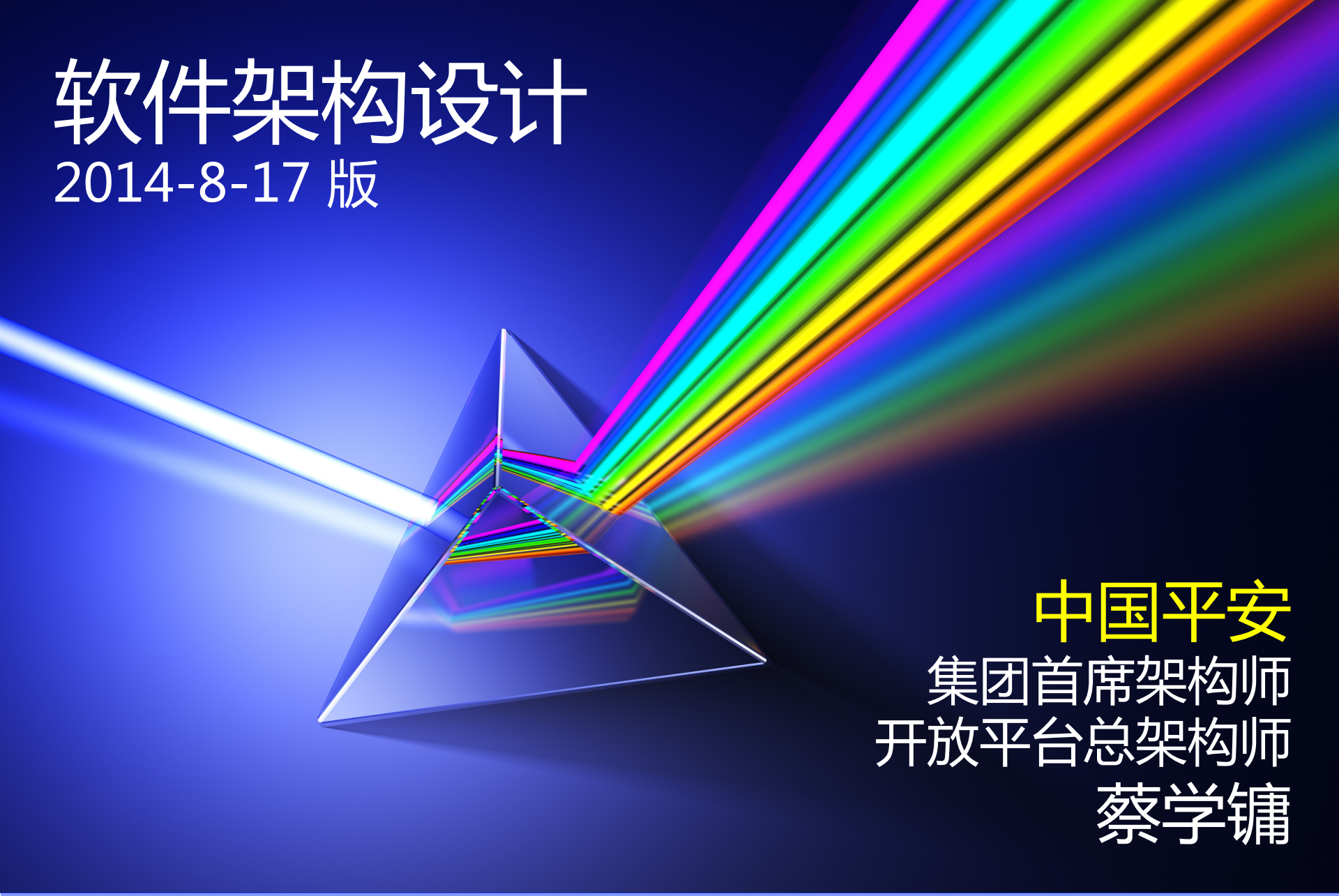
软件架构设计

2014-8-17 版



中国平安

集团首席架构师

开放平台总架构师

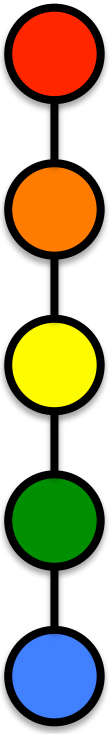
蔡学镛

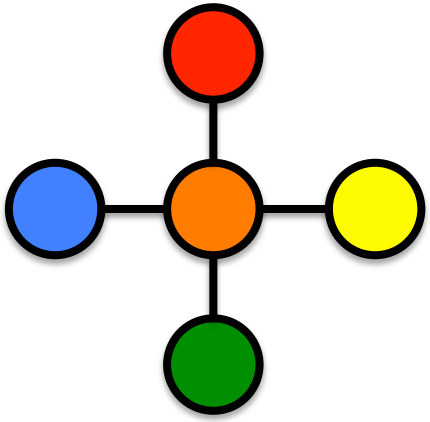
A Methodology of Software Architecture Design

