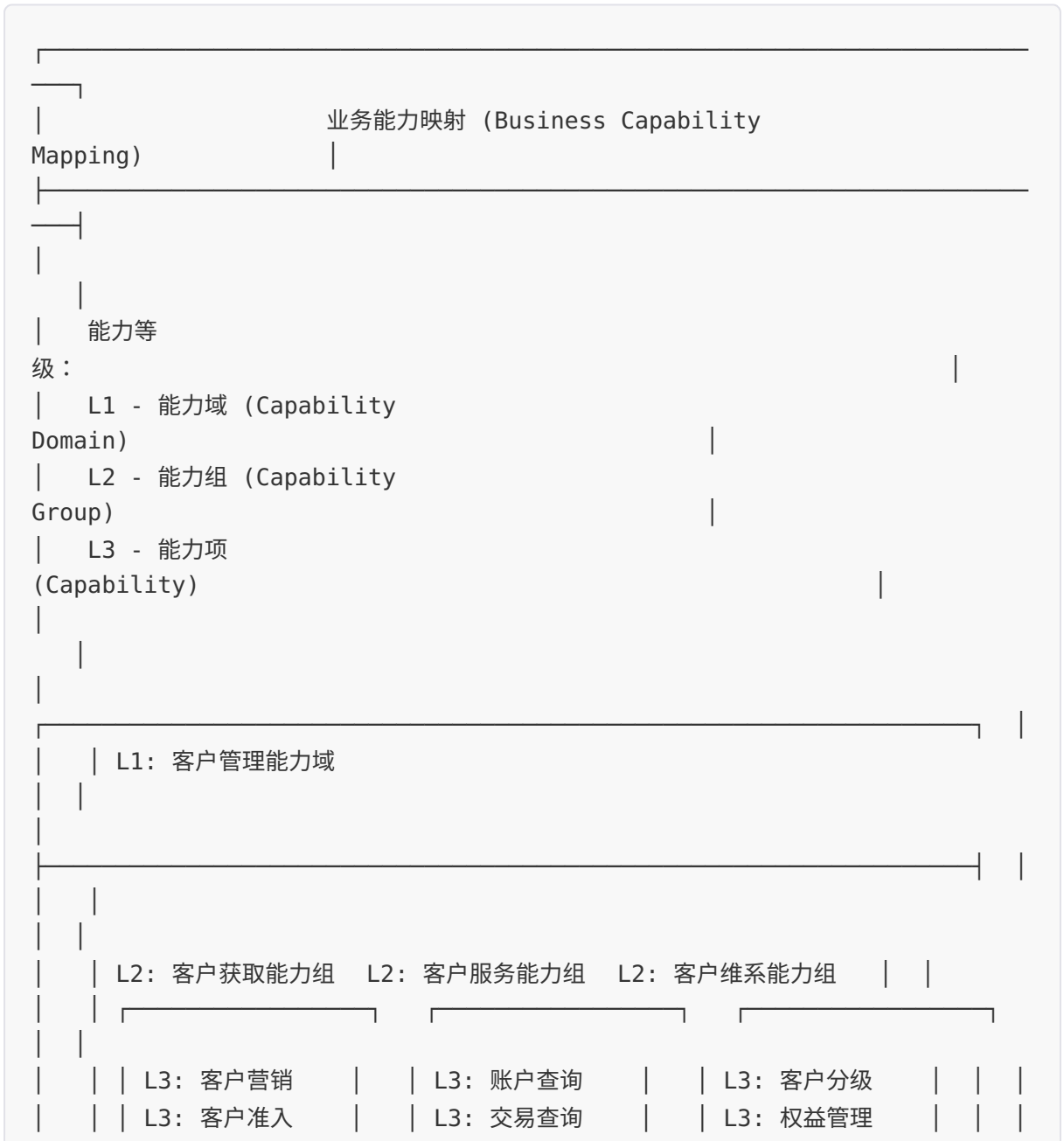
银行核心系统限界上下文示例：

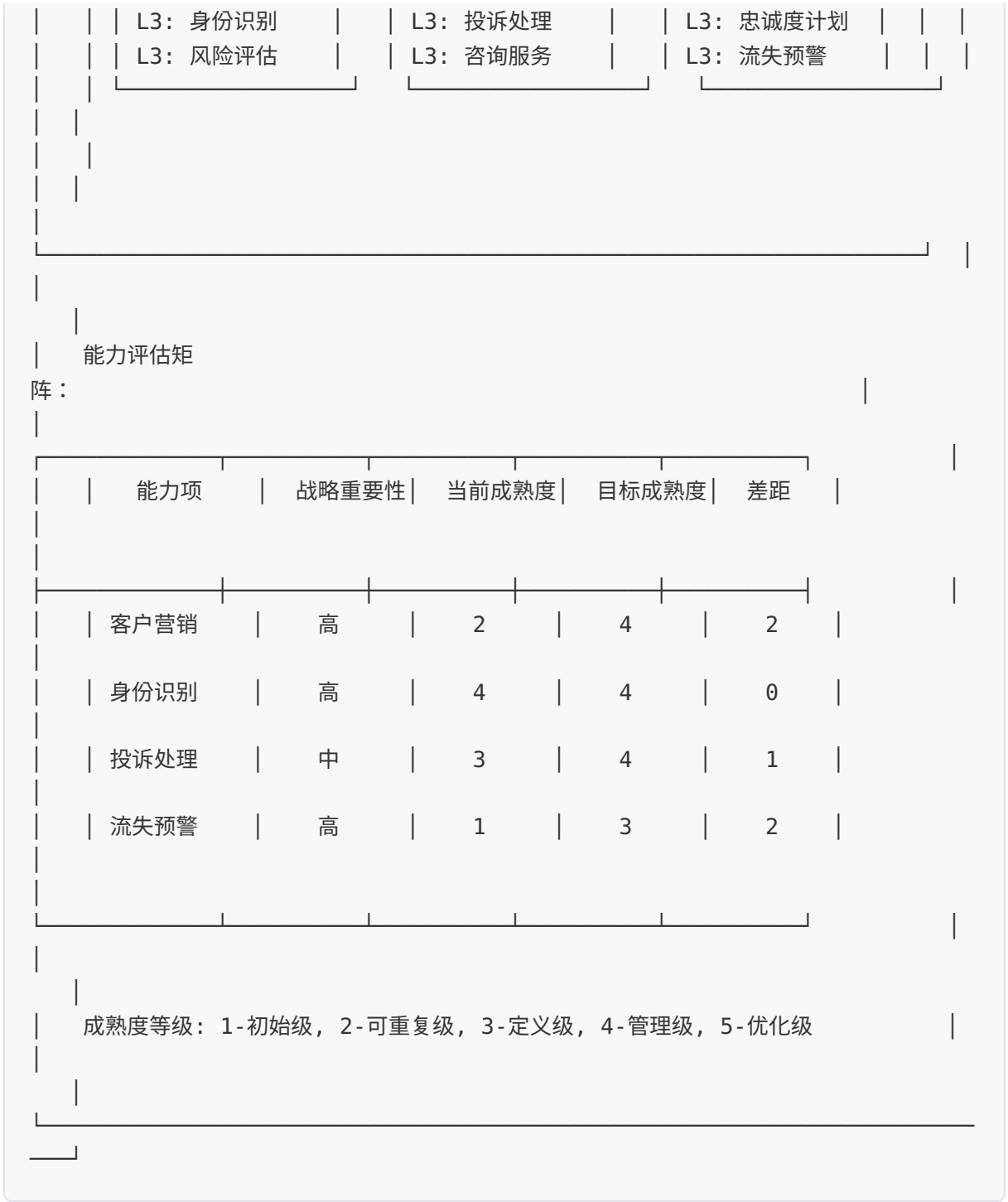




第20章 业务能力映射

20.1 业务能力框架




L3: 身份识别
L3: 风险评估
L3: 投诉处理
L3: 咨询服务
L3: 忠诚度计划
L3: 流失预警
能力评估矩阵：
能力项 | 战略重要性 | 当前成熟度 | 目标成熟度 | 差距
客户营销 | 高 | 2 | 4 | 2
身份识别 | 高 | 4 | 4 | 0
投诉处理 | 中 | 3 | 4 | 1
流失预警 | 高 | 1 | 3 | 2
