

# 顺丰微服务探索及实践

文彦峰

顺丰科技

# 公司简介

顺丰科技就是为了解决物流互联网化而诞生的



每天数以百万计的包裹从顺丰发出，这么多的包裹其实是通过科技作为驱动力来精准送达用户手中。

# 公司简介



**顺丰科技有限公司**是隶属于顺丰速运集团，拥有超过2000人的IT专业技术队伍，具备“深圳市重点软件企业”、“国家高新技术企业”等资质。顺丰科技目前致力于支持顺丰内外多项业务的研发、运维和内容服务，包括智慧物流解决方案、大数据产品、云计算产品、智能穿戴、无人机、人工智能等多个方向。无论从系统应用规模、监管手段、调度能力、研发能力、科技含量、处理能力等方面，顺丰在国内速运行业都处于领先地位。地址识别技术、路由规划技术等技术领域已经达到国际标准水平。



2000人的IT专业技术队伍



230多个大中型系统



3个数据中心



1.5PB交易数据

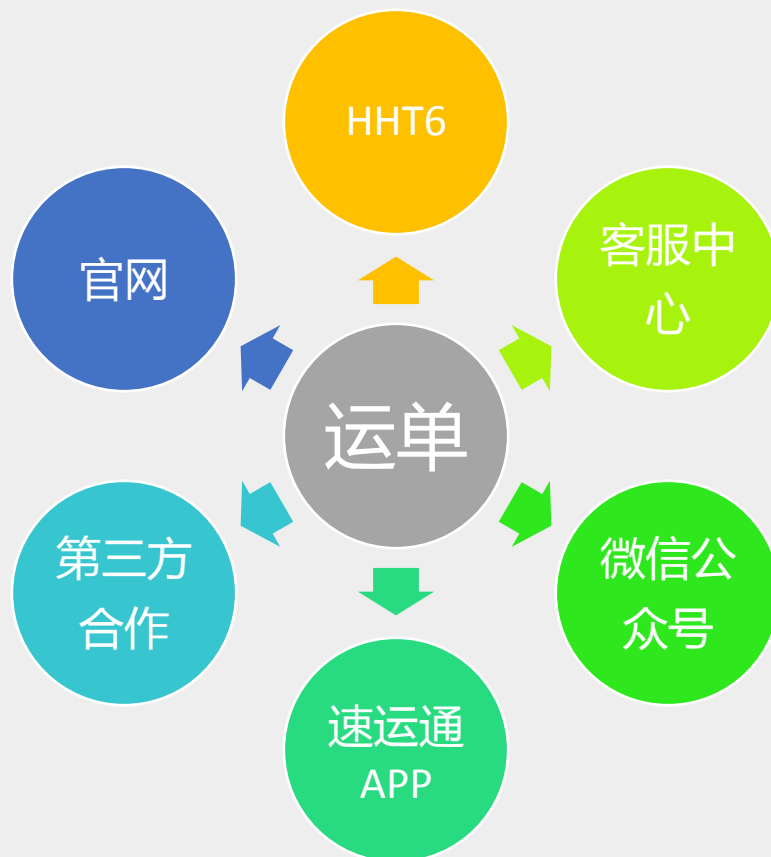


12000个网络分支节点

# 问题篇

# 渠道的变化

官网



互联网时代

移动互联网时代

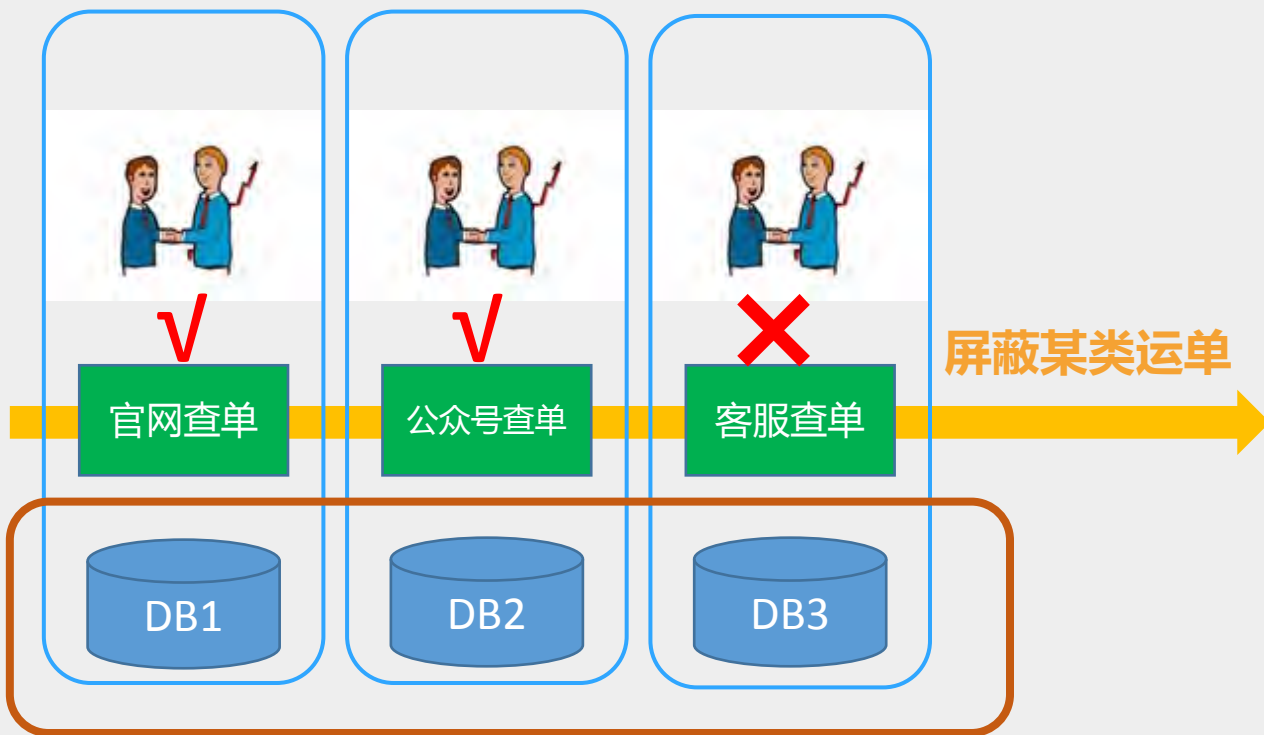
# 产品化

用非常简单通用的构建可以组合成非常复杂的形状



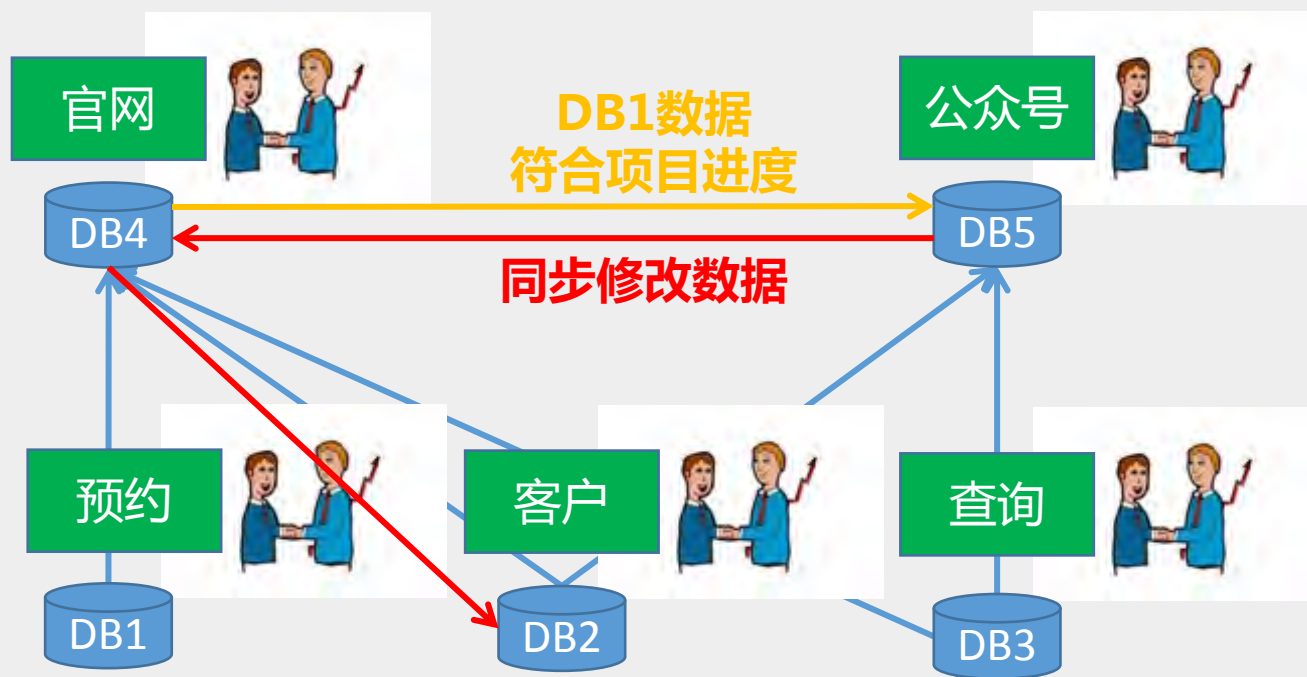
## 研发成本高：

- 代码重复率高、耦合严重
- 跨渠道的需求变更困难
- 发版周期严重依赖
- 性能的扩展变更困难
- 渠道间的风险影响



## 数据关系乱：

- 主数据散落