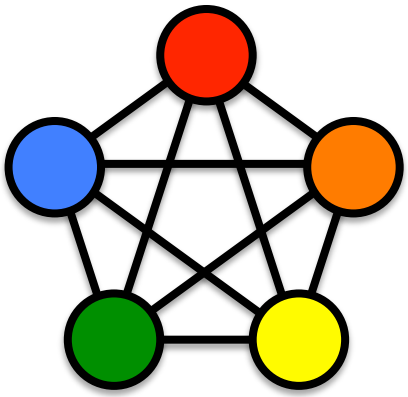
认识软件架构

一句话说明架构是什么

架构就是...代码的组织方式

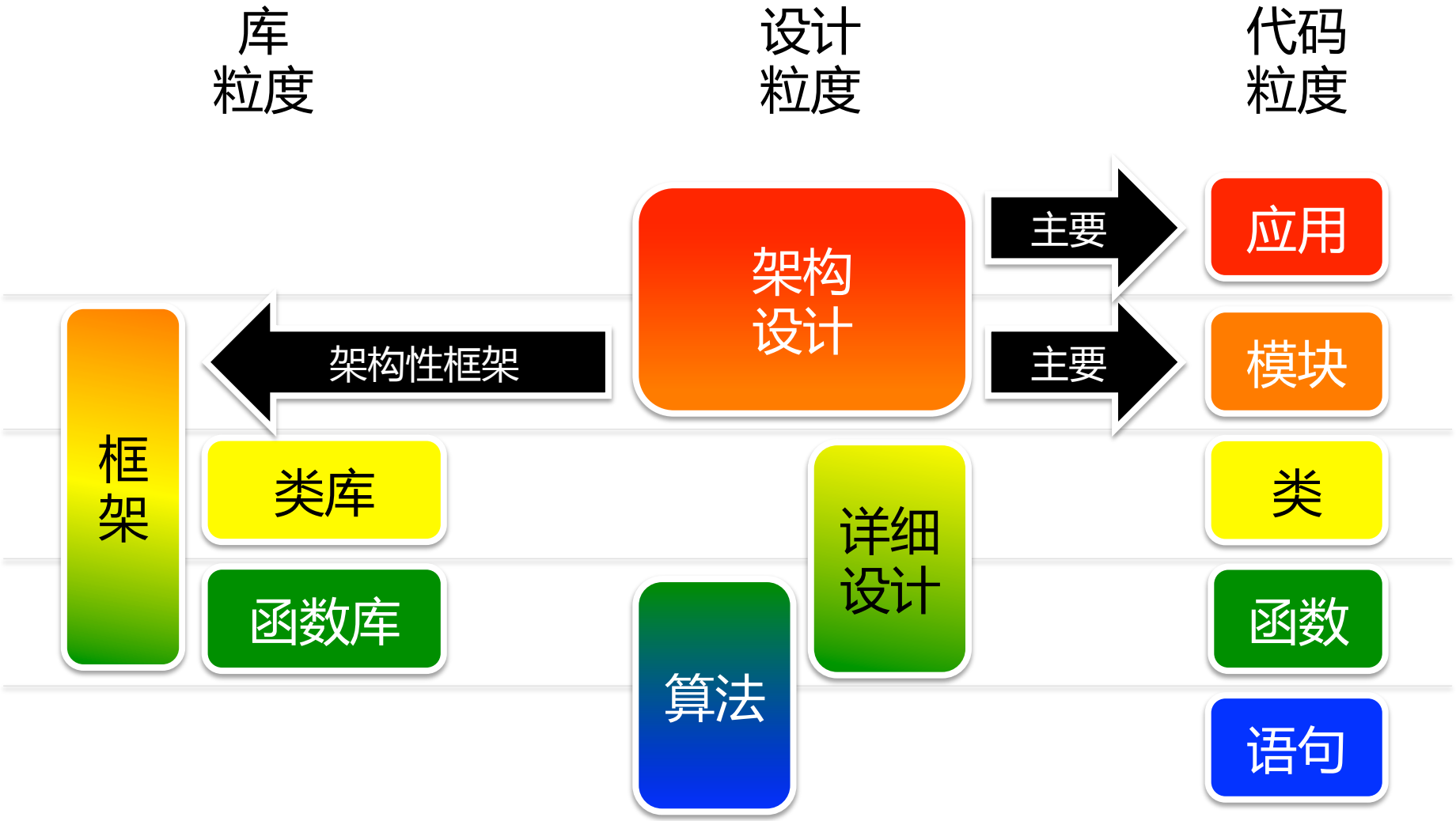






说说这三个架构各自的优缺点

但架构只著眼于大处



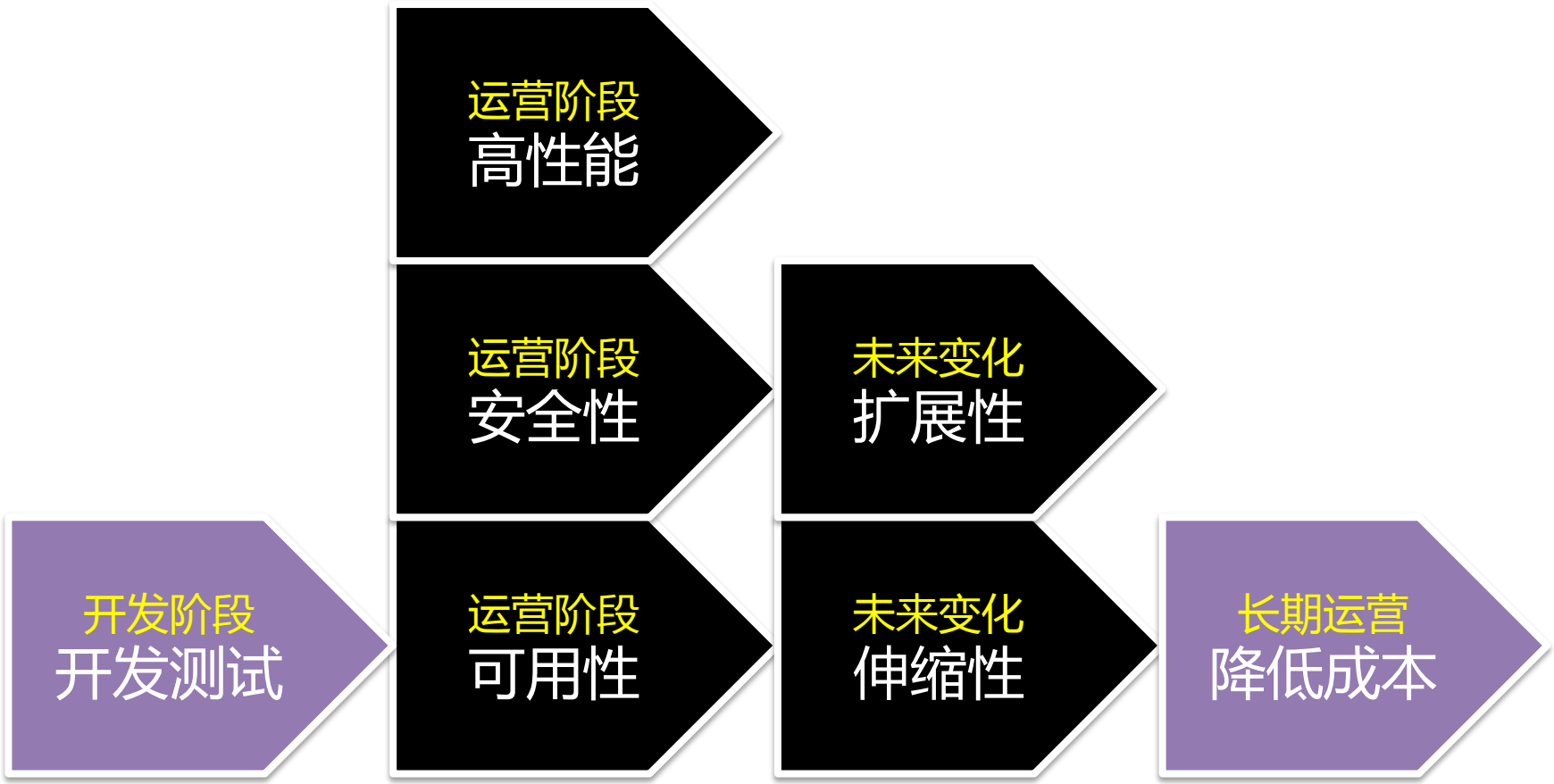
许多人常将架构与设计模式和框架混为一谈，这是错的

$$架构粒度 = f(项目, 阶段, 层)$$

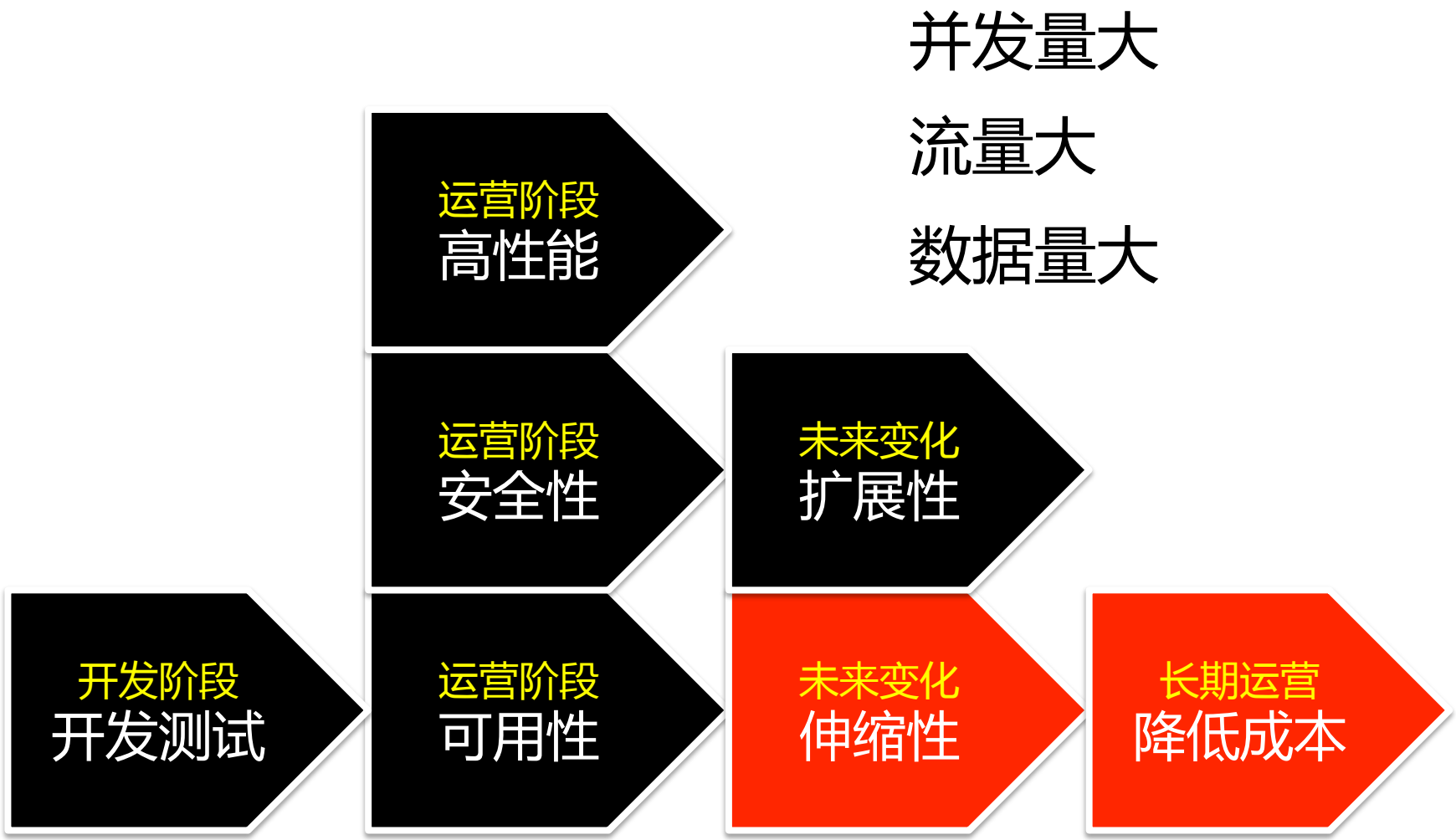
设计师角色


【架构师】
关注大格局的设计需求
架构设计
【系统设计师】
关注小的局部设计需求
详细设计
【算法工程师】
关注具体问题的代码解决方式，效率为主
算法

什么是「大格局」的需求？

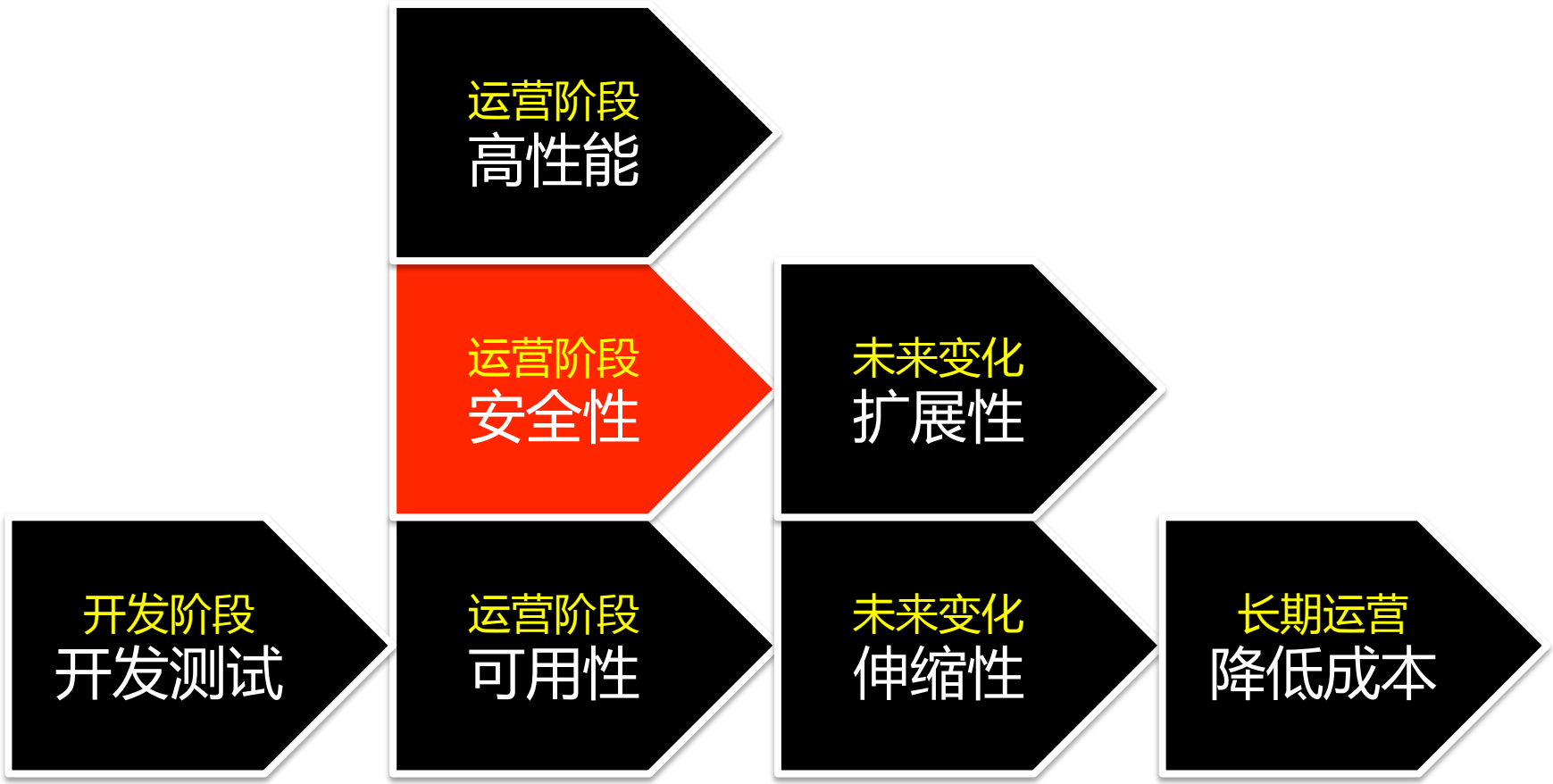

开发阶段
开发测试
运营阶段
高性能
运营阶段
安全性
运营阶段
可用性
未来变化
扩展性
未来变化
伸缩性
长期运营
