系统优化 - 背景与思路

支撑业务目标

对抗流量暴增

交易系统优化

常规思路：

不足

- 不系统，不全面
- 被动
- 难以复制

本质

- 目标不明确
- 目标不一致

思路

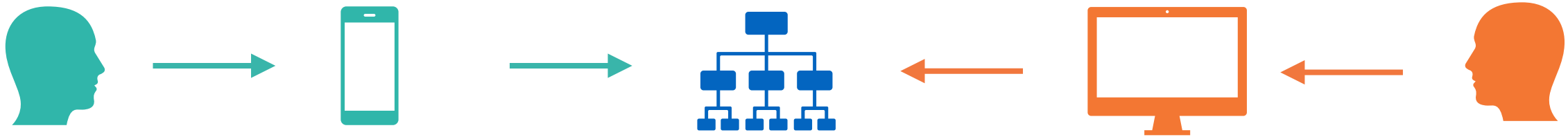
1. 确定目标抓手
2. 目标导向
3. 抓手考查

系统优化 - 确定目标与抓手

常规系统优化关注点：高性能、一致性、高可用、可扩展性、可维护性 ……

需要关注哪些？怎么分类？

思路：回归本源，抽象归纳



用户
（用户、运营、RD等）

关注点：可用

关注点：可维护

RD

可用的标准：高性能（够快）、高一致（够对）

可维护标准：快速发现、解决问题（存量）
快速支撑业务扩展（增量）

优先级P0：容易定量，面向用户，不容有失

优先级P1：定性，锦上添花

结论：以“可用性”作为抓手，优化同时兼顾性能、一致性、可维护性与可扩展性

系统优化 - 目标拆解细化

思路1：细化因子，找到关注点

已知： 可用性 = 无故障时间 / (无故障时间 + 故障维护时间)

故障次数 * (**定位时间** + **修复时间**)

运行时间 * **故障率**

关注点：降低故障率 & 缩短发现时间 & 缩短修复时间

系统优化 - 目标拆解细化

思路2：横向对比，拓展思考维度

BG	方向	关注点与方法
外卖订单系统	系统演进 & 优化方案	高性能：异步、并行、缓存 一致性：异步、并行、缓存 高可用：限流、降级、资源隔离 可扩展性：分库分表 可运维性：**事前**监控，**事中**SOP，**事后**总结
点评账号系统	高可用建设	**快速发现故障**：系统&业务监控 **延长不出故障**：资源隔离（缩小影响范围），核心服务降级，冗余（异地多活、主从、分布式等）
外卖客户端	高可用建设	**快速发现问题**：业务&系统监控 **快速定位问题**：核心链路日志体系 **快速解决问题**：容灾降级备份限流
酒旅	服务容错 Design for failure	穷举容错方法：重试，限流，熔断，资源隔离等

结论：选用上述关注点 X 时间维度 进行细化

系统优化 - 目标拆解细化

	事前					事中	事后
	设计阶段	开发阶段	测试阶段	上线阶段	运维阶段	事中	事后
降低故障率	**领域设计评审** **难点方案评审**	**代码规范** **开发脚手架**	**单元测试**	**上线规范**	预防问题：**冗余**（异地多活，主从，分布式）、**资源隔离**、**缩短核心链路**（异步） 发现隐患：**吞吐量化**（压测）、**故障演练**	**SOP**	**总结复盘**
缩短发现时间	**系统核心能力指标&监控设计** **业务指标与监控设计**	**日志规范**			**系统&业务监控配置** **核心链路日志体系**		
缩短修复时间	**系统相关开关设计** **业务操作后门设计**	**开关、后门开发**			**容灾降级**：熔断、限流、重试、补偿、回滚 **一键式无脑解决方案**		

问题：虽然全面，但是太细，不容易执行，如何考核?

系统优化 - 执行与考核

思路：相似方法收敛为事项，排期执行
制定制度抓手，考查执行效果

目标（可用性建设）	兼顾项	事项	抓手
设计正确、合理 避免频繁开发、重构 从而降低故障率	高性能：读100，写200 高一致：核心数据（钱、订单状态等）强一致、非核心最终一致 高可扩展：可以通过横向扩展支持业务发展	**制定设计文档规范**（包括领域、系统、架构、业务监控设计等方面）	设计评审制度
代码简洁优雅 技术工具成熟 日志准确完善 降低故障率、缩短发现问题时间		**制定开发规范**（包括日志规范、代码规范、分支规范）	CodeReview制度
单测尽可能覆盖 简化回归 降低故障率		**制定测试规范**（单测用例编写、覆盖等）	提测制度（可与QA共同制定）
缕清上线依赖及顺序 提供回滚方案 降低故障率		**制定上线规范**	上线制度（文档、review、周知等）
降低问题影响范围 降低故障率		**预防问题策略建设**（冗余、隔离、缩短核心链路等）	系统巡检 故障演练

系统优化 - 执行与考核

第一时间发现问题 缩短发现时间	**系统&业务监控建设**	
问题自解决 缩短修复时间	**容灾策略建设**（熔断、限流、重试等）	系统巡检 压测、故障演练
规范化处理问题流程、梳理隐患 有助于提高可用性	**事中SOP流程规范**	wiki记录，定期评审思考复盘
简单化处理问题流程 有助于提高可用性	**无脑解决方案建设**（全局入口开关等）	系统巡检 压测、故障演练
做有积累的事 有助于提高可用性	**复盘总结制度**	caseStudy & 节假日总结

系统优化 - 成果

定性

能够快速发现问题：推动完善交易系统、业务打点及波动监控

降低故障率：推动完善降级策略、容灾策略建设

吞吐可量化：完成压测

输出方法：组内演绎中

定量

核心接口性能提升：TP99 150ms -> 90ms

核心接口可用性提升：4个9 -> 6个9