- 数据流杂乱
- 上线排期互相依赖





## 运维效率低：

- 测试、部署、验证成本高
- 发布周期长，窗口少
- 可伸缩性差
- 可靠性差

## 客户体验系统



预约

查询

会员

DB1

DB1

DB1



运维难度增加



# 微服务的优势



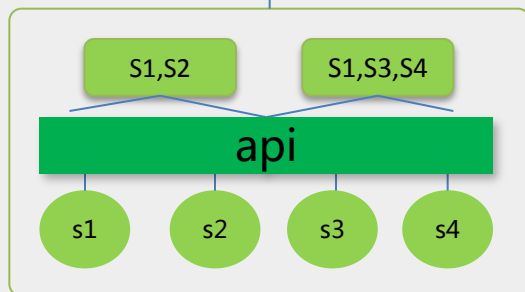
技术  
开放



敏捷  
开发

微服务优势

低耦合  
高重用



硬件要求低  
扩展性好



所以  
我们选择了**微服务**

# 设计篇

# 你准备好了吗？

- 为什么用？
  - 大型复杂系统？
  - 新业务领域？
  - 不用会怎么样？（分布式和单体）
- 技术
  - 开发环境、测试环境、预生产环境（基础设施）
  - 自动化集成、自动化测试、自动化发布、自动化安全扫描、自动化业务验证
  - 配置分离、动态配置管理、容器化
  - 常见问题的解决方案，比如数据一致性
  - 治理平台、监控平台
  - 线上问题定位
- 组织
  - 架构团队、中间件开发团队
  - 敏捷开发模式
  - 运维团队准备好了吗？
- 沟通
  - 沟通机制（敏捷）
  - 纪律性

# 微服务的划分

一个子业务？

一个数据库？

一个接口？

## 细粒度优点：

- (1) 服务都能够独立部署
- (2) 扩容和缩容方便，有利于提高资源利用率
- (3) 拆得越细，耦合相对会减小
- (4) 拆得越细，容错相对会更好，一个服务出问题不影响其他服务
- (5) 扩展性更好

....