成熟度等级: 1-初始级, 2-可重复级, 3-定义级, 4-管理级, 5-优化级

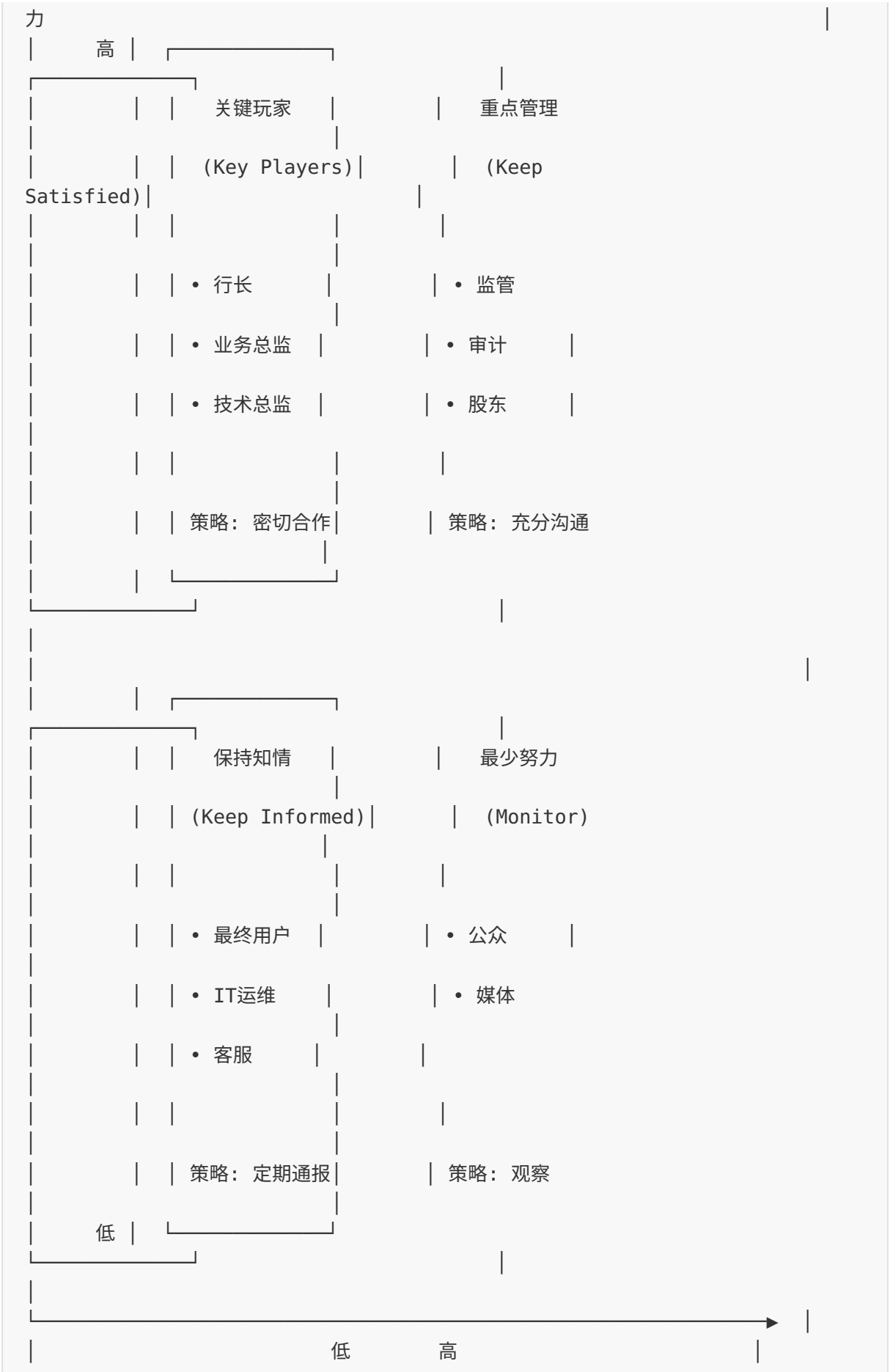
第21章 监管需求分析

21.1 金融监管框架

层面	法规/标准	主要内容
国家层面	网络安全法	数据安全、关键信息基础设施保护
	数据安全法	数据分类分级、数据安全审查
	个人信息保护法	个人信息处理规则、跨境传输
	密码法	密码应用安全性评估
银行业	商业银行法	银行业务规范
	巴塞尔协议III	资本充足率、流动性要求
	反洗钱法	客户身份识别、可疑交易报告
技术标准	等保2.0	网络安全等级保护
	JR/T 0171-2020	个人金融信息保护技术规范
	JR/T 0197-2020	金融数据安全分级指南