降低成本

相比于企业级系统，互联网系统的差异



相比于企业级系统，互联网系统的差异

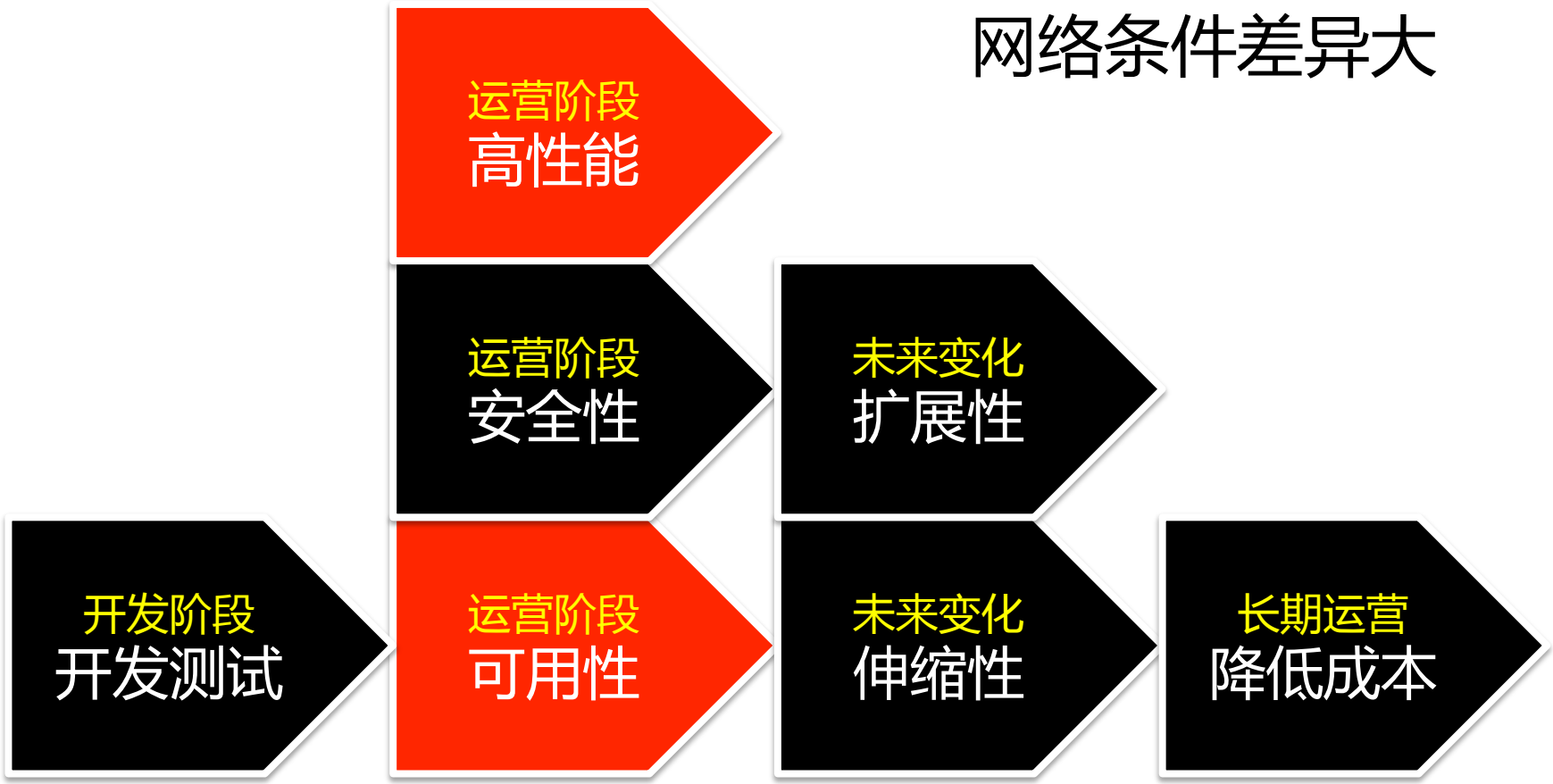
暴险危机高



相比于企业级系统，互联网系统的差异

地理分布广

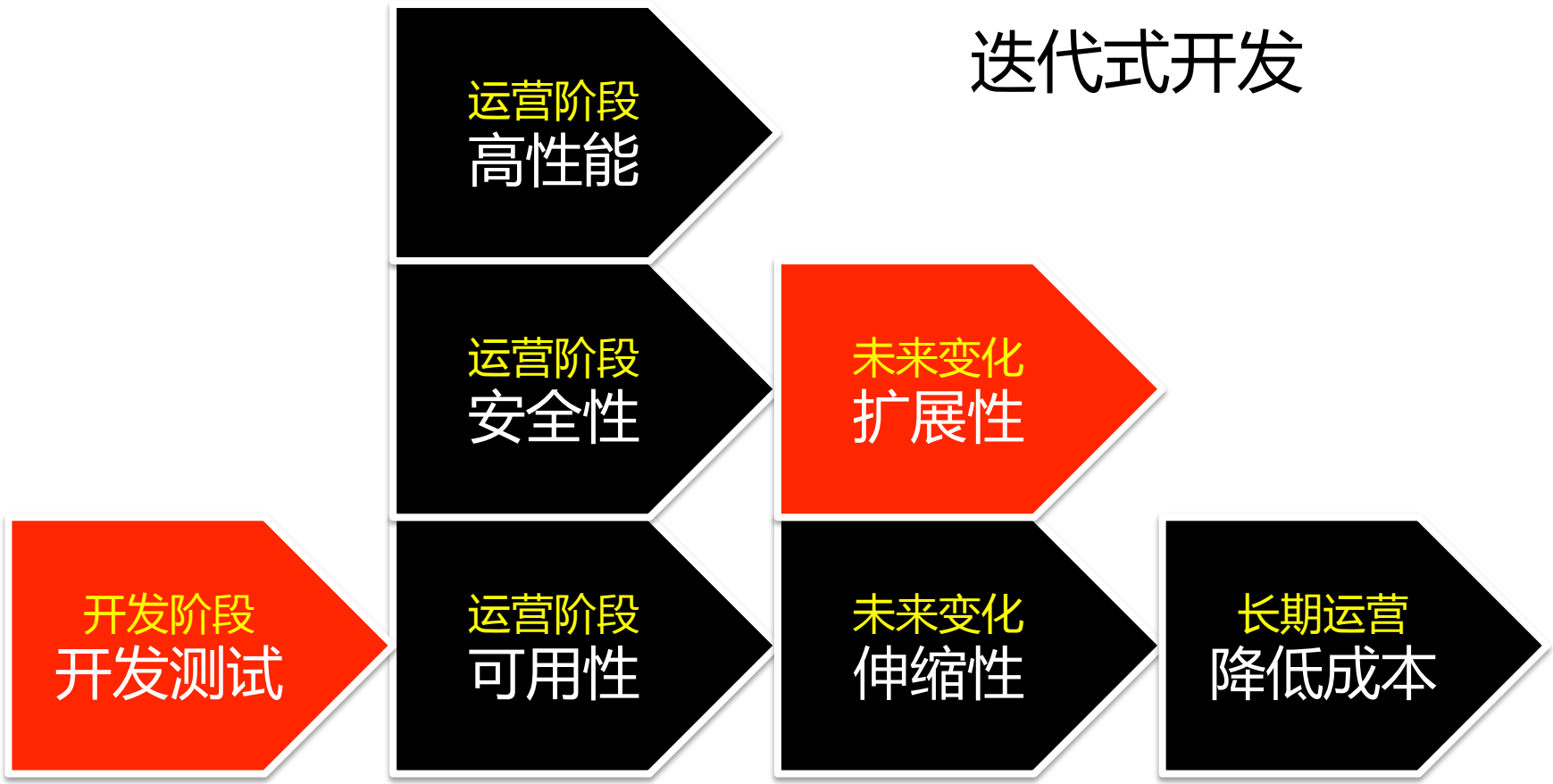
网络条件差异大



相比于企业级系统，互联网系统的差异

需求变化快

迭代式开发



一个软件系统怎么可能如此完美？

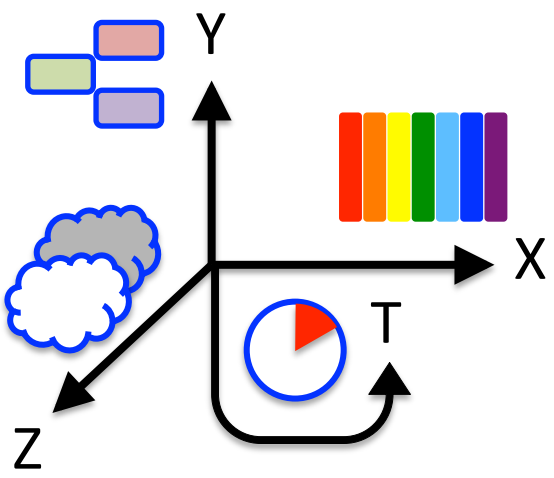
答案是：整体的目标与局部的目标分開

【问题】要如何规划设计模块，并组织这些模块，使其成为「好的架构」，满足大格局的目标？

【答案】第一步是切割出足够细粒度的模块，用正确的方法连结起来。

架构的 4D 座标系统

架构的四维座标系统



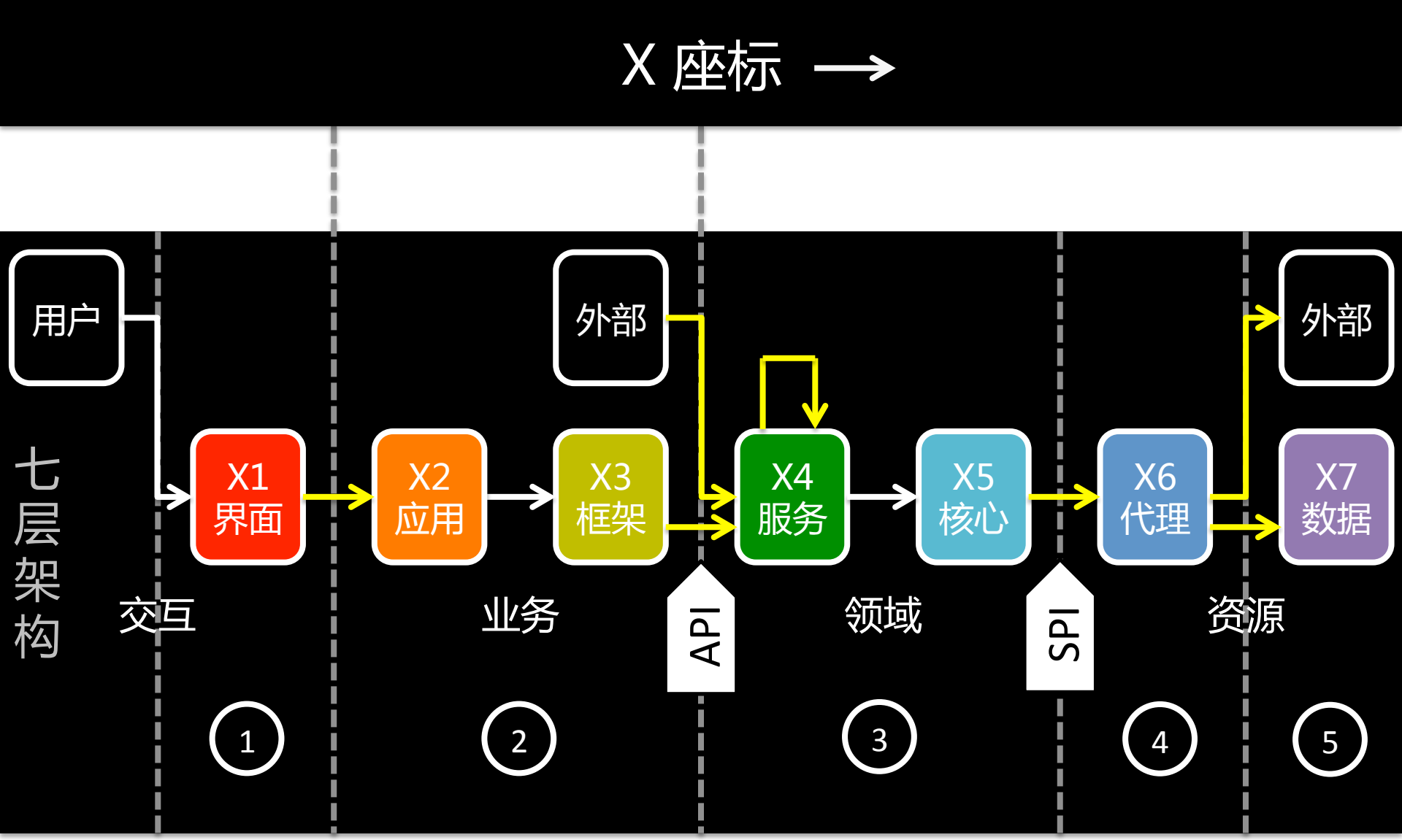
前后端维度（$X_1..X_7$）：界面（红）、应用（橙）、框架（黄）、服务（绿）、核心（蓝）、代理（靛）、数据（紫）