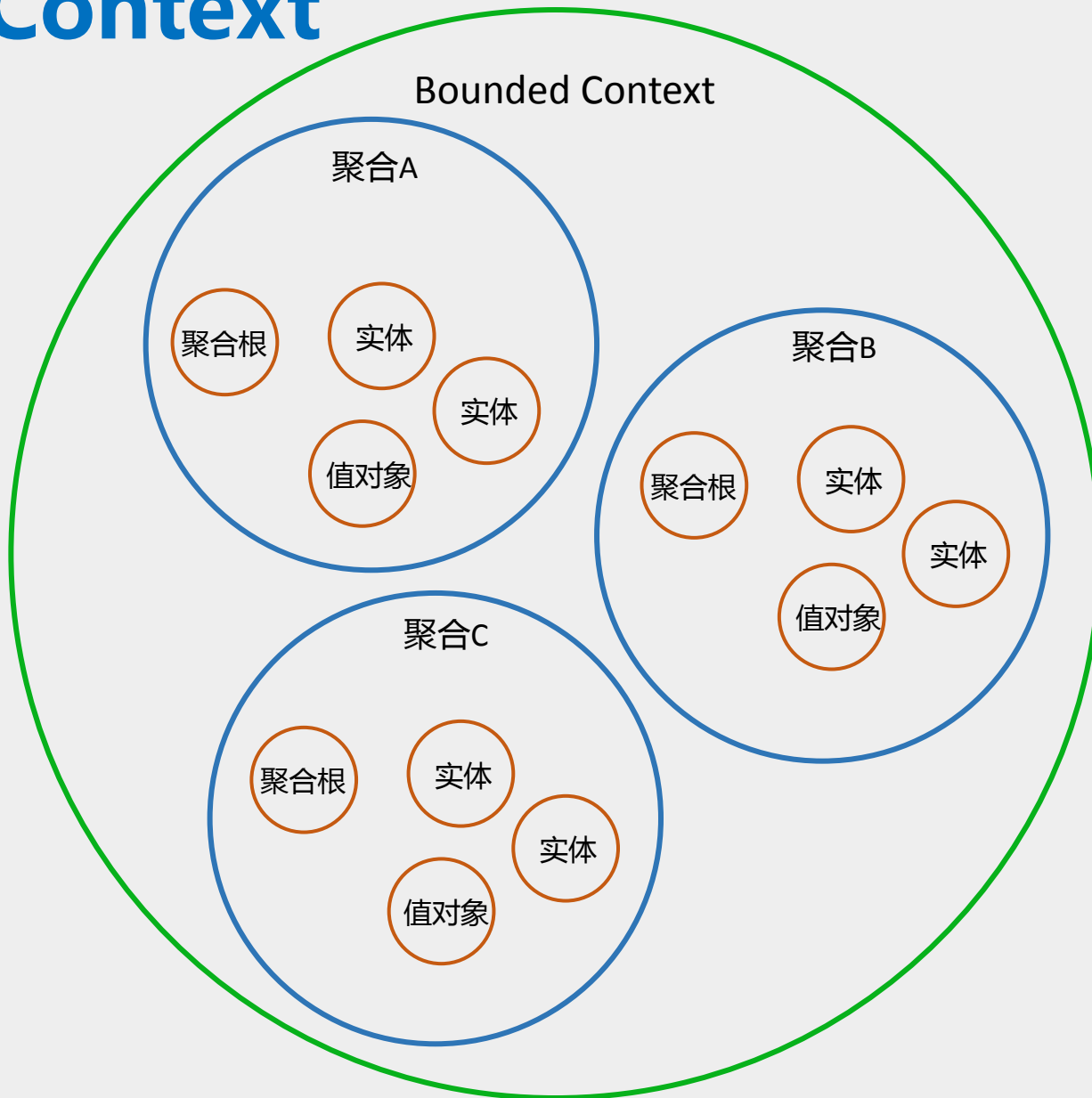
## 细粒度缺点：

- (1) 拆得越细，系统越复杂
- (2) 系统之间的依赖关系也更复杂
- (3) 运维复杂度提升
- (4) 监控更加复杂
- (5) 出问题时定位问题更难

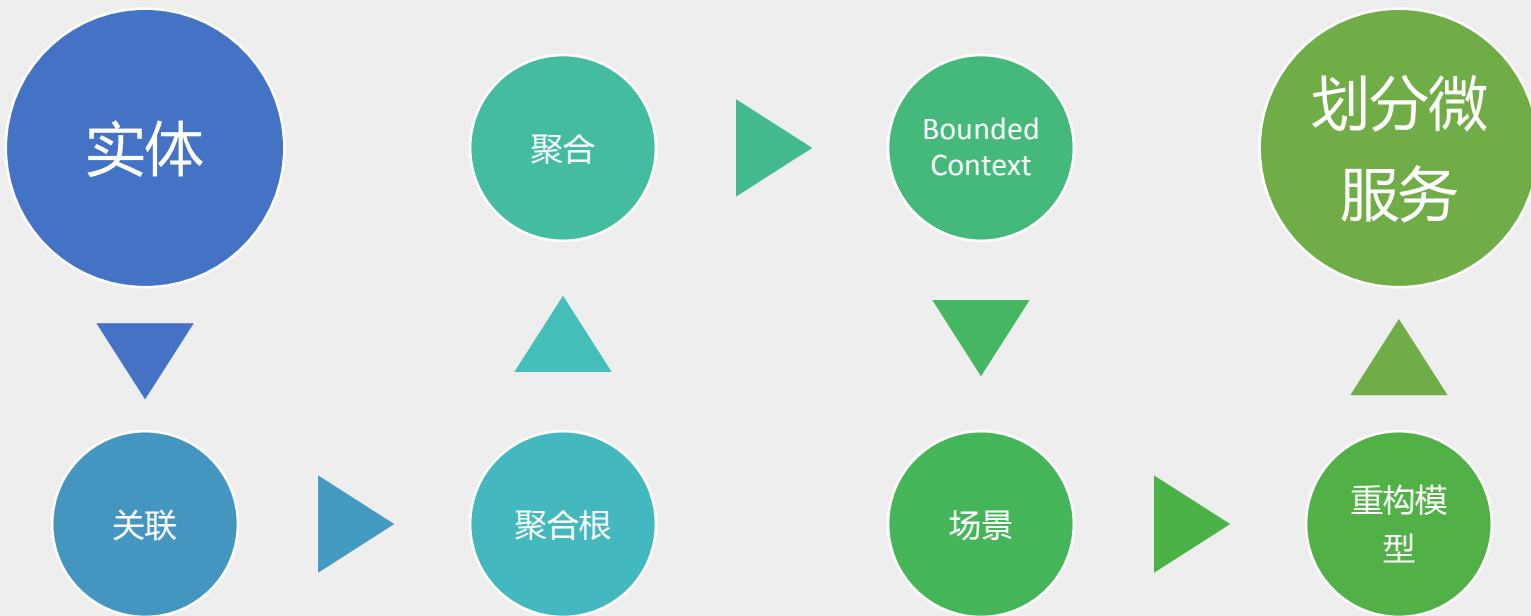
....