第22章 干系人分析与管理

22.1 干系人分析矩阵


力
高
关键玩家
(Key Players)
重点管理
(Keep Satisfied)
• 行长
• 业务总监
• 技术总监
• 监管
• 审计
• 股东
策略: 密切合作
策略: 充分沟通
保持知情
(Keep Informed)
最少努力
(Monitor)
• 最终用户
• IT运维
• 客服
• 公众
• 媒体
策略: 定期通报
策略: 观察
低
低
高

第23章 业务场景分析

23.1 场景分析框架

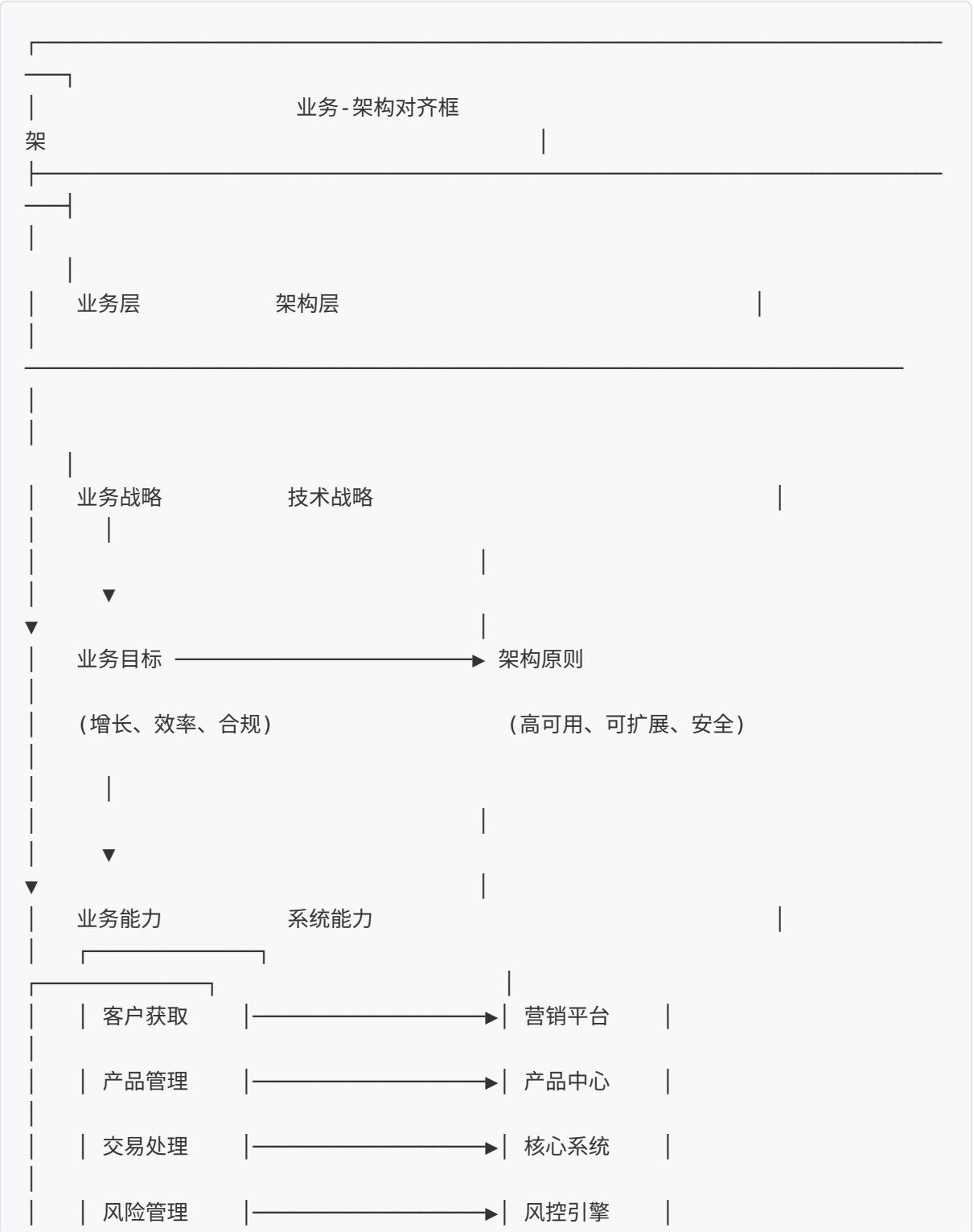
业务场景分析模板

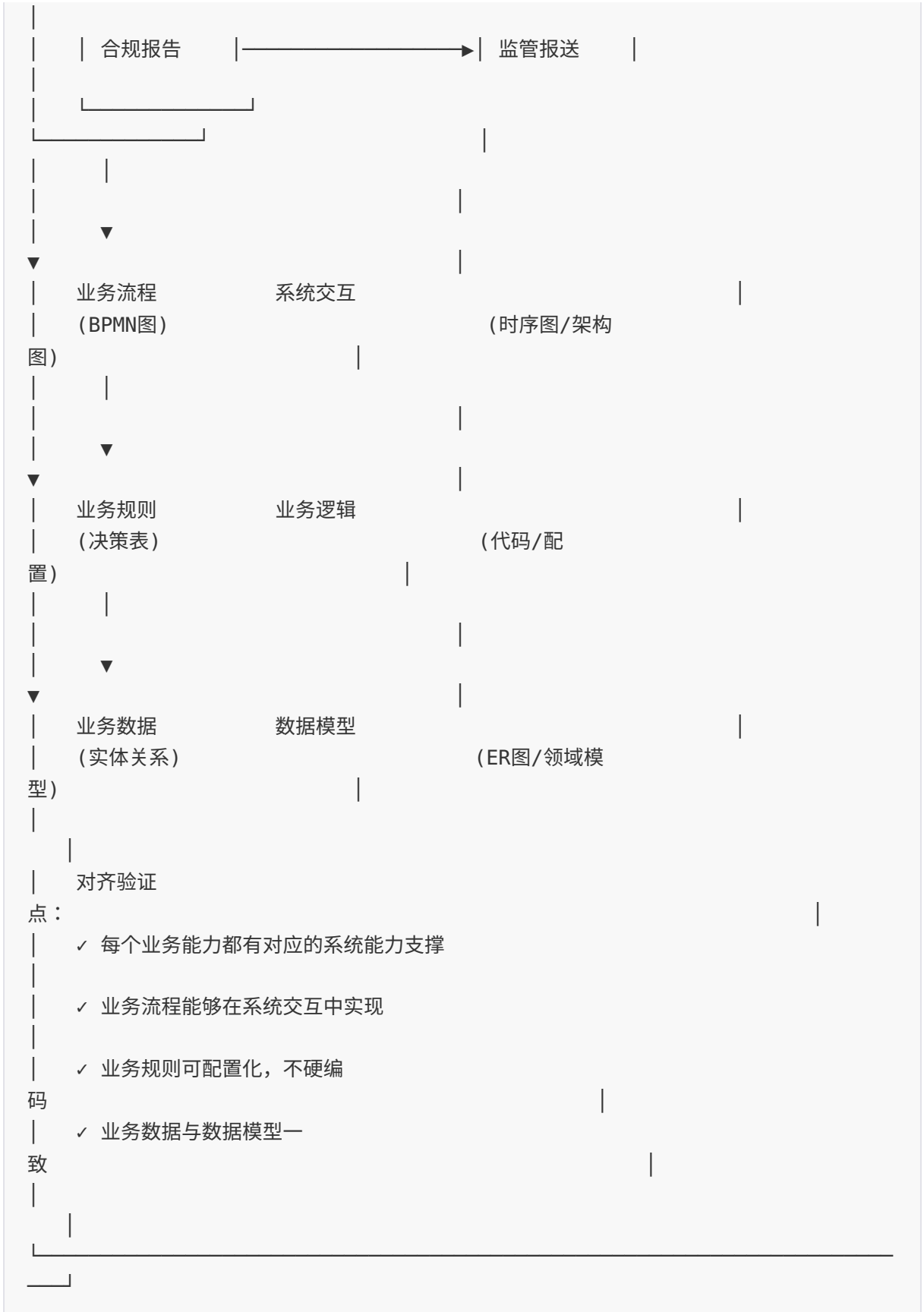
章节	内容	说明
1. 场景描述	在什么情况下，谁，要做什么	一句话描述场景
2. 触发条件	什么事件触发这个场景	可以是用户操作或系统事件
3. 参与者	主要参与者、次要参与者	人、系统、外部服务
4. 前置条件	执行场景前必须满足的条件	状态、权限、数据
5. 基本流程	主成功场景的正常步骤	使用编号列表
6. 扩展流程	替代场景和分支	6a, 6b...
