分析设计说明书写作指引

XX分析设计说明书

文档历史

修订日期	修订内容	修订版本	修订人

（要点：规范标题、列清信息、变更完善）

摘要

（要点：以终为始看问题，说清楚背景、需求提出方、解决什么业务问题或者解决什么痛点问题，可以从5W1H视角来看，即what、why、who、how、when、where）

编写目的

概述本文档编写目的，如：此分析设计说明书是对渠道介入提供技术设计方案，功能分配，模块划分，系统总体结构、运行设计以及出错设计等方面做了全面的概括性说明，为后期开发、测试人员进行编码、功能测试提供指导和帮助。

项目背景

概述本文档相关的项目背景，例如时间，项目起因，预期的业务结果等。

任务概述

概述本文档项目相关任务，如：完成物流费用对账，主要包含账单倒入，费用批次对账，差错处理等。

（要点：从产品需求出发，明确要做成什么样子）

简要说明此设计实现的产品需求，功能如

- 支持xxx的增加
- 支持xxx的获取

目标人员

指明文档阅读人员。包括架构师、产品经理、开发人员、测试人员、运维等。

规范与约定

概述本文档相关的规范与约定，如：

package结构：如model包、manager包等；

环境配置、其他常量约定等；

术语与缩略语

描述与本文档相关的业务或技术上的术语，如：

缩略语/术语	全称	说明
FM	First Mile	从发件端到中转仓之间的物流服务商类型

参考资料

列出参考资料名称以及出处。

系统分析设计

系统设计目标

描述系统设计的目标，比如*pv/uv*，*tps*，容量，预估数据量，并发等，系统分析设计将以此作为目标展开。

（要点：从容量、稳定性以及安全性角度出发，明确目标）

容量评估

说明此设计的容量评估目标，主要包括

- 系统容量
- 性能要求，如关键接口qps、RT等

稳定性需求（可选）

简要说明此设计的稳定性需求，可以包括但不限于

- 可靠性要求
- 可用性保障
- 数据一致性等

安全性需求（可选）

- 输入输出的验证与转义
- 身份验证与密码管理
- 会话管理
- 错误处理和日志
- 通讯安全，数据安全
- 文件管理

总体架构分析

描述项目涉及的系统以及应用的总体架构分析，比如新的应用架构，新的系统，全新的系统交互以及通信方式等，并且说明其设计的背景原因和必要性，如评估渠道流量压力，设计合适的系统架构。

（要点：合理运用视觉元素表达架构设计，如C4模型，能够清晰梳理出不同方案选择过程和决策的考量；系统架构图以模块作为最小交互单位，层次需要清晰，箭头可以被理解为数据流或依赖关系）

优秀范例：

失败范例：

调用链路分析

新增或改动现有接口调用时，需要重点分析调用场景和链路，及时发现性能隐患

领域模型分析

不同于数据库表结构，领域模型适用于业务层，是对数据库数据在业务上的加工，比如商品描述和商品基本信息是属于一个领域模型，但在数据库中是两个表，因为商品描述通常是*text*类型，数据量比较大，和主表在一张表会拖慢主表的查询。请使用*uml*图中的类图来描述领域模型。

（要点：结合实际场景合理抽象业务模型，分析模型间关系，如果采用DDD可以在此章节对实体、值对象、领域服务进行描述）

优秀范例：

状态机

描述领域模型的状态转换和生命周期，以及状态转换的条件。从满足业务的角度出发，论证状态存在和转换的必要性。请使用*uml*图中的*statemachine*图来进行描述。

用例分析

用例分析分别从业务和系统维度描述业务以及系统实现，方便项目成员以及团队其他成员一目了然的了解项目的整体业务，请使用*uml*工具用例图来进行描述。

（要点：从业务、系统等不同角度给出用例来描述交互，可以融合在一副图中，但对用例间的关联、包含、扩展及泛化关系要描述准确，否则会失去用例图的意义）

优秀范例：

业务用例图

业务用例图提供全局业务视图，是对项目整体的任务分解。需求经过分析后，细化到每个任务都是可评估工作量的程度。每个用例都应该有相应的操作者*(actor)*。一个完整的业务用例图，能够让项目成员更容易的天然的可以用于项目任务分解。

系统用例边界

系统用例主要描述全局的业务用例在各个系统间的分布，以明确业务在系统间的流转，以及确认系统边界。

核心业务规则

核心业务规则用于描述项目涉及的业务流程以及约束，说明在技术实现上的设计细节，阐述设计的合理性。由于规则和模型之间本身存在业务以及技术上的关联性，有可能在领域模型中已经涉及到了一些核心业务规则的设计说明，但重复描述不会降低文档的可理解性。请使用*uml*图中的流程图辅助描述。

业务规则1

（要点：通过时序图和流程图结合的方式对核心功能从设计细节上做进一步分析，此过程中依然需要体现对方案细节的推敲对比）

优秀范例：

失败范例：

现有系统影响分析

列出本项目涉及的对现有哪些系统、哪些业务的影响分析。需要涉及配合改造或者部署有变更的才是有影响，比如依赖的package升级或者go第三方库不兼容，对接的地方发需要外围系统配合改造，或者某个服务实现增加了判断，结果码有增加，需要外围系统配置文案，或者变更业务判断等。