# Bounded Context

生命周期



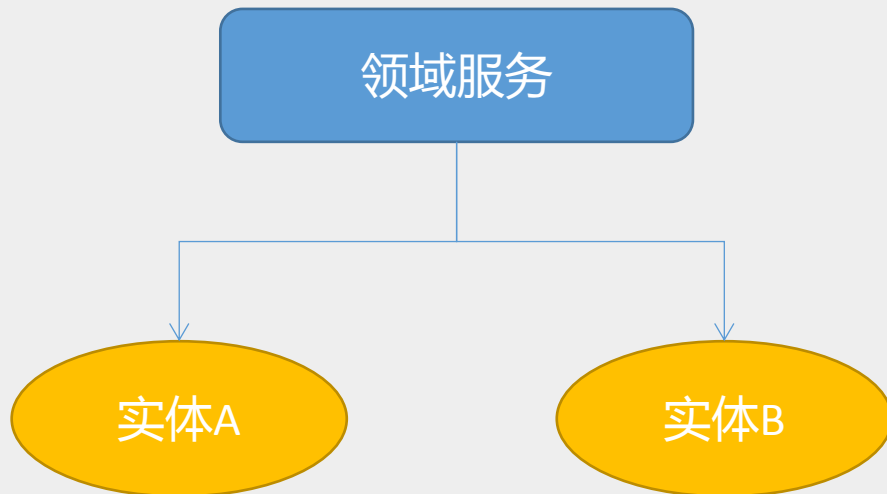
# 微服务的设计步骤





# 微服务的领域服务

跨聚合实体业务逻辑，没办法放到某个实体中。  
当领域中的某个重要过程或转换操作不属于实体或值对象的自然职责时，  
应该在模型中添加一个作为独立接口的操作，并将其申明为Service。