业务维度（$Y_1..Y_n$）：每个业务系统

本教材 不包含 Y Z T 的详细架构方法

系统维度（$Z_1..Z_n$）：软件、容器、运行时、操作系统、虚拟机、到硬件。跟行业无关

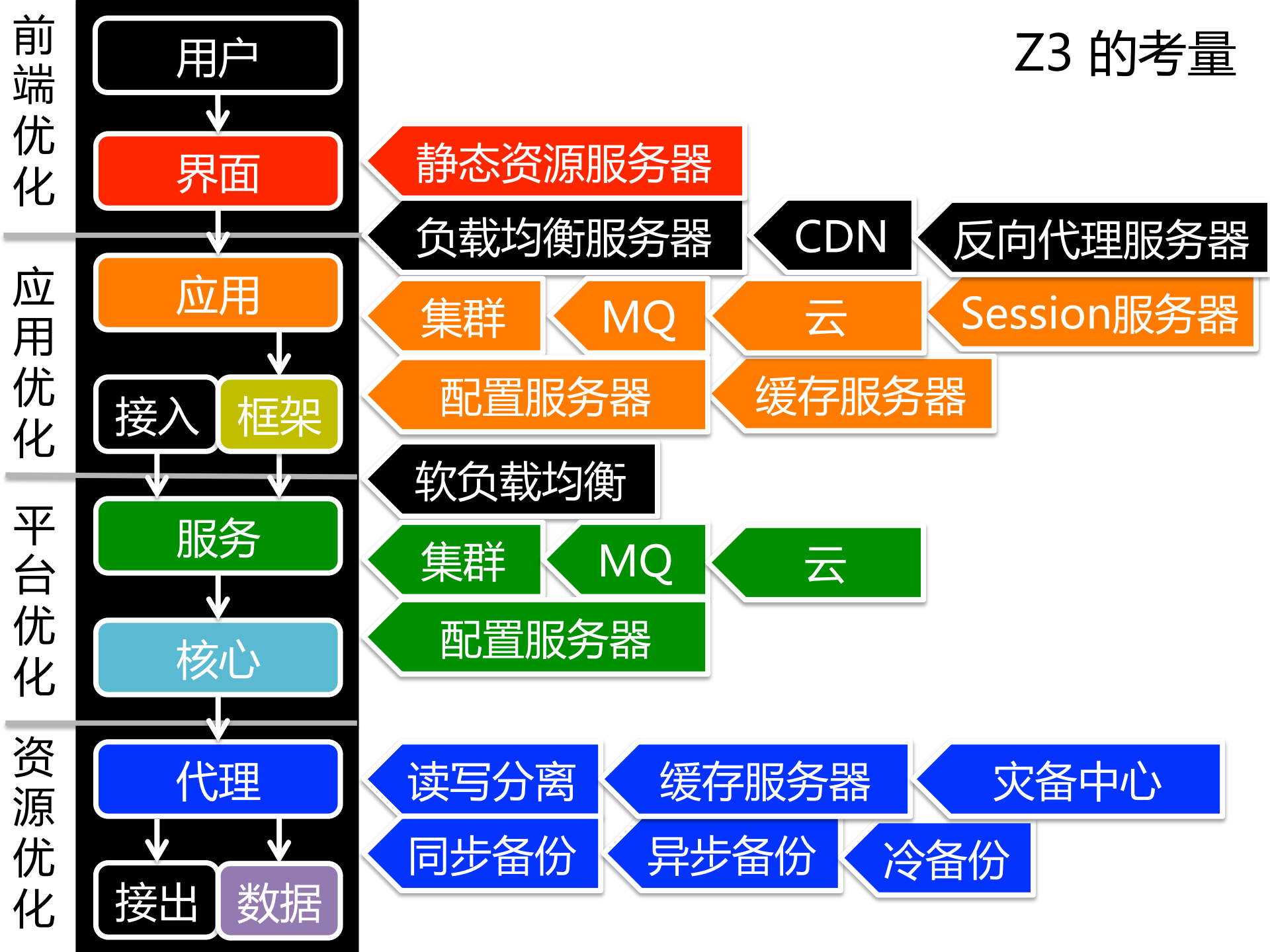
时间维度（$T_1..T_n$）：初始架构到成熟架构



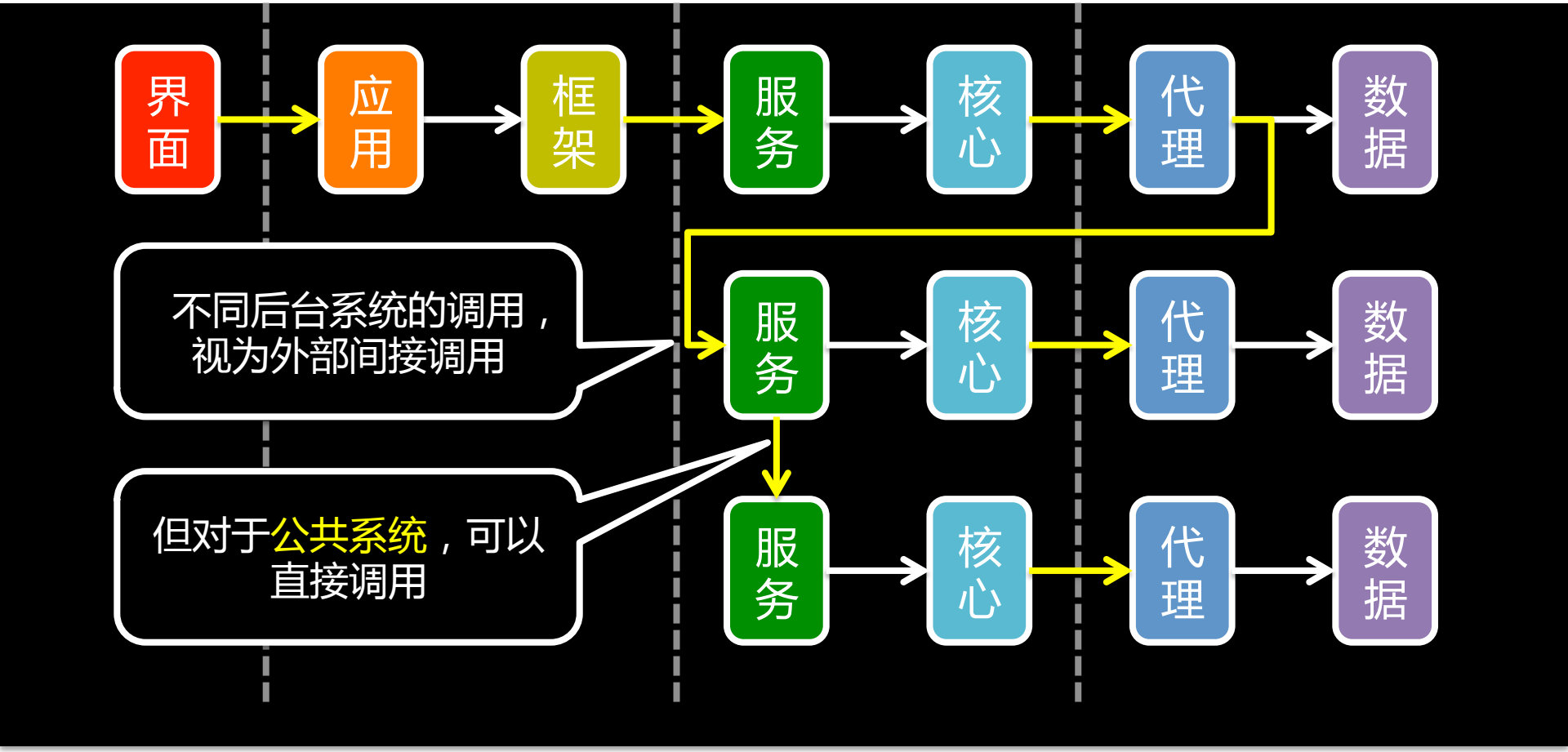
注意：每一层内由多个模块构成，层只是一种逻辑概念，层在架构中不具备实体

黄色箭头是跨系统的调用，白箭头是系统内调用

Z3 的考量

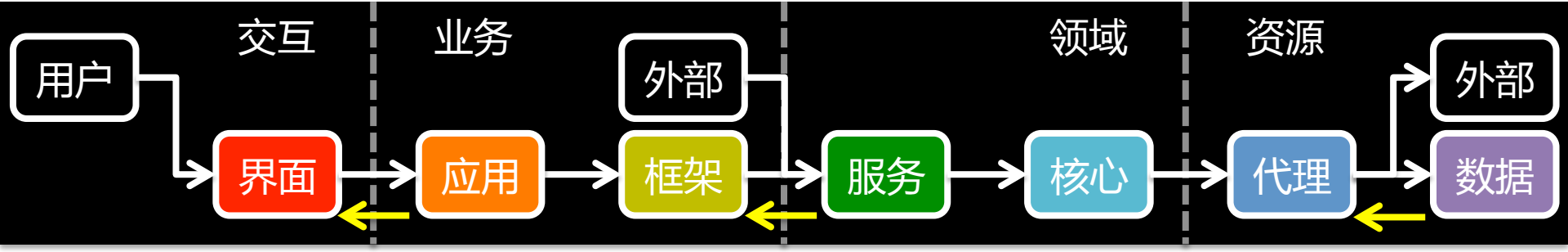

前端优化
应用优化
平台优化
资源优化
用户
界面
应用
接入
框架
服务
核心
代理
接出
数据
静态资源服务器
负载均衡服务器
CDN
反向代理服务器
集群
MQ
云
Session服务器
配置服务器
缓存服务器
软负载均衡
集群
MQ
云
配置服务器
读写分离
缓存服务器
灾备中心
同步备份
异步备份
冷备份

XY 座标



公共系统是指大家都可能需要的系统，包括短信发送、加密服务。公共系统不可以依赖任何非公共系统。公共系统接口简单不易改变。公共系统没有独立成为一家公司运作的可能。

七层架构详解



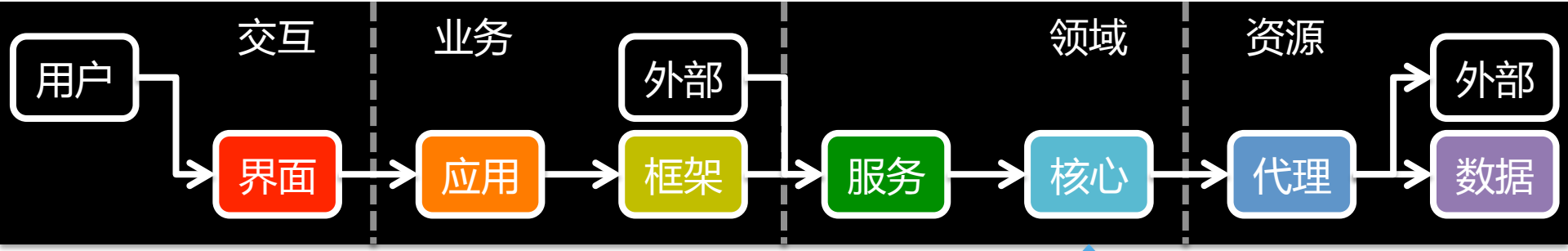
跟外部有接触的，只有三个地方。

三个外部系统，包含一个人，一个接入系统，一个接出系统

箭头指的是接口依赖，不是信息流向

黃色箭頭是回调（Call-Back）。想一想，为什么这三层要允许回调？

黃色箭头部分也可改用 Message Queue 的低耦合設計方式

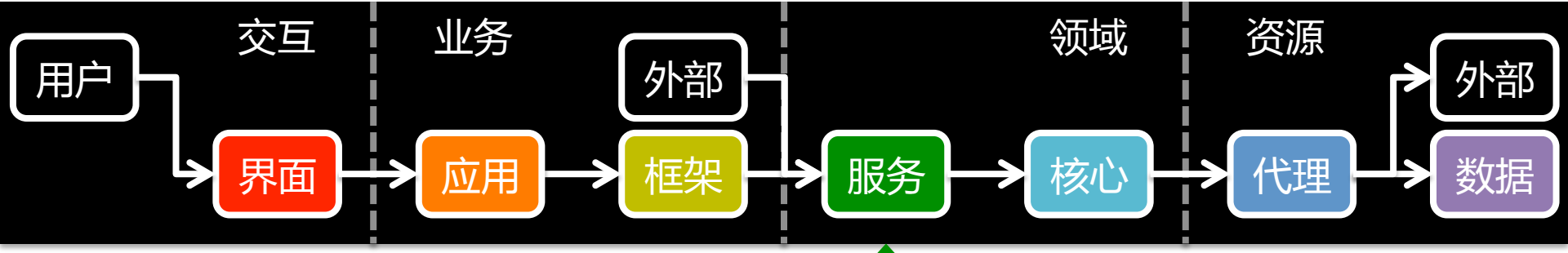


核心层反映出「领域模型」

核心层的接口基本就是对此领域模型进行操作

为何要建立领域模型？

1. 帮助接口设计
2. 帮助数据存储设计，梳理出更具有弹性的存储方式



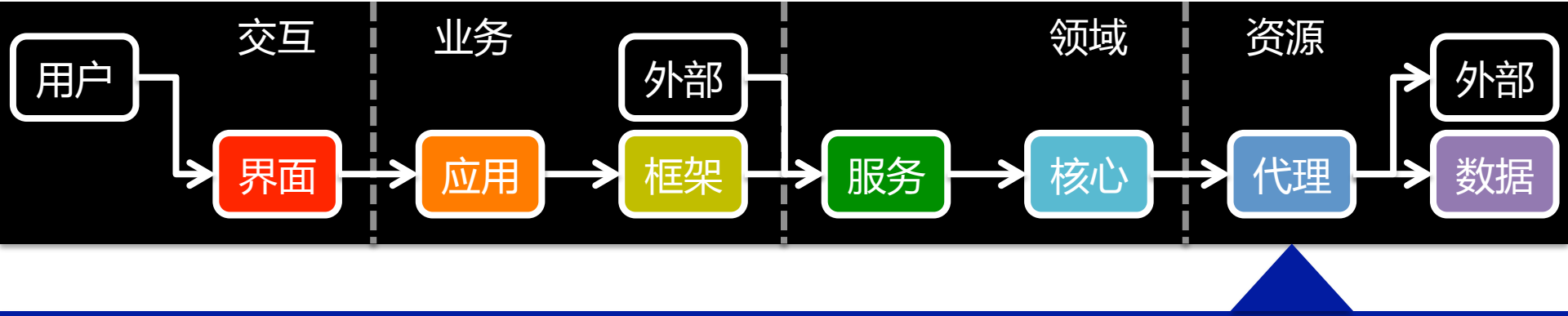
服务层针对「领域对象」进行操作，并提供弹性的调用接口

服务层接口通常数目不多，但每个接口通常参数相当多

服务层没有状态，也不做缓存

实现 API。如果公开，就是开放接口

调用服务层的接口，通常需要授权

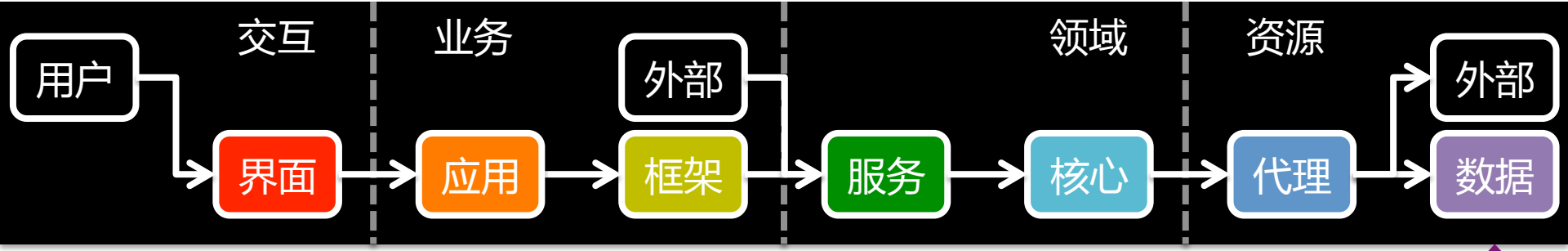


驱动作用：

- 數據代理：代表外部系统或数据库
- Z3 缓存：为了效率或提高可用性（当外部系统掉线）
- Z3 数据模块，支持读写分离
- 转接或转发
  - 转接到外部系统
  - 转发到日志系统，数据备份系统（通过事件钩子）
  - 热备系统接入