7. 异常流程	错误场景和处理	7a, 7b...
8. 后置条件	场景执行后的系统状态	数据变化、状态更新
9. 业务规则	必须遵守的规则	规则编号+描述
10. 非功能需求	性能、安全、可用性要求	可量化的指标

第三部分：架构与业务融合篇 - 跨职能协作实践

第24章 架构与业务融合方法

24.1 从业务到架构的映射




合规报告
监管报送
业务流程
(BPMN图)
系统交互
(时序图/架构图)
业务规则
(决策表)
业务逻辑
(代码/配置)
业务数据
(实体关系)
数据模型
(ER图/领域模型)
对齐验证点：
✓ 每个业务能力都有对应的系统能力支撑
✓ 业务流程能够在系统交互中实现
✓ 业务规则可配置化，不硬编码
✓ 业务数据与数据模型一致

24.2 业务架构与技术架构协同

阶段	业务架构输出	技术架构输入	协同活动	交付物
战略对齐	业务战略、能力地图	技术愿景、原则	架构委员会会议	架构蓝图
需求分析	业务需求、流程模型	技术约束、标准	联合工作坊	需求规格
方案设计	业务能力映射	系统组件设计	架构评审	设计文档
实施规划	业务能力优先级	技术依赖分析	迭代计划会	实施路线图
交付验证	业务验收标准	技术验收标准	UAT测试	验收报告