# 微服务的集成

微服务的集成做到好，可以保持自治性、可以独立发布修改和发布。

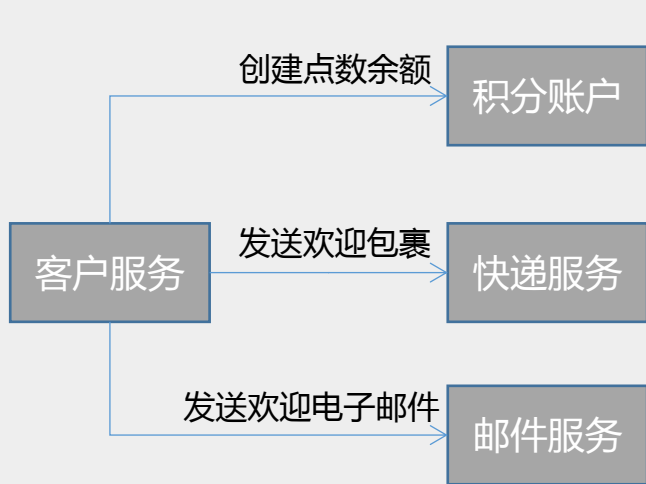
## 原则：

- 避免破坏性修改  
服务的一些修改不能导致该服务的消费方发生改变（破坏聚合）
- 保证API与技术的无关性
- 保证API的易用性
- 隐藏内部实现细节

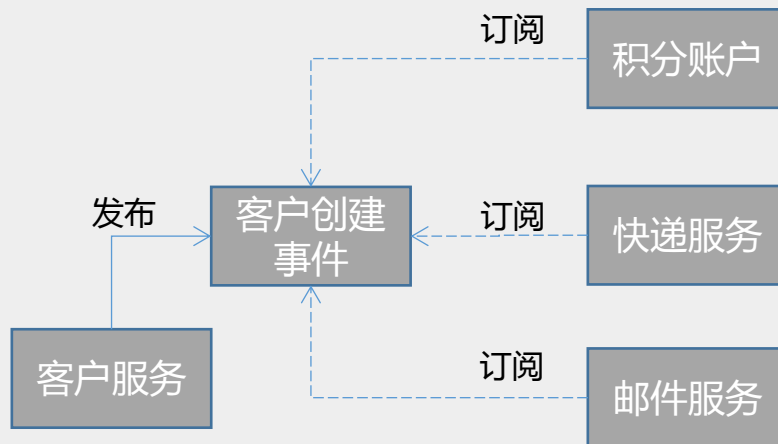
# 微服务的集成

**编排**：同步调用一组服务，等待各个服务的返回结果。优点知道业务流程中每一步跨服务调用结果，缺点容易承担太多的调用，太耗时，导致调用方的不稳定性。

**协同**：异步调用一组服务或服务调用加入队列中，降低服务之间的耦合度，带来的额外工作业务流程跨服务的监控，不过可通过消费方处理完成后，回调服务方告知处理结果。

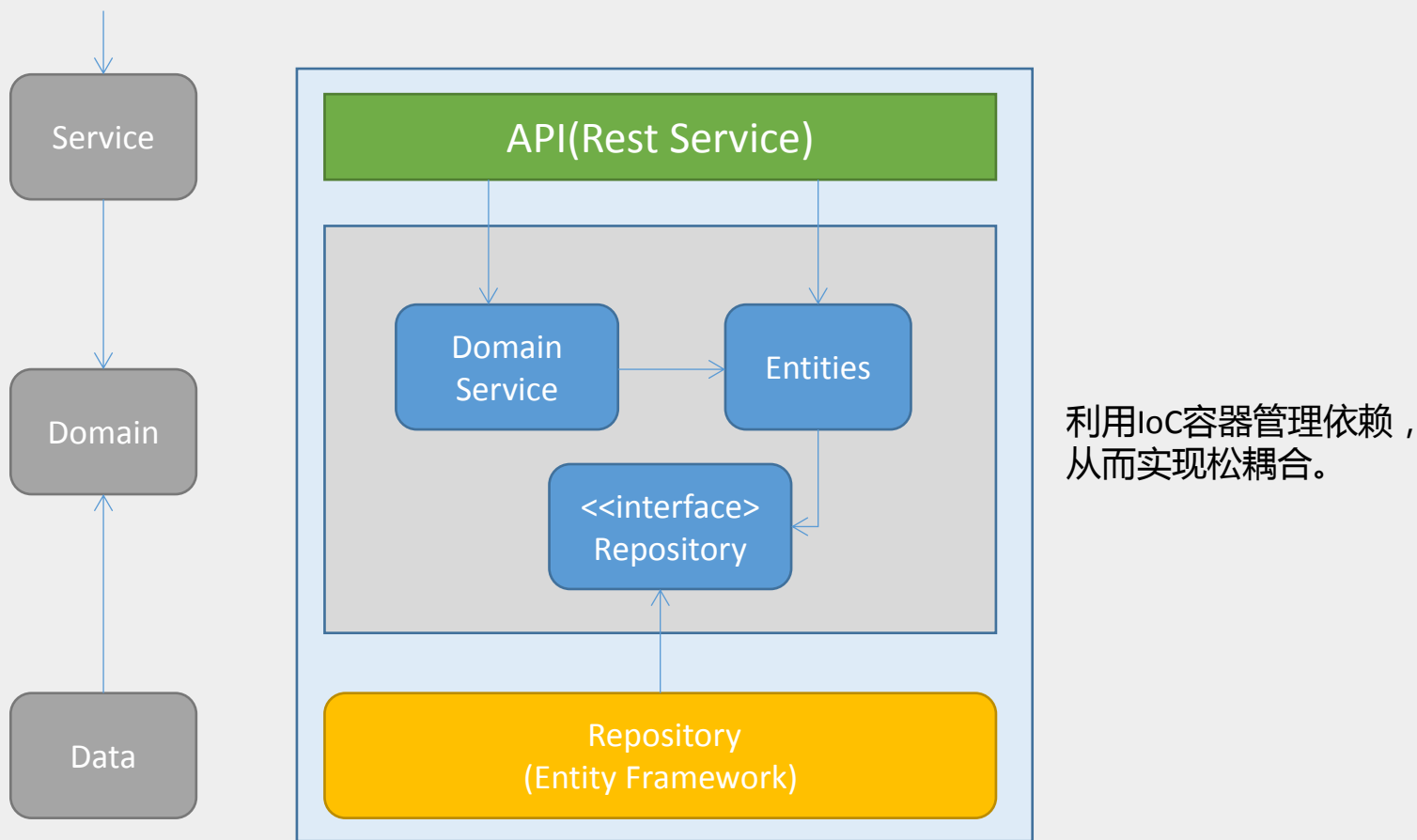