SPI 作用：

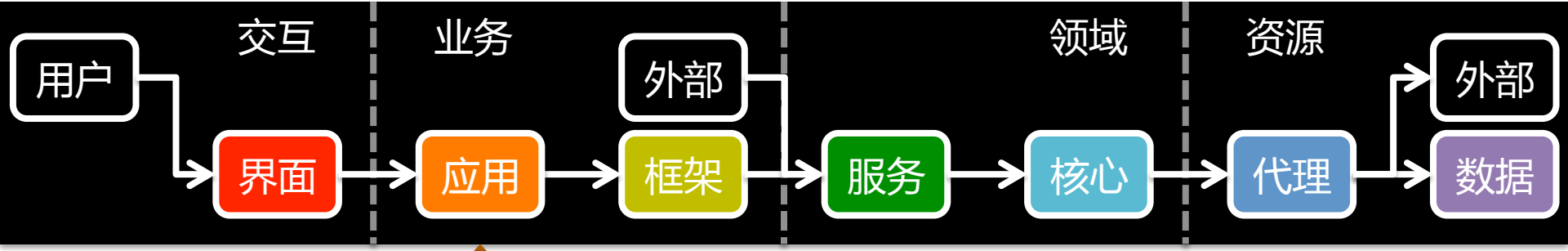
- 隔离：避免依赖特定的外部系统或数据库



数据是公司最重要的资产，数据层负责记录系统运作后的最终结果

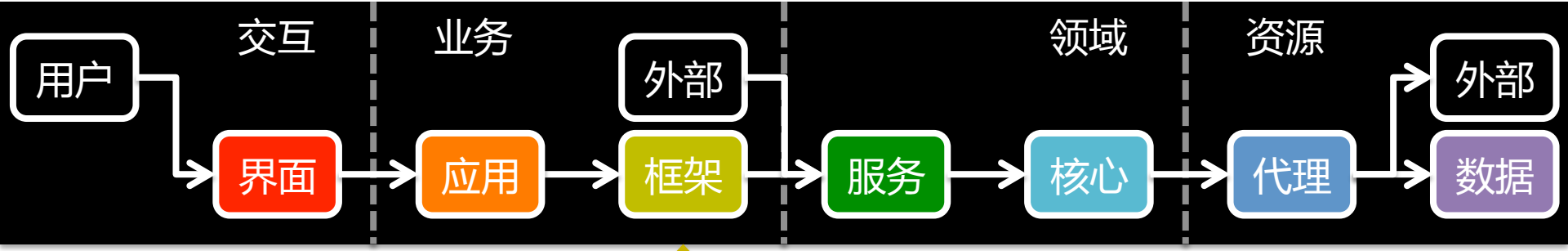
根据数据的特性，数据库可以是：

- 关系式数据库
- 列数据库
- Associative DB
- Key-Value
- 文件数据库
- 日志
- …



根据市场需求，开发各种应用，并以接口的方式展现。

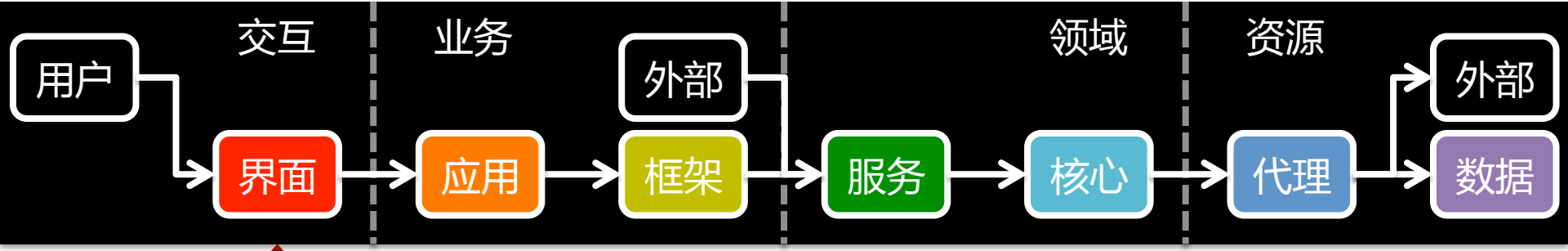
如果是 Web 应用，则这里的 Z3 包含 Web 服务器层



将常用的应用流程设计成框架，后续开发「同类型应用」时，只要通过参数或者 DSL，就可以轻易订制应用，减少开发应用的成本

框架也可以用接口的方式开放让外部调用

Z3 缓存与 Session



界面更像是用户的延伸，而非应用的延伸。界面可被視為用戶代理（User Agent）

根据用户喜好、语言、平台（手机、电脑、平板...）进行开发各种用户界面的开发。

一个应用可以有多个界面

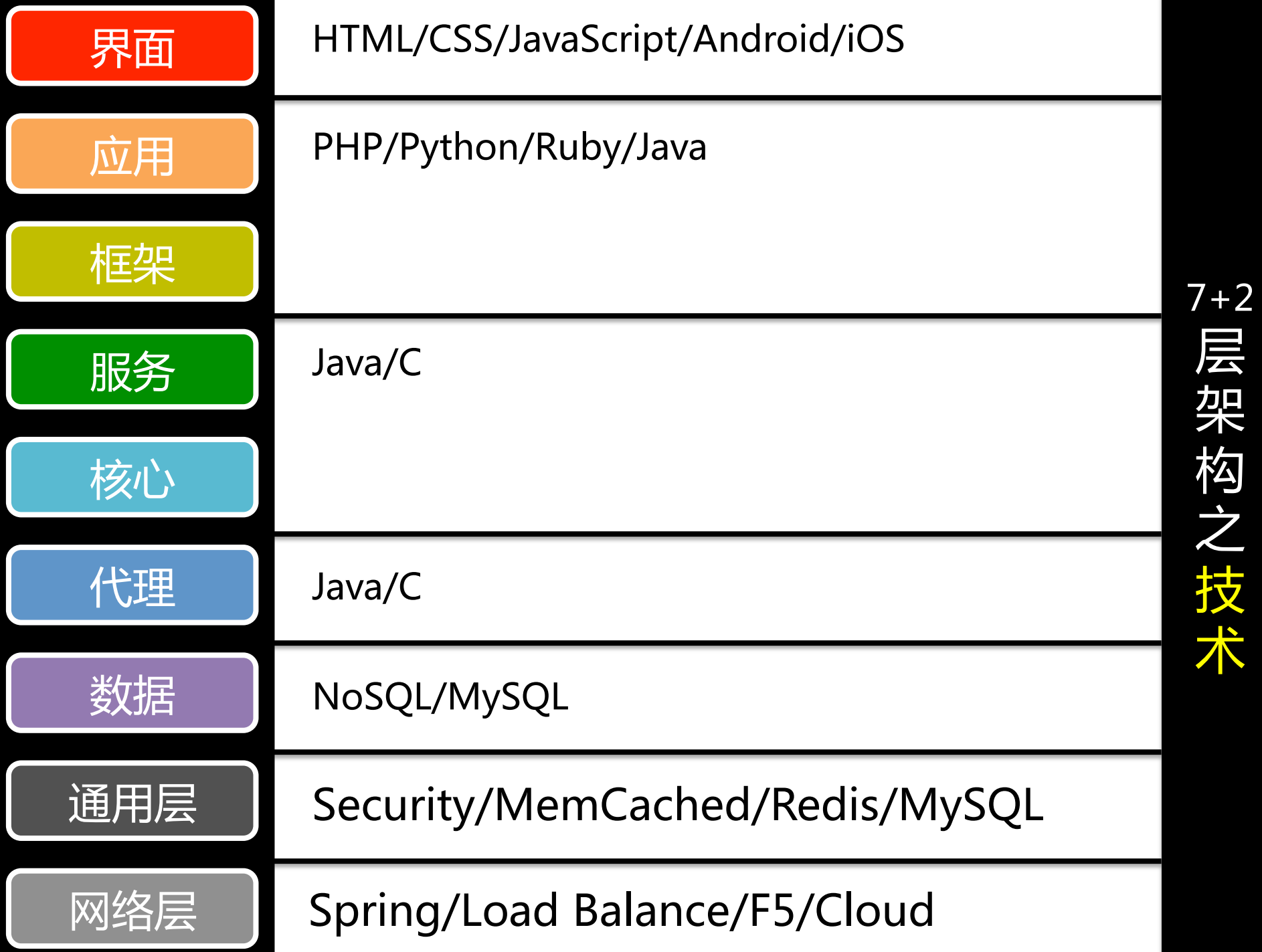
如果是 Web 应用，则这里的 Z3 包含 Web 浏览器

界面	可用性（Usability）
应用	可用性（Usability）,扩展性, 安全（防攻击）
框架	
服务	可用性（Availability），安全
核心	高效率
代理	透明性
数据	安全性，持久性
通用层	通用性
网络层	稳定，可用（Availability），伸缩，效率

7+2层架构之系统素质

界面	了解用户，且具有审美观
应用	了解市场与用户，具有产品设计能力
框架	了解市场与用户，且擅长归纳总结
服务	了解领域和公司的战略，有接口设计能力
核心	有比较强的计算机知识与算法能力
代理	了解领域与合作夥伴
数据	了解领域与数据库
通用层	了解语言、程序框架、各种开源项目
网络层	了解操作系统、网络、云计算