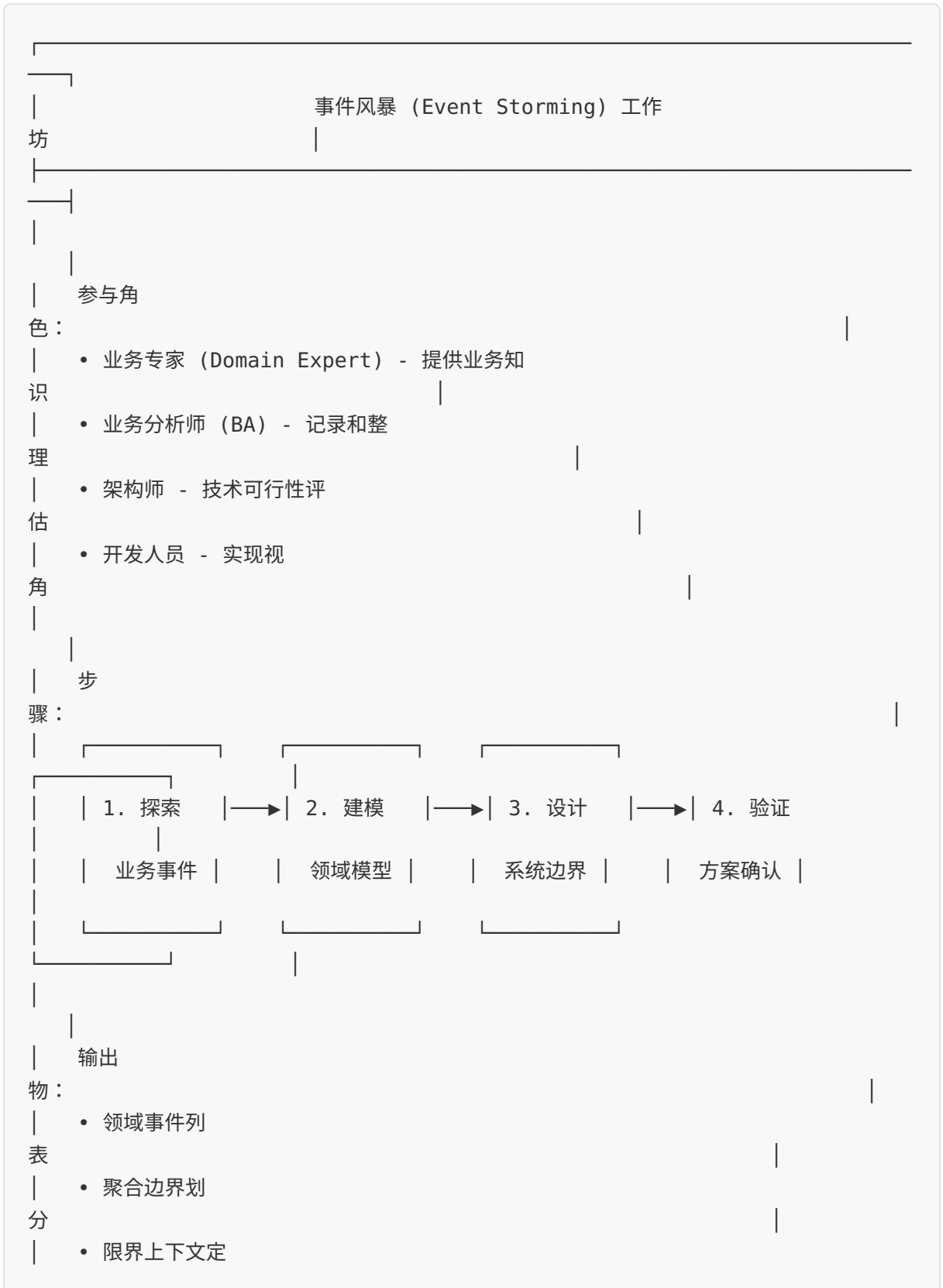
第25章 从业务需求到架构设计

25.1 需求到架构的转换矩阵

业务需求类型	架构关注点	设计决策	技术实现
功能需求	组件划分	微服务拆分	服务边界定义
性能需求	容量规划	缓存策略	Redis集群
可用性需求	冗余设计	多活架构	同城双活
安全需求	安全控制	纵深防御	WAF+零信任
合规需求	审计追踪	不可篡改	区块链存证
扩展性需求	弹性设计	云原生	Kubernetes

第26章 协作模式与工作坊

26.1 事件风暴工作坊



义 |
| • 微服务拆分建
议 |
|
|
└──
──┘

26.2 架构-业务协作检查清单

检查项	业务分析师	架构师	协作产出
需求理解	业务需求文档	技术可行性评估	确认的需求范围
流程梳理	业务流程图	系统交互图	端到端流程
规则定义	业务规则表	规则引擎设计	可配置规则
数据建模	业务实体	数据架构	领域模型
验收标准	业务验收条件	技术验收条件	统一验收标准

第27章 实战案例详解

案例1：大型商业银行核心系统重构

项目背景

- 银行规模：资产规模超5万亿
- 原核心系统：20年历史的AS/400主机系统
- 业务痛点：产品创新慢、扩展性差、维护成本高

架构决策

决策点	选项A	选项B	选择	理由
架构风格	分布式核心	主机延伸	分布式	自主可控、弹性扩展
数据库	OceanBase	TiDB	OceanBase	金融级可靠性、兼容Oracle