编排



协同

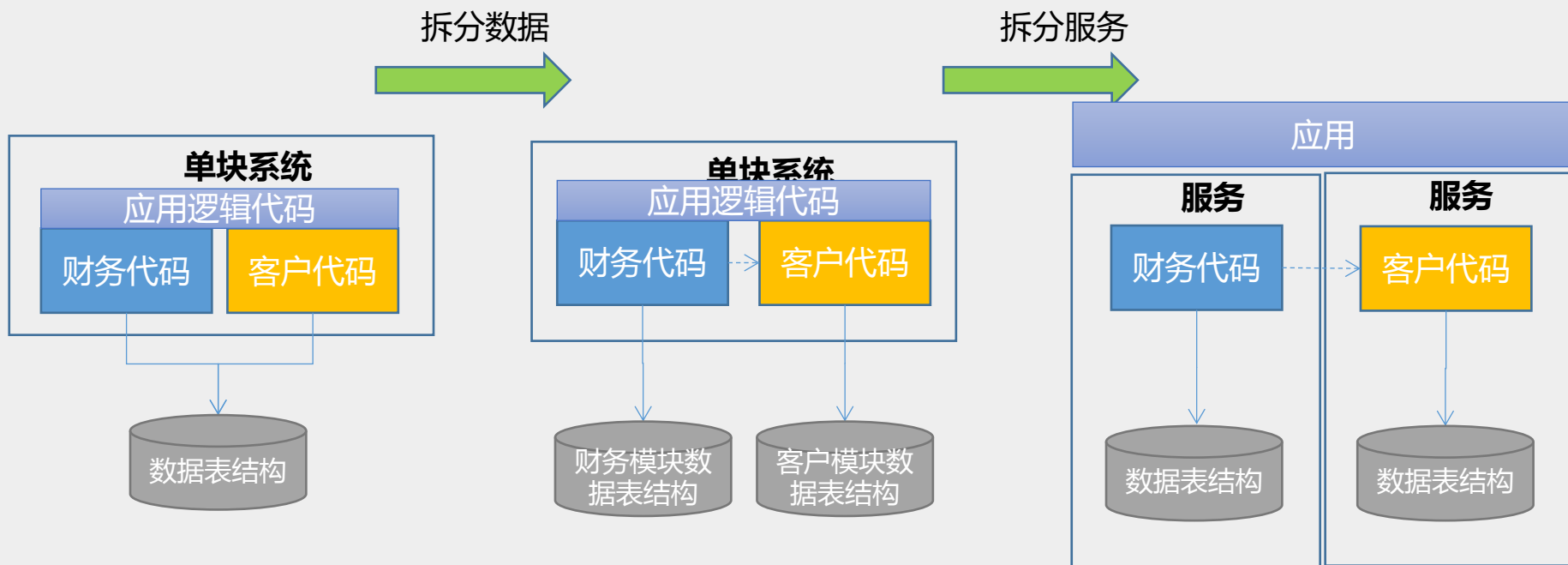
# 基于领域驱动的微服务实现



遵循依赖倒置原则，将Repository的实现注入到Entity或Domain Service中。

# 存储过程逻辑

## 单块系统拆分方案1-先拆数据库



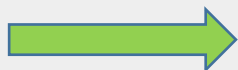
1. 不同模块的数据表拆开
2. 模块之间改成接口调用

1. 应用和服务分离
2. 将模块拆分成不同服务

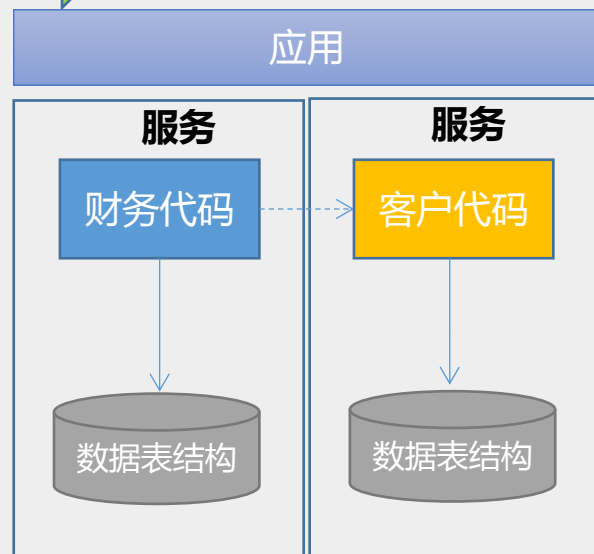
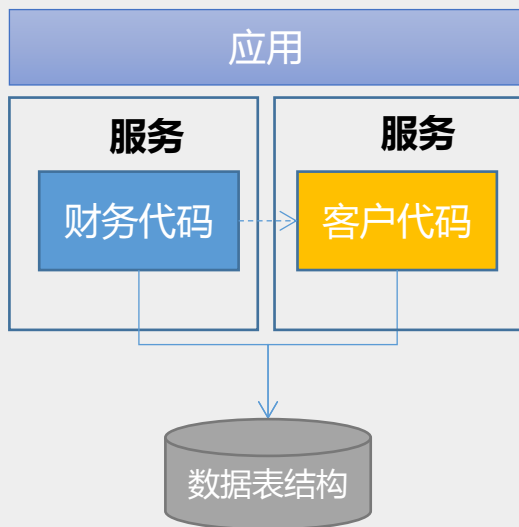
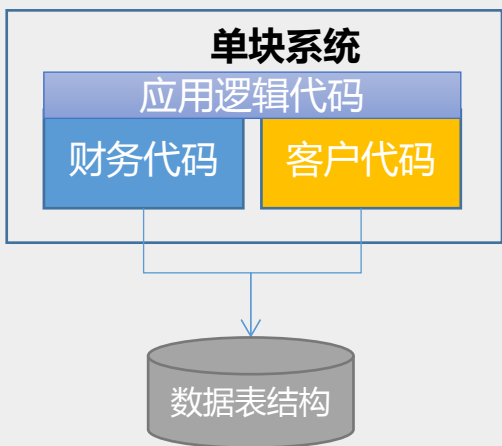
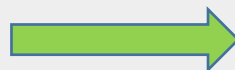
# 应用代码逻辑