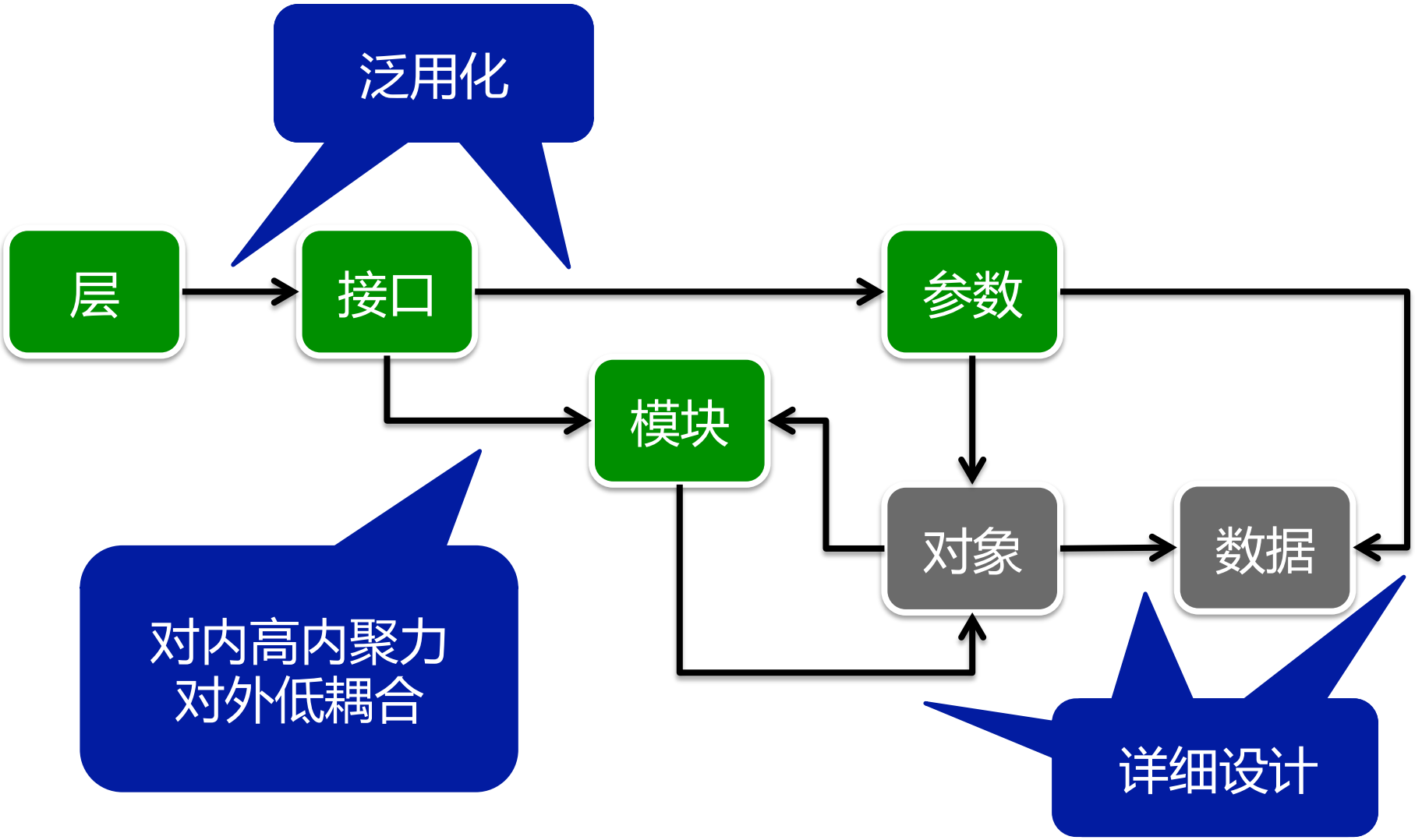
7+2层架构之人员素质

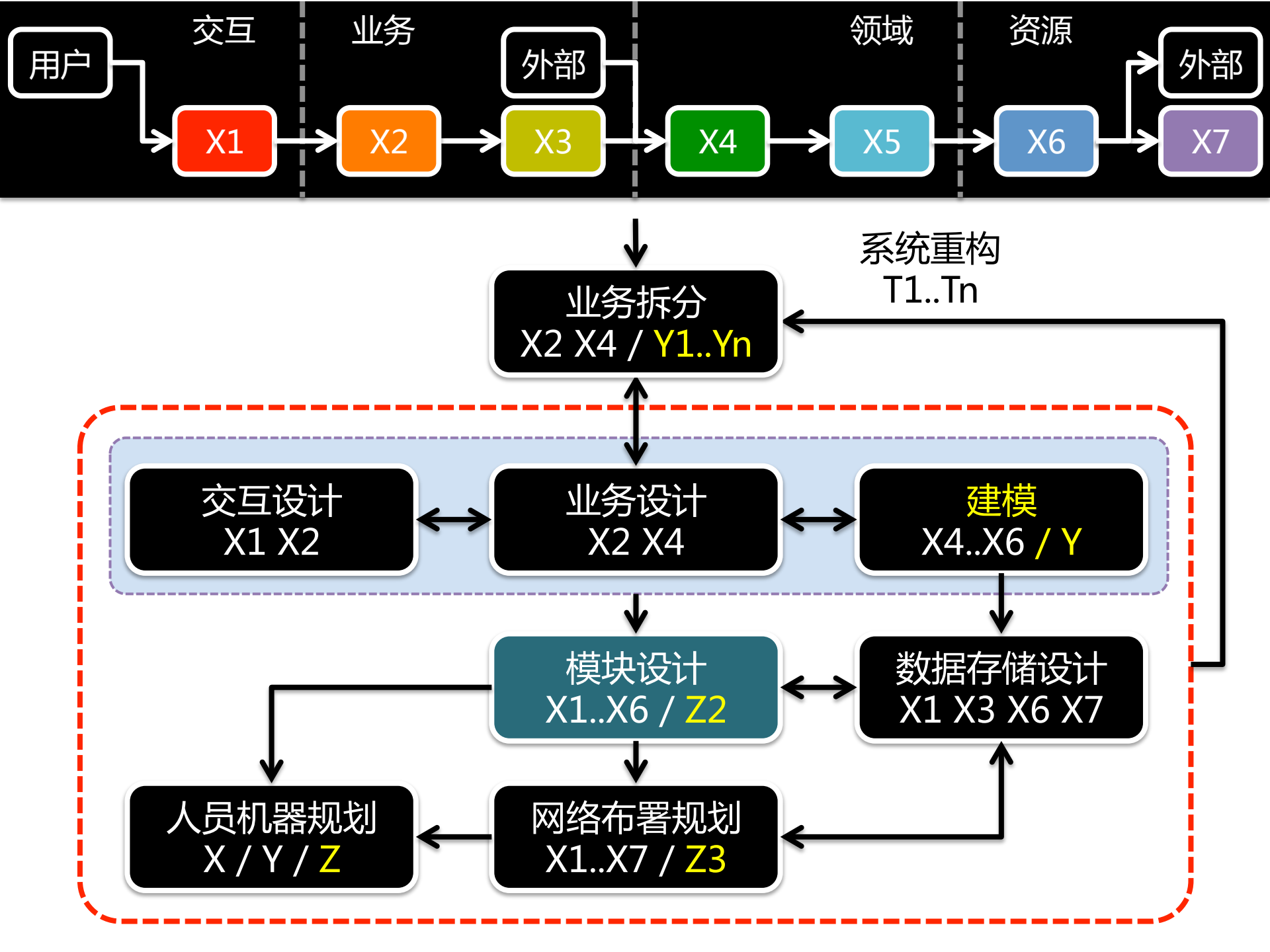

界面
HTML/CSS/JavaScript/Android/iOS
应用
PHP/Python/Ruby/Java
框架
服务
Java/C
核心
代理
Java/C
数据
NoSQL/MySQL
通用层
Security/MemCached/Redis/MySQL
网络层
Spring/Load Balance/F5/Cloud
7+2
层架构之技术


界面
每周
应用
每月
框架
每季
服务
每年
核心
代理
不一定
数据
不一定
通用层
每季
网络层
每半年
7+2
层架构之代码迭代进度

架构流程

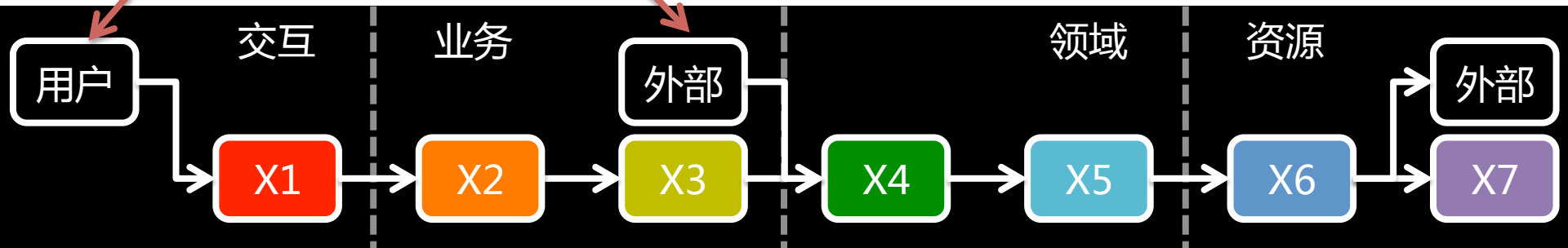
架构推导过程



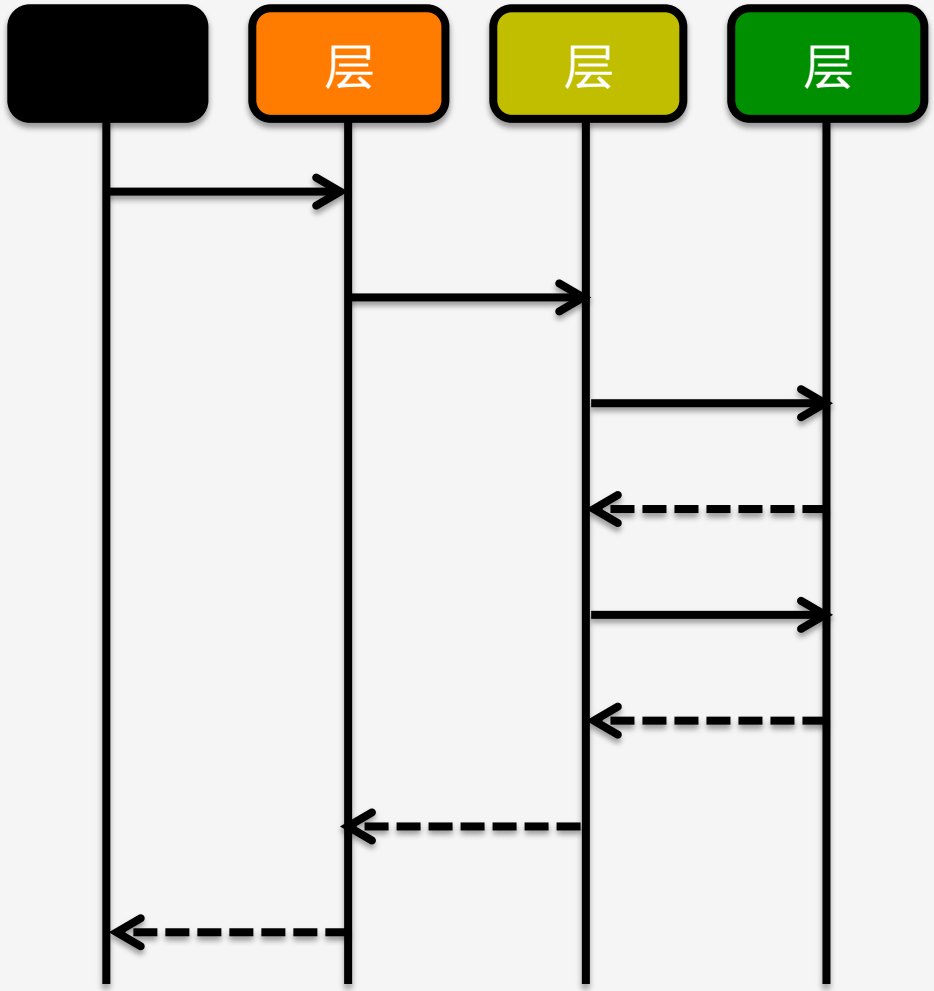

交互
业务
领域
资源
用户
外部
外部
X1
X2
X3
X4
X5
X6
X7
系统重构
T1..Tn
业务拆分
X2 X4 / Y1..Yn
交互设计
X1 X2
业务设计
X2 X4
建模
X4..X6 / Y
模块设计
X1..X6 / Z2
数据存储设计
X1 X3 X6 X7
人员机器规划
X / Y / Z
网络布署规划
X1..X7 / Z3