系统1

（要点：清晰列出受影响的系统，评估可能存在的兼容风险和发布依赖）

部署架构（可选）

（要点：图中应准确阐述系统分布、交互关系、资源使用情况和容量规划，配合文字说明决策依据；一般以服务器或者集群或者机房为交互最小单位，展现出接入层、业务层和存储层的实例配置信息）

优秀范例：

网络拓扑图

说明项目设计的应用系统、中间件、*db*以及网络设备的部署方案，并且描述其合理性。

资源规划

根据性能和容量目标描述资源计划和列表。

（要点：根据产品需求、未来电商活动预估ccu或order数据评估系统容量，包含CPU、Mem、network、disk、cache容量、db容量等；分析清楚应用属于ccu敏感还是order敏感）

配置&配置规约

描述系统需要的配置，以及配置方面的约定，如不涉及则可忽略。

部署目录结构

明确应用部署目录结构，如无新增或者变更，可忽略。

API设计

中间件设计（可选）

数据库

注意此处不是表结构设计，而是对采用何种数据库、分库分表的选择、数据增长、事务使用及对应的高可用分析

缓存

消息队列

对象存储

（要点：阐述明确一目了然，按需包含数据库、缓存、消息队列、对象存储的设计分析，不能仅仅停留在库表设计分析或结构变更上，要有对事务使用的分析、高可用设计、过载保护等；对于缓存，如果一致性要求较高，需要考虑数据更新策略及不一致带来的影响）

非功能性特性设计

优秀范例：LLS点线履约系统概要分析设计说明书中非功能特性设计章节

可靠性

描述系统在可靠性上的分析和设计，系统在容错性上是如何处理的，故障恢复、服务降级、熔断等的处理机制，以及在数据可靠性方面的考虑等。

（要点：遵循简单、冗余、标准、健壮的设计原则，消灭单点，做好数据一致性，注意对热点、极限值的处理；抛开系统自身的性能瓶颈，系统的不稳定往往由外部依赖带来，因此可靠性分析可以从依赖出发，做到控制依赖、弱化依赖，说清楚当外部故障发生时的应急预案，如何容错，如何熔断，如何服务降级；对于外部依赖，需要主动提出依赖的SLA）

可扩展性

可扩展性涵盖了系统的编译期和运行期的设计，运行期如何具备水平扩展能力来提升系统性能，编译期如何设计良好的代码结构来适应业务在未来可预见时间内的变化。这里需要描述系统在可扩展性上的设计思路。

（要点：包含编译和运行两个方面，编译期可以从设计模式出发，结合目录结构进行扩展性说明，这一部分可以和可维护性有重叠，不要简单理解为应用部署水平扩展能力）

失败范例：

可运维

描述在提升系统运维方面的分析及设计，好的系统应该是尽可能自动化的完成业务，自身应该具备相当能力的容错处理，不需要运维人员介入处理；同时也尽可能是标准化的，方便运维人员部署，或者提供可视化的运维操作。

（要点：重点阐述一个鲁棒性服务是如何进行容错处理的）

失败范例：

安全性

配合安全要求，阐述系统安全方面（例如*XSS*，*SQL*注入，*DDOS*，数据安全等）的设计。

（要点：对潜在的web攻击是如何预防的，数据安全如何保证，特别是对外系统，这里需要重点分析）

失败范例：

可测试性

描述系统各个业务在可测试性上的分析，发布应该满足灰度要求，系统应该具备在生产环境的可测试性。必须等到某个时间点、修改操作系统时间，或者直接不可测，都不应该发生。

可维护性

可维护性包含可扩展性在编译期的设计，可维护性的系统必然具备可扩展性，此外，可读性、命名规范、包路径约定等也是可维护性的考量。

监控

描述需要监控的业务功能点，浮动/绝对值等。

（要点：必须明确需求涉及的核心业务指标，围绕其制定监控和告警策略；监控内容清晰列出，相应的告警阈值需要说明设置缘由）

失败范例：

其他

描述其他关注点的分析，例如灰度发布，回滚方案，数据迁移方案等对系统设计的影响，系统设计如何处理才更好的支持。

（要点：日志规范要提前确定；设计过程中要考虑系统的向前和向后兼容性，列出可能存在的问题和解决方案；上线计划要从发布依赖、灰度能力、配置变更、数据迁移等多角度结合实际进行分析）

灰度发布

这里需要描述的是系统如何设计才能满足灰度发布原则，而不是描述如何进行灰度方案。

发布回滚

这里需要描述的是系统如何设计才能更好的进行发布回滚，而不是描述回滚方案，设计合理可以确保回滚更易于处理。

数据迁移

和发布回滚类似，这里也是需要描述系统如何设计才能更好的进行数据迁移，而不是说明迁移方案。