决策点	选项A	选项B	选择	理由
迁移策略	绞杀者模式	大爆炸	绞杀者	风险可控、渐进式迁移

实施成果

- 产品上线周期从3个月缩短到2周
- 日交易量从5000万提升到2亿
- 系统可用性从99.9%提升到99.99%
- 年度运维成本降低40%

案例2：互联网银行风控中台建设

项目背景

- 纯互联网银行，无线下网点
- 挑战：线上获客成本高、欺诈风险大
- 目标：建设智能风控中台，实现秒级审批

业务成效

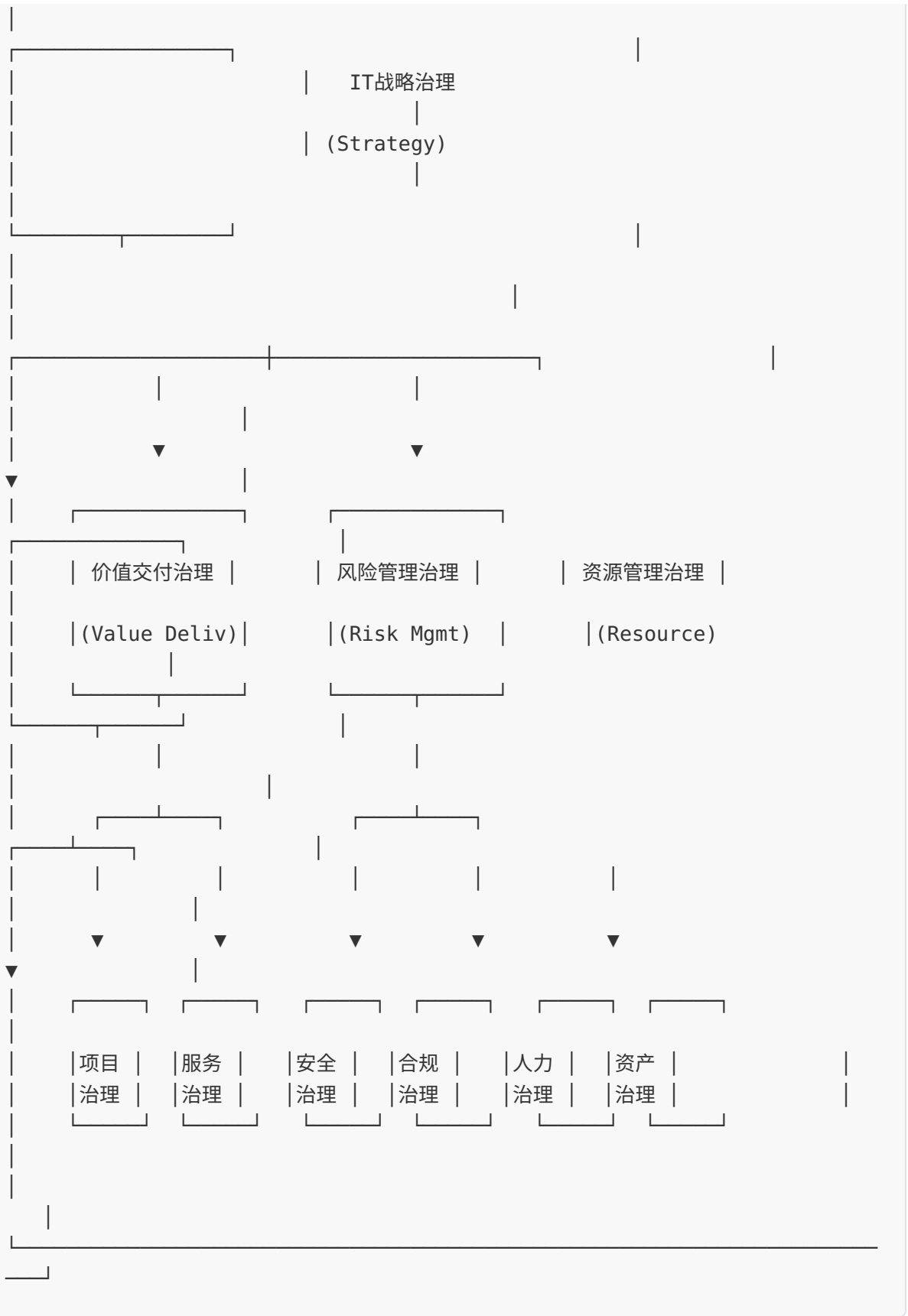
- 贷款审批时间从小时级缩短到秒级
- 自动审批率提升到85%
- 欺诈损失率降低60%
- 坏账率控制在1.5%以下

第四部分：IT治理篇 - 架构治理与风险管理

第28章 IT治理框架

28.1 IT治理核心领域

```
┌──────────────────────────────────────────────────────────────────────────
──┐
│                    IT治理体系框架 (COBIT/ITIL/ISO
38500)                    │
├──────────────────────────────────────────────────────────────────────────
──┤
│
   │
```


IT战略治理
(Strategy)
价值交付治理
(Value Deliv)
风险管理治理
(Risk Mgmt)
资源管理治理
(Resource)
项目治理
服务治理
安全治理
合规治理
人力治理
资产治理

第29章 架构治理

29.1 架构治理框架

治理领域	治理目标	治理措施	度量指标
架构合规	确保系统符合架构标准	架构评审、合规检查	合规率 > 95%
技术债务	控制技术债务增长	债务登记、偿还计划	债务占比 < 15%
架构演进	推动架构持续优化	演进路线图、版本管理	按计划完成率
知识管理	保障架构知识传承	文档管理、培训体系	文档完整度
决策质量	提升架构决策质量	ADR管理、决策评审	决策返工率 < 5%

29.2 架构委员会运作机制

评审级别	影响范围	审批人	评审内容
L1级	单系统	部门架构师	技术选型、接口变更
L2级	多系统	架构委员会	跨系统集成、重大变更
L3级	架构方向	CTO/委员会	技术战略、架构演进