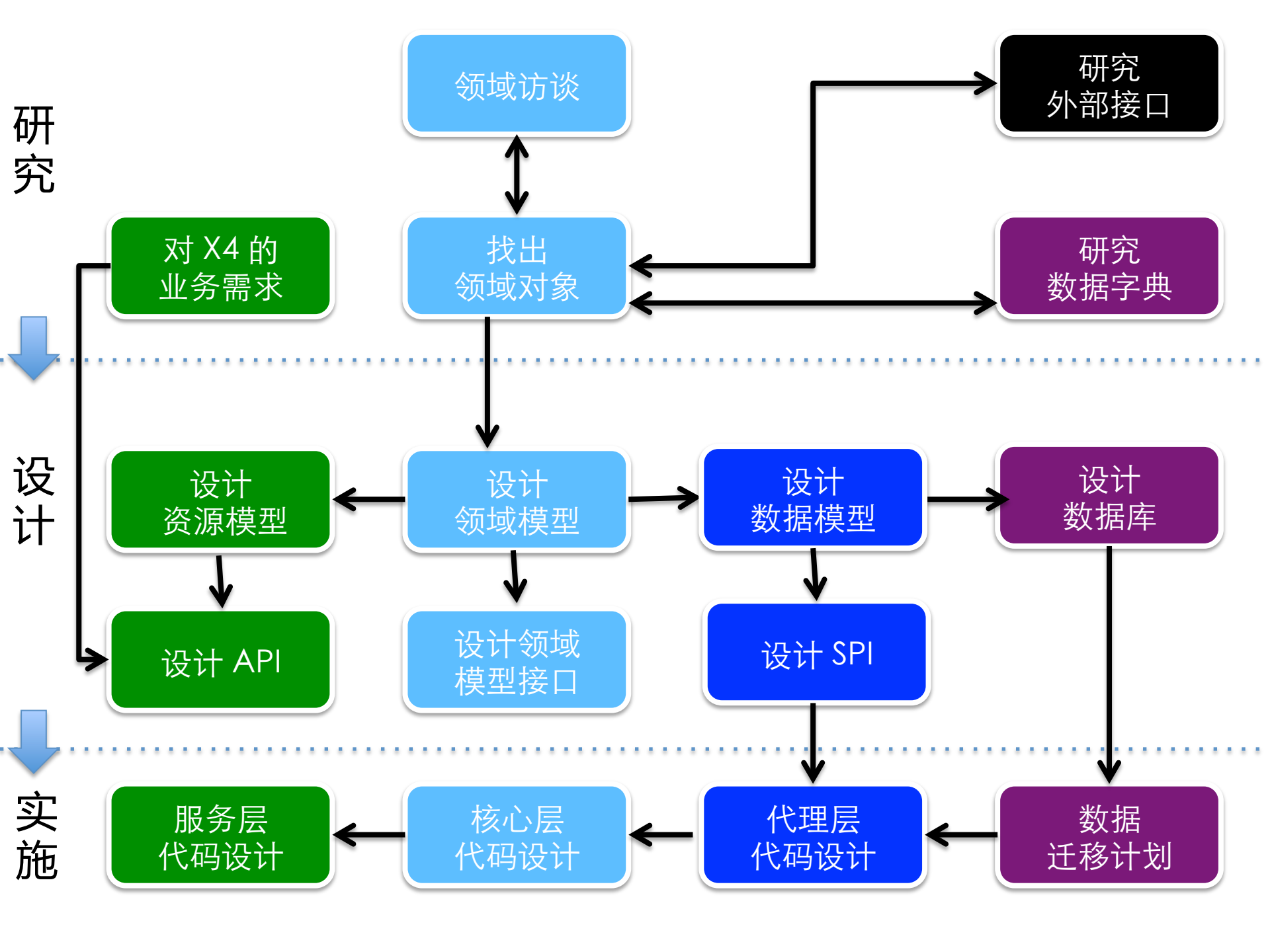
根据业务拆分系统

角色名称	人？	系统名称	用例描述

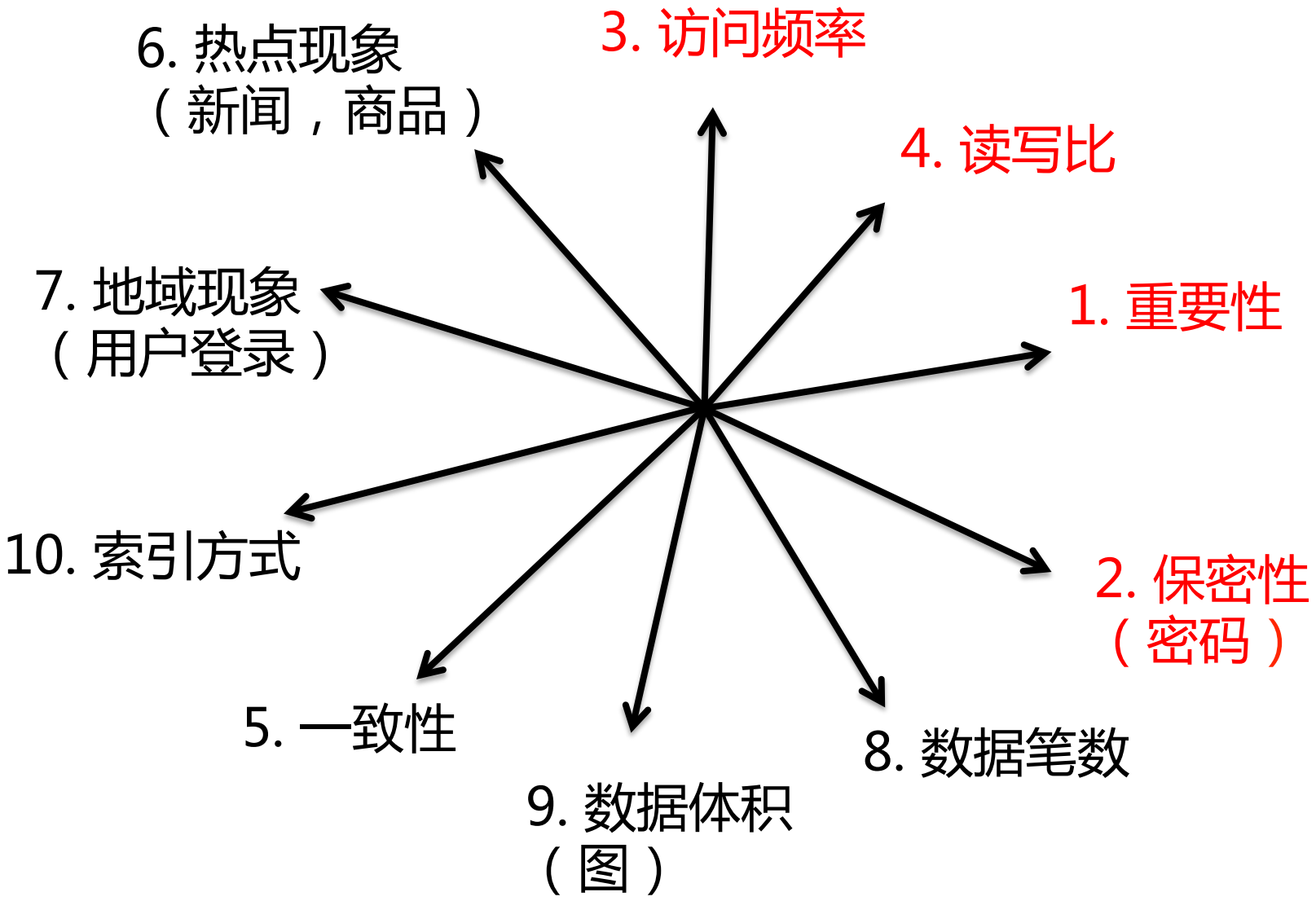


业务设计（找出接口与参数）




研究
设计
实施
领域访谈
研究
外部接口
对 X4 的
业务需求
找出
领域对象
研究
数据字典
设计
资源模型
设计
领域模型
设计
数据模型
设计
数据库
设计 API
设计领域
模型接口
设计 SPI
服务层
代码设计
核心层
代码设计
代理层
代码设计
数据
迁移计划