## 单块系统拆分方案2-先拆服务

拆分服务



拆分数据



1. 数据库表保持不变
2. 将应用逻辑拆分出来
3. 将不同的业务模块拆分成服务

1. 重构服务，将数据库进行拆分

# 微服务设计原则

## 微服务划分

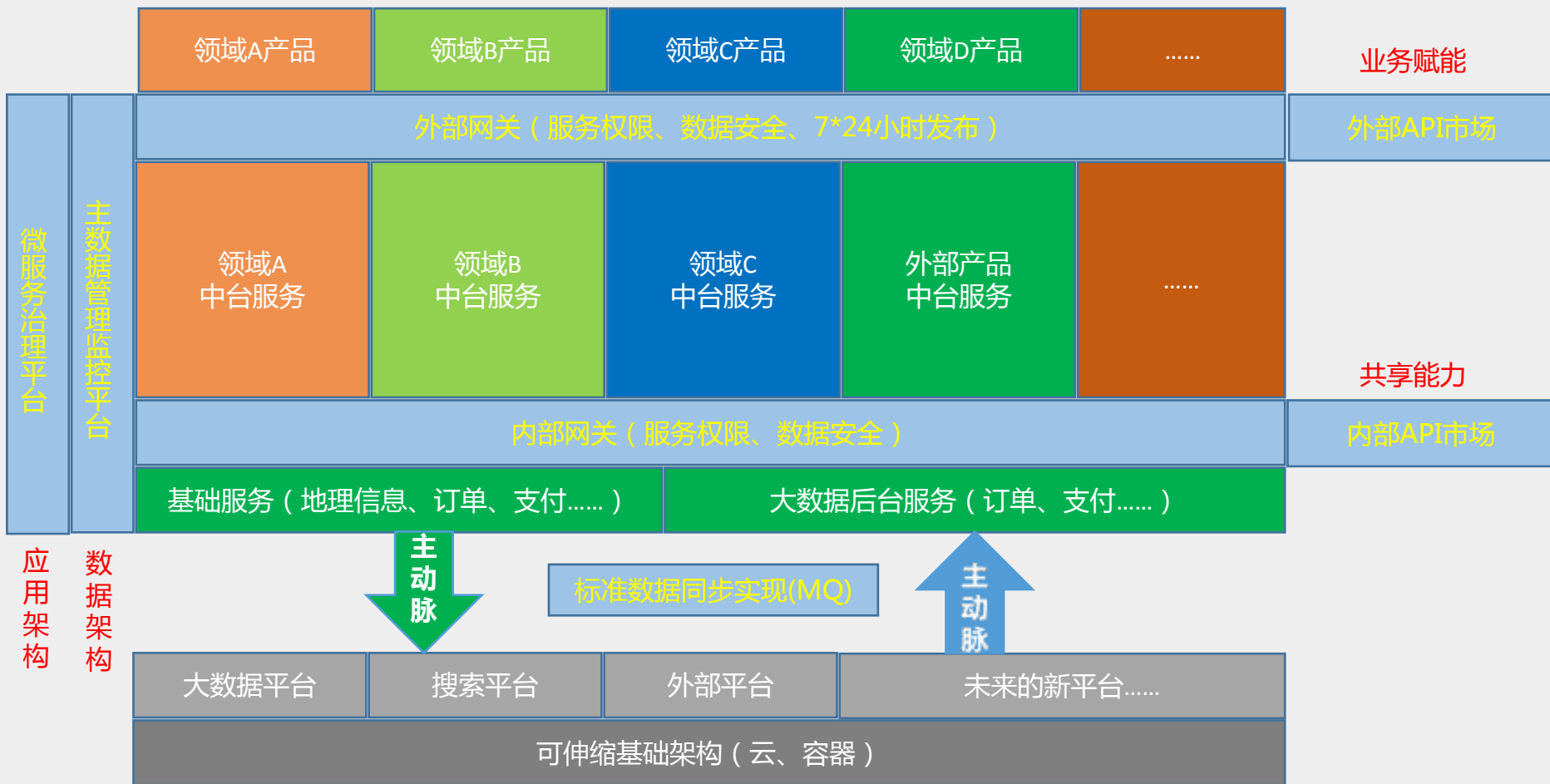
- 根据**业务领域**或**特定功能组件**划分微服务，粒度根据具体业务场景。
- 满足划分出来的微服务**不存在循环依赖**。
- 微服务一般**不跨多个数据库**，不同存储方式除外，例如同时使用关系数据库和Nosql数据库；读写分离除外。
- 对同一业务对象操作的服务不进行拆分，**不应出现多个微服务操作（修改/新增）同一数据对象**。

## 微服务调用

- 同一套环境内（同一领域）的**微服务之间采用RPC调用方式**。
- 微服务之间不允许循环依赖调用，一个请求中**调用层次尽量不要超过三层**。
- **不同环境之间调用微服务需通过网关调用**，例如速运业务中的微服务需要调用某公共微服务，则通过公共微服务的网关来调用该公共微服务。
- 应用层**通过网关调用**微服务。