第30章 技术债务管理

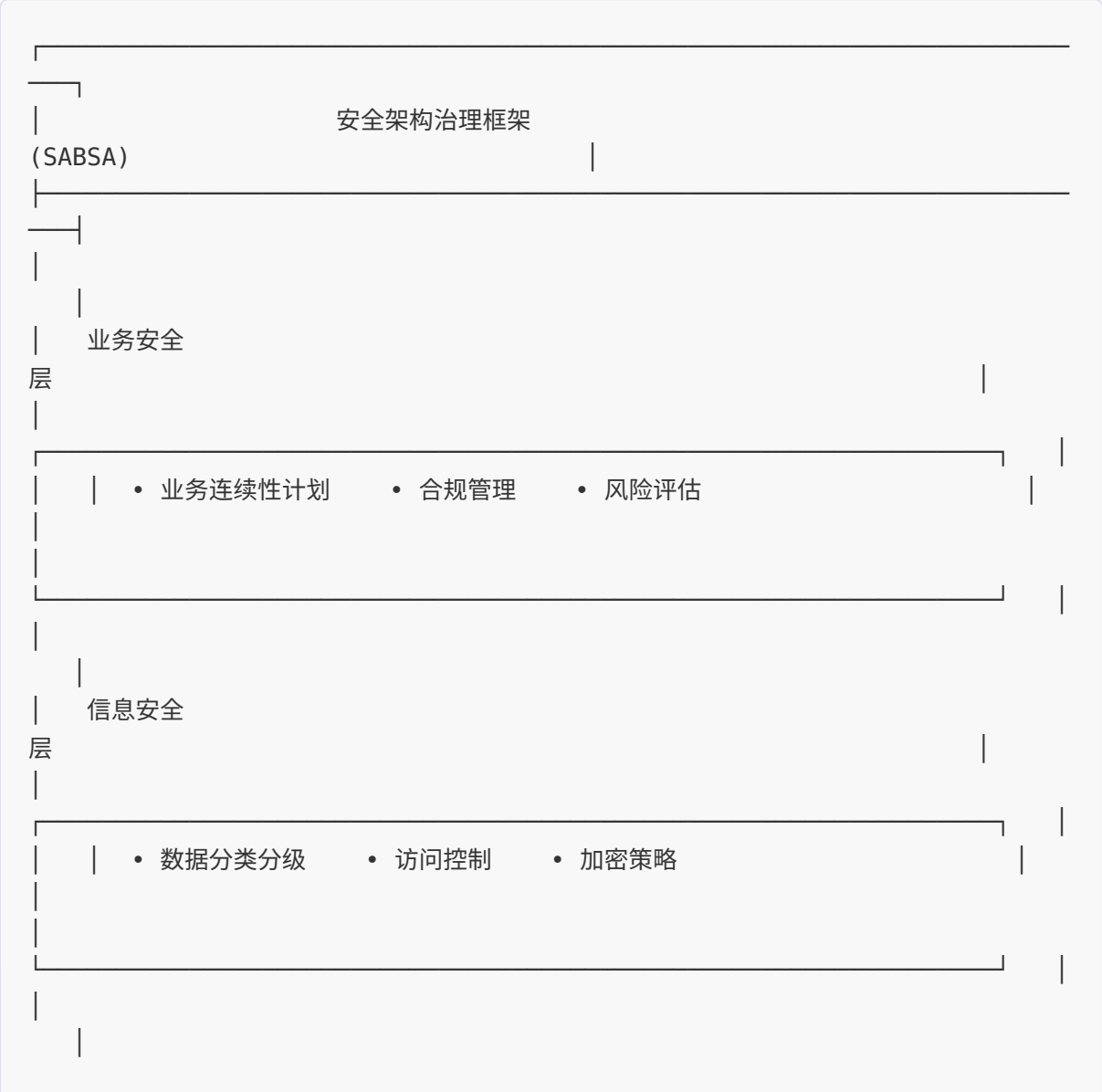
30.1 技术债务评估模型

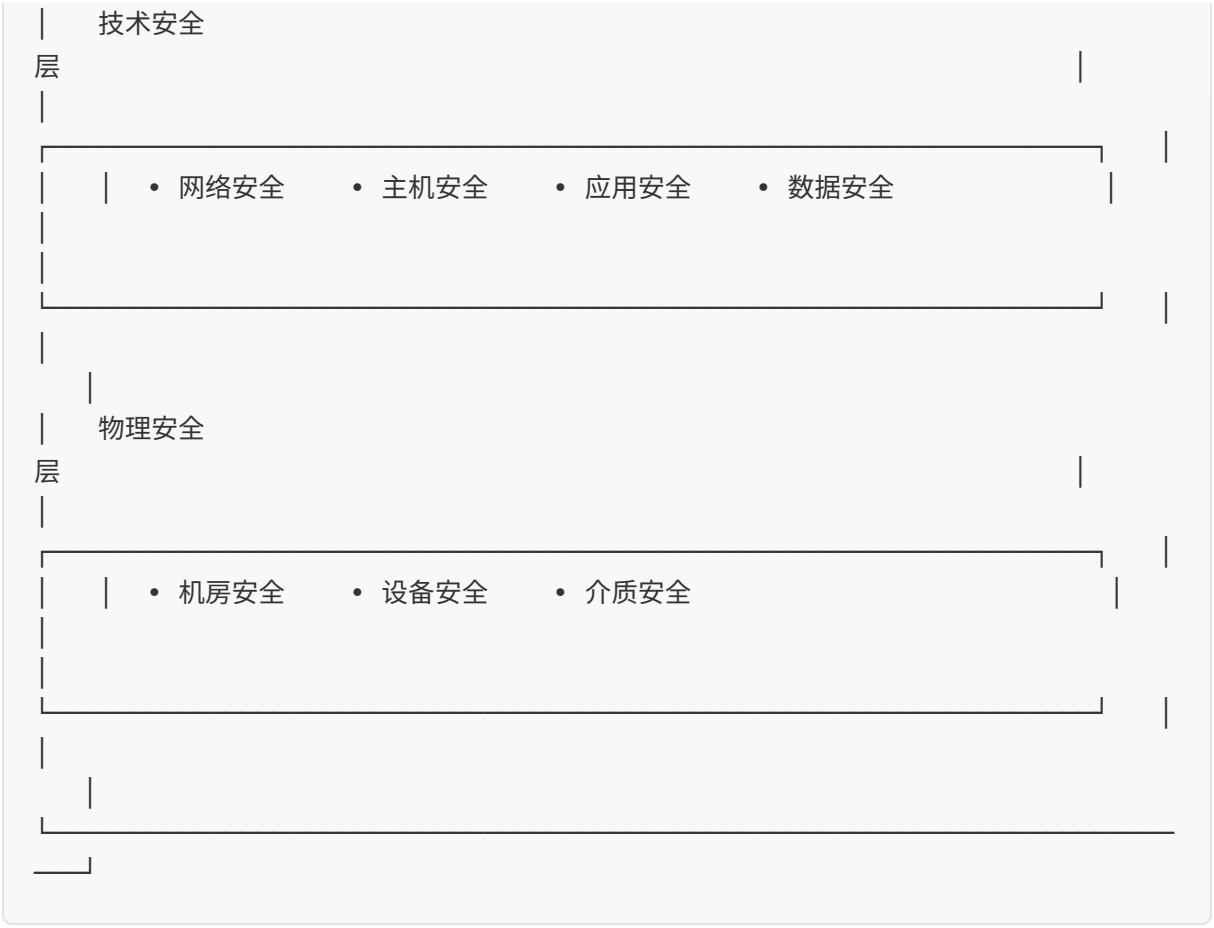
债务类型	描述	评估标准	处理策略
代码债务	代码质量差	圈复杂度、重复率	重构、Code Review
架构债务	架构设计缺陷	耦合度、扩展性	架构调整
测试债务	测试覆盖不足	覆盖率<80%	补充测试

债务类型	描述	评估标准	处理策略
文档债务	文档缺失或过时	文档完整度	文档补全
基础设施债务	基础设施落后	EOL组件比例	升级替换
安全债务	安全漏洞	漏洞数量、等级	安全加固