红色对于领域模型的设计有帮助



详编模块定义

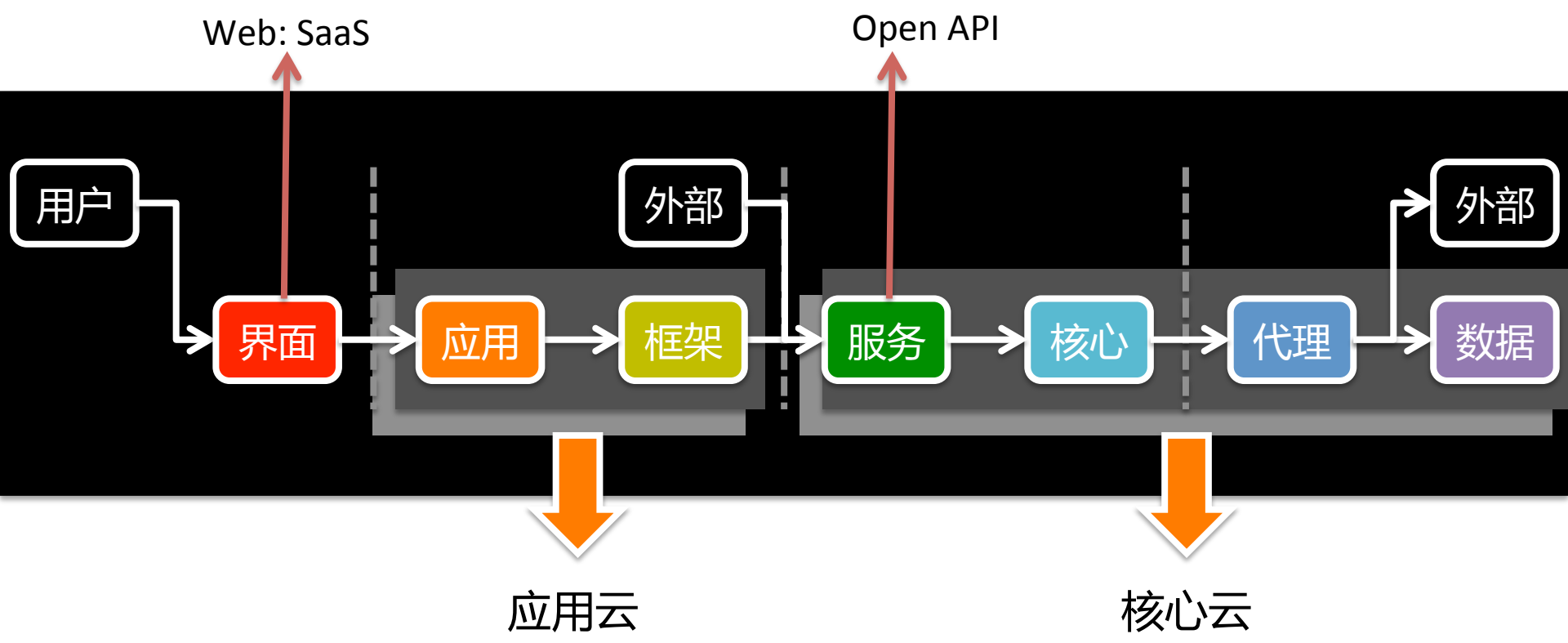
ID		名称	
座标		负责人	
目的			
依赖			
接口与参数			


数据特性
分析
数据库
选型
备份策略
十大指标
数据库特徵表
缓存策略

云计算与大数据

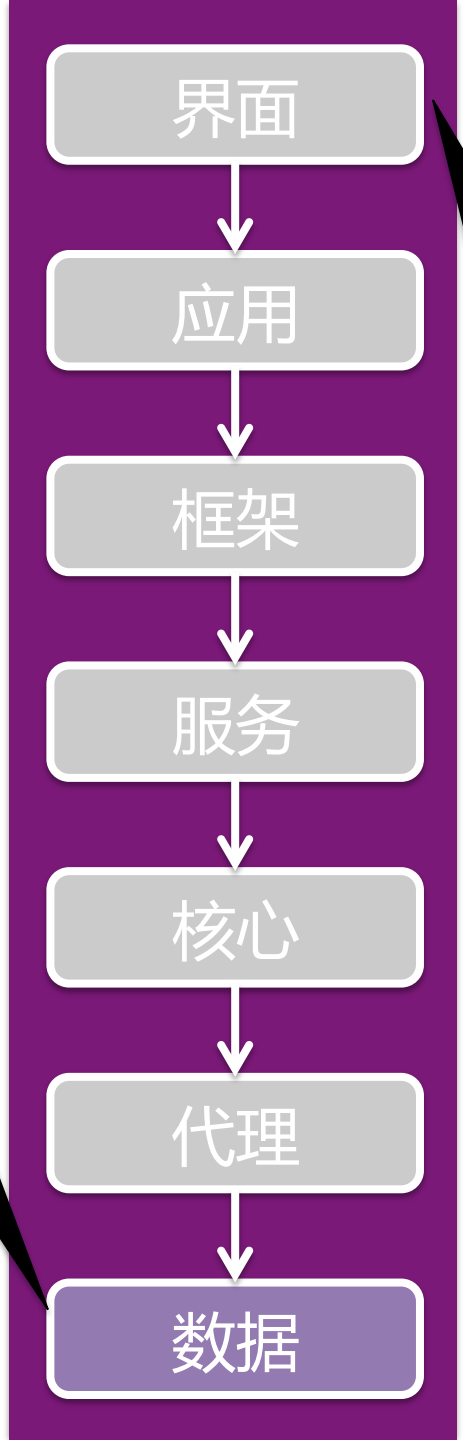
Z 座标和云平台的关系

XZ 座标和云平台的关系



所有业务系统的服务层以下都要收归集团统一管理。优点：

- 只有内部（服务以下就是内部）可以依赖
- 数据方便管理
- 方便管理与外部系统的关系
- 可以逐渐形成统一平台

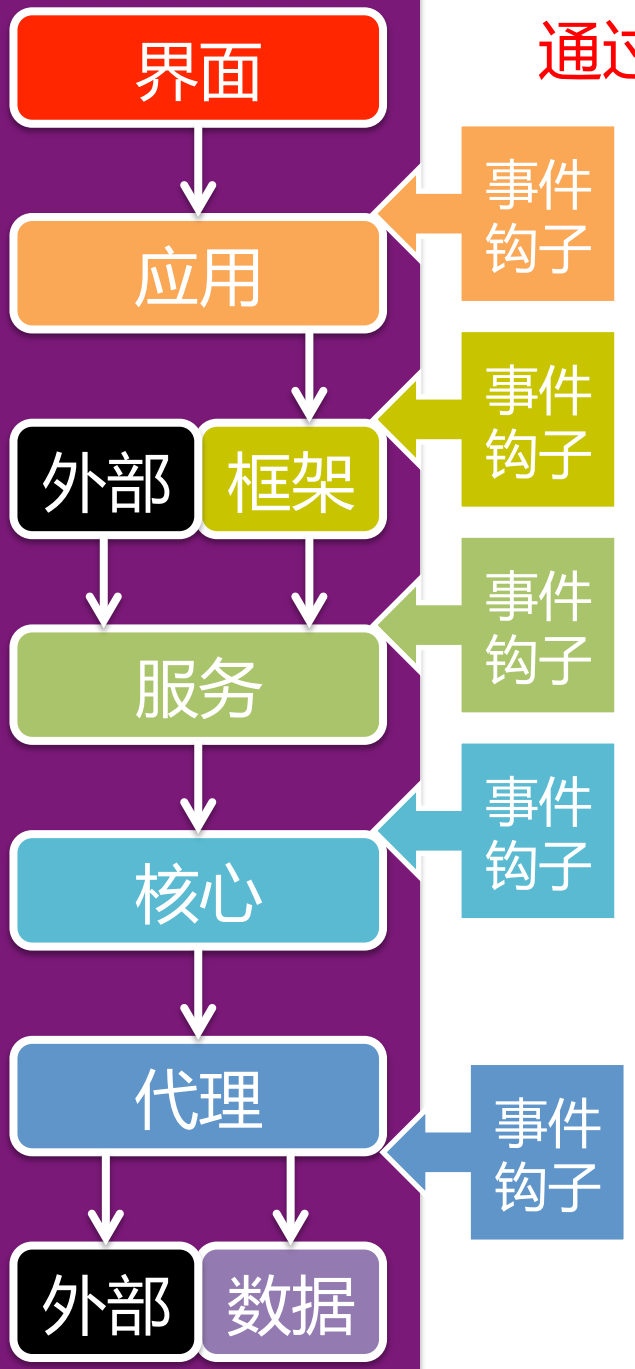


多数人提到大数据时，都是指来自这里（数据库）的数据。而系统运行过程中的许多有价值的数据，都被丢弃忽略了。

比方说：当多数用户到了某页面后，就不往下进行，可能页面设计有问题，只要改善页面用户体验，业绩就会大幅提升。但只通过数据库，无法分析出这点

七层架构与五个事件钩子

通过 Z3 层的配置，可以直接得到事件

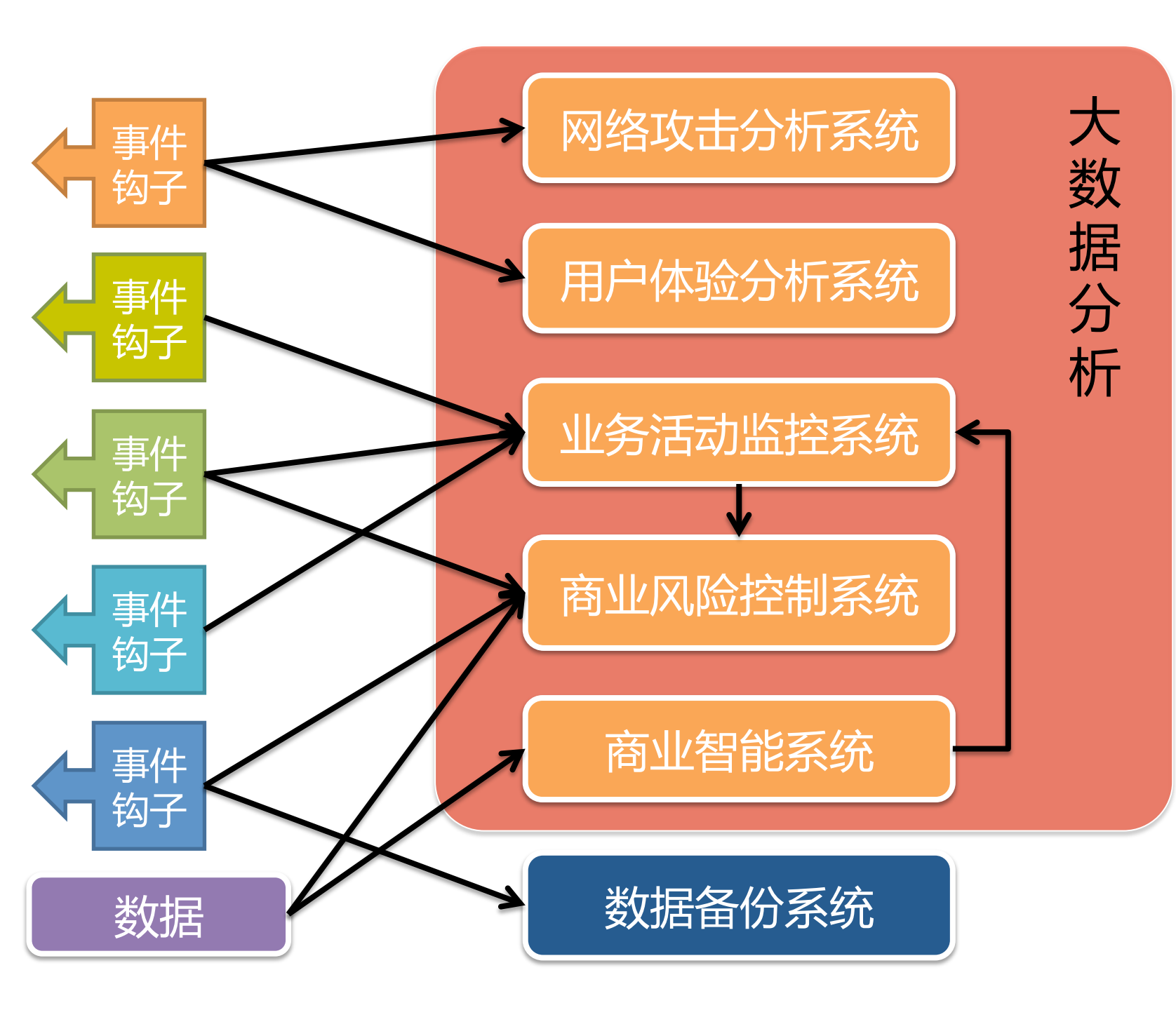


业务系统的层与层之间，都可设置事件钩子，避免系统改造。

事件钩子可以将事件原地处理，也可将事件送到其他服务器处理

事件钩子同时具备日志的效果，关键事件可以送到统一日志中心

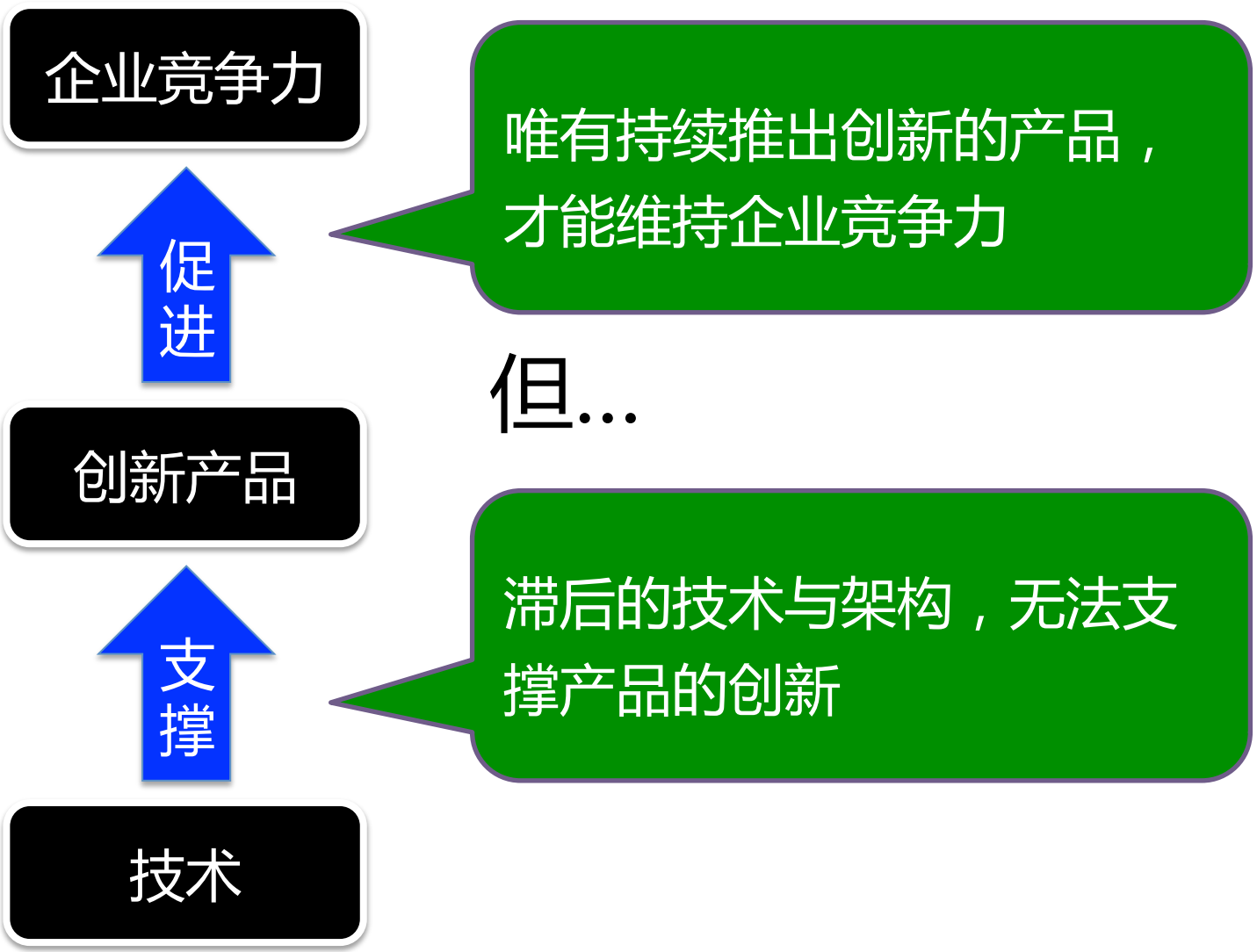
注意：代理层的事件钩子是设置在代理层出口，而不是入口



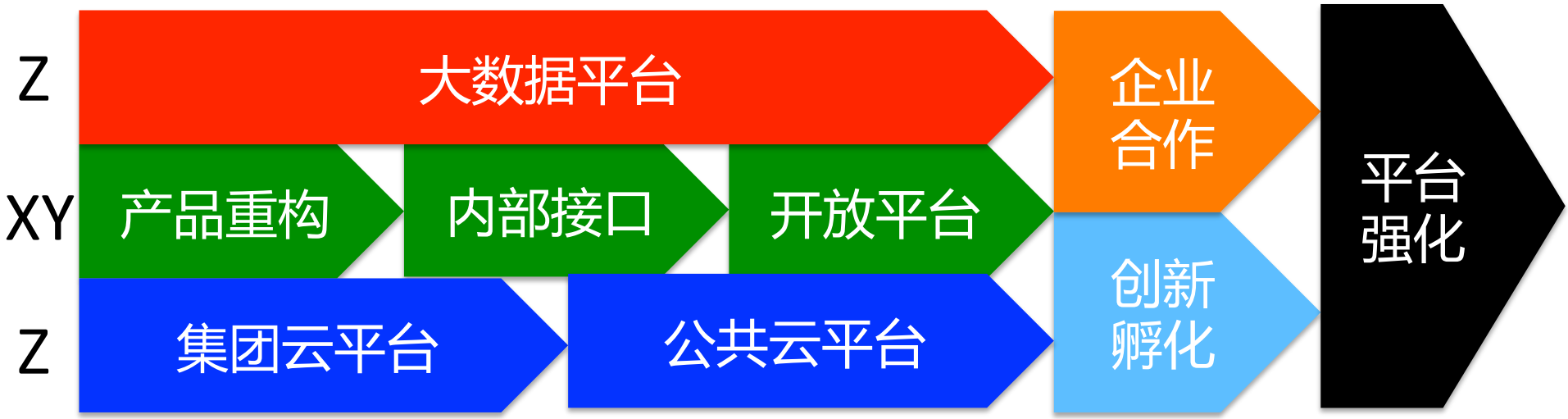
事件信息流向与数据分析系统

平台化战略

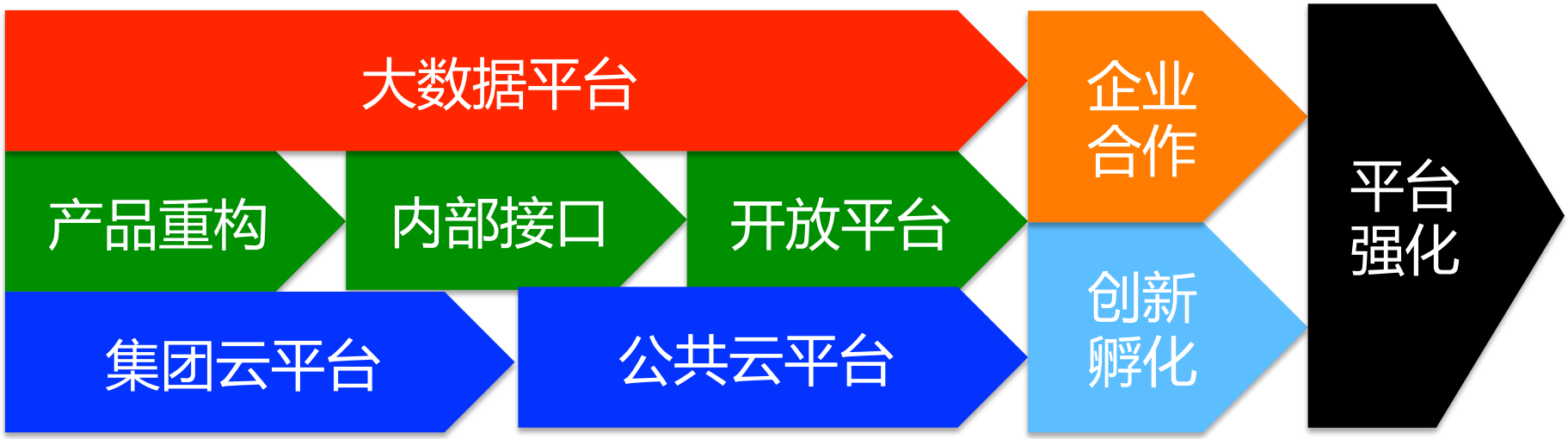
产品创新需要技术支撑


企业竞争力
促进
唯有持续推出创新的产品，才能维持企业竞争力
但...
创新产品
支撑
滞后的技术与架构，无法支撑产品的创新
技术

平台化促进合作与创新，进而巩固平台实力

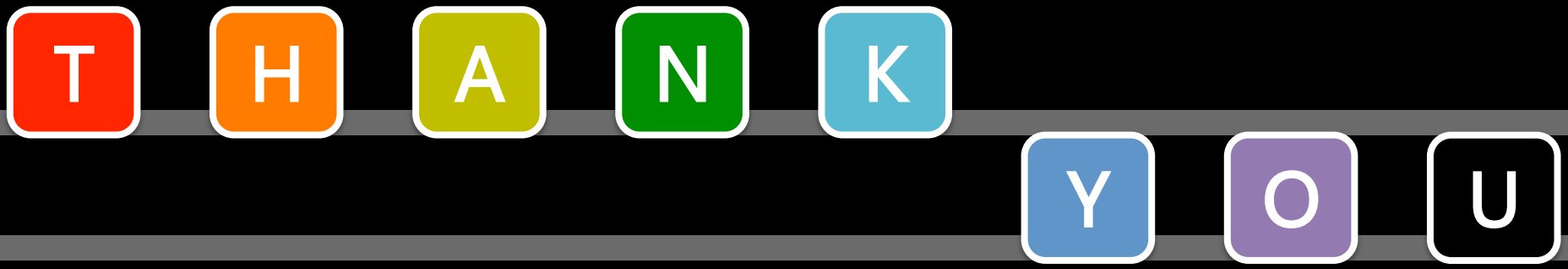


想要做到 ...



依靠的是 ...

首席架构师的架构设计能力
与公司的执行力



蔡学镛 email: JerryTsai1218@Gmail.com
新浪微博 @蔡学镛

Version: 2014-8-17