第31章 安全与合规治理

31.1 安全治理框架





31.2 等保2.0合规要求

安全层面	控制点	技术要求	管理要求
安全物理环境	物理位置、访问控制、防盗窃	门禁、监控、UPS	巡检制度、应急预案
安全通信网络	网络架构、通信传输	冗余设计、加密传输	网络拓扑管理
安全区域边界	边界防护、访问控制	防火墙、IDS/IPS	变更管理、日志审计
安全计算环境	身份鉴别、访问控制	多因素认证、EDR	账号管理、介质管理
安全管理中心	系统管理、审计管理	SIEM、堡垒机	审计策略、权限分离

第32章 数据治理

32.1 数据治理框架

治理领域	关键活动	工具/方法
元数据管理	数据字典、数据血缘	Apache Atlas
数据质量	质量规则、监控告警	Apache Griffin
数据安全	分类分级、访问控制	Apache Ranger
数据标准	标准制定、标准执行	数据标准平台
主数据管理	主数据识别、统一管理	MDM平台

第五部分：工具篇 - 模板、检查清单与工具

第33章 架构设计模板

33.1 系统架构设计文档模板

```
# [系统名称] 架构设计文档

## 1. 文档信息
| 属性 | 内容 |
|------|------|
| 文档版本 | V1.0 |
| 编制日期 | YYYY-MM-DD |
| 编制人 | [姓名] |
| 审核人 | [姓名] |
| 批准人 | [姓名] |

## 2. 业务背景
### 2.1 业务目标
### 2.2 业务范围
### 2.3 业务场景
```

```
## 3. 架构需求
### 3.1 功能需求
### 3.2 非功能需求
### 3.3 约束条件

## 4. 架构设计
### 4.1 总体架构
### 4.2 逻辑架构
### 4.3 数据架构
### 4.4 部署架构
### 4.5 安全架构

## 5. 关键技术决策
### 5.1 技术选型
### 5.2 架构决策记录(ADR)

## 6. 接口设计
### 6.1 外部接口
### 6.2 内部接口

## 7. 风险与应对

## 8. 实施计划
```

第34章 业务分析模板

34.1 业务需求规格说明书模板

```
# [系统名称] 业务需求规格说明书

## 1. 引言
### 1.1 目的
### 1.2 范围
### 1.3 定义和缩略语

## 2. 总体描述
### 2.1 业务背景
### 2.2 用户群体
### 2.3 运行环境
```

```
## 3. 业务需求
### 3.1 功能需求
### 3.2 非功能需求
### 3.3 接口需求
### 3.4 数据需求

## 4. 业务模型
### 4.1 业务流程图
### 4.2 业务实体图
### 4.3 业务规则

## 5. 附录
```

第35章 检查清单汇总

35.1 架构评审检查清单

类别	检查项	检查标准	通过
功能	功能覆盖	100%覆盖需求	☐
性能	响应时间	满足SLA	☐
可用性	高可用	无单点	☐
安全	认证授权	统一认证	☐
可维护	模块化	低耦合	☐

35.2 系统上线检查清单

阶段	检查项	负责人	状态
发布前	代码评审通过	技术负责人	☐
	测试报告	测试负责人	☐
	配置项确认	运维负责人	☐

阶段	检查项	负责人	状态
发布中	发布窗口确认	项目经理	☐
	监控告警检查	运维负责人	☐
发布后	功能验证	测试负责人	☐
	业务验证	业务负责人	☐

第36章 行业基准与参考

36.1 性能基准数据

系统类型	指标	业界优秀	业界平均	业界较差
核心交易	平均响应	<50ms	100ms	>200ms
	P99响应	<200ms	500ms	>1s
	日交易量	>1亿笔	1000万笔	<100万笔
网银系统	页面加载	<1s	3s	>5s
	并发用户	>10万	5万	<1万
风控系统	规则响应	<10ms	50ms	>100ms

36.2 分布式数据库厂商对比

维度	TiDB	OceanBase	GoldenDB	TDSQL
厂商	PingCAP	蚂蚁集团	中兴	腾讯云
开源	是	部分	否	否
金融案例	多	多	较多	较多

维度	TiDB	OceanBase	GoldenDB	TDSQL
国产化	是	是	是	是
价格	中	高	高	中

附录：参考资料与标准

附录A: 参考资料

A.1 架构相关标准

- TOGAF 10 - The Open Group Architecture Framework
- ArchiMate 3.2 Specification
- ISO/IEC/IEEE 42010 - 系统和软件工程架构描述

A.2 业务分析相关标准

- BABOK v3 - Business Analysis Body of Knowledge
- BPMN 2.0 Specification
- DMN 1.3 - Decision Model and Notation

A.3 金融行业标准

- JR/T 0071-2020 金融行业网络安全等级保护实施指引
- JR/T 0171-2020 个人金融信息保护技术规范
- JR/T 0197-2020 金融数据安全 数据安全分级指南

附录B: 术语表

术语	英文全称	定义
ADR	Architecture Decision Record	架构决策记录

术语	英文全称	定义
DDD	Domain-Driven Design	领域驱动设计
TCC	Try-Confirm-Cancel	分布式事务模式
SLA	Service Level Agreement	服务等级协议
RTO	Recovery Time Objective	恢复时间目标
RPO	Recovery Point Objective	恢复点目标

附录C: 常用工具推荐

C.1 架构设计工具

工具名称	用途	推荐度
draw.io	架构图绘制	★★★★★
PlantUML	文本化UML	★★★★☆
Structurizr	C4模型	★★★★☆

C.2 业务分析工具

工具名称	用途	推荐度
Visio	流程图绘制	★★★★☆
Axure	原型设计	★★★★★
Jira	需求管理	★★★★★

文档总结

核心内容覆盖

部分	核心内容	章节数
架构师篇	架构方法论、技术决策、架构演进、韧性设计	12章
业务分析师篇	业务分析、流程建模、价值分析、领域分析	11章
融合篇	业务-架构映射、协作模式、实战案例	4章
治理篇	IT治理、架构治理、技术债务、安全合规	5章
工具篇	模板、检查清单、行业基准	4章

版本历史

- v3.0 (2026-02) - 专业优化版，重构文档结构，强化架构师与业务分析师双视角
- v2.0 (2024-02) - 专业增强版
- v1.0 (2023-06) - 初始版本

本文档由 Kimi Code CLI 生成优化版
生成日期: 2026-02-07