# 微服务带来的影响

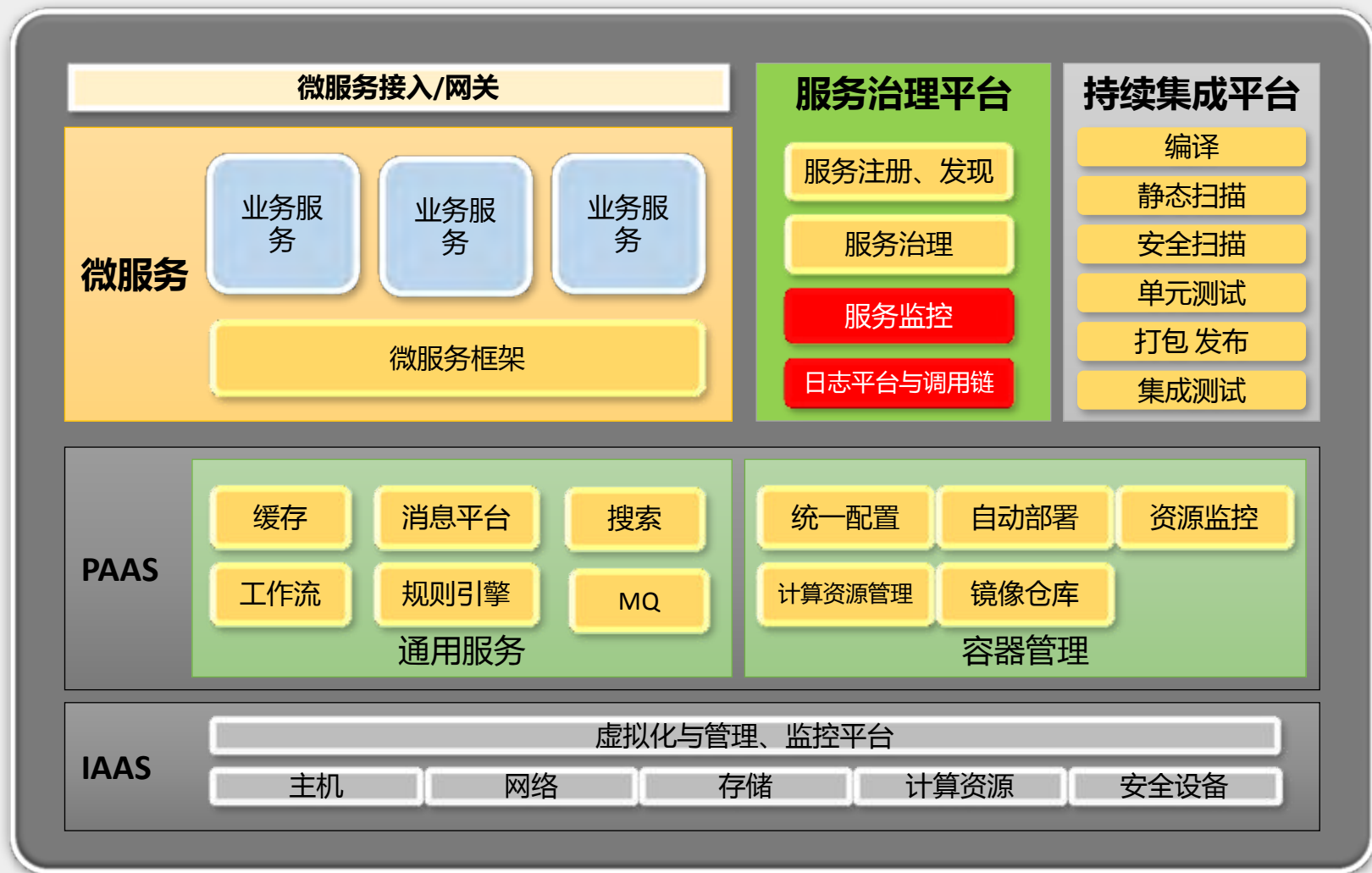
企业信息透明便于管理决策、企业能力开放便于整合集成





# 工具篇

# 微服务体系概览



# 微服务体系



LeechV2.0代码生成助手

首页 创建工程 基础工程 其它需求

工程结构

- ggs
  - ggs-common
    - src/main/java
      - com.sf.ggs
        - pom.xml
  - ggs-dao
  - ggs-manager
  - ggs-service
  - ggs-web
    - pom.xml
  - sss
  - test
  - tester

文件内容 配置文件 页面设计

表格

表名唯一ID 必填项

添加选项 新增 修改 删除

列	scoreid	scoreid	
列	usualscore	usualscore	
列	testscore	testscore	
列	id	id	
列	classid	classid	

数据库 单表 多表

- class
- score
- student

字段区域

字段名	数据类型	操作
scoreid	int	已添加
usualscore	int	已添加
testscore	int	已添加
id	int	已添加
classid	int	已添加

属性区域

replace指定内容替换: 1替换为指定

Max指定大于替换: 大于20替换为不满足

Min指定小于替换: 小于20替换为不满足

确定 取消

# “代码”机器人

## 模板工程

快速搭建工程

依赖版本统一

代码风格一致

## Leech

自动化生成代码

支持不同框架

使用便捷

任务或app标识	测试主键	期望纪元	0: 未成功; 1: 成功	创建时间	更新时间	操作
0	0	0	true	2016-10-27 16:01:55	2016-11-08 14:38:34	查看 修改 删除
100	100	99	true	2016-11-08 14:13:05	2016-11-08 14:13:05	查看 修改 删除
22	11	101	true	2016-07-26 15:27:50	2016-11-08 14:28:19	查看 修改 删除
22	22	2222	true	2015-11-11 12:12:12	2016-11-08 14:13:27	查看 修改 删除
5	50	1	false	2016-08-22 20:27:44	2016-11-09 09:51:53	查看 修改 删除
5	51	2	false	2016-08-22 20:27:44	2016-11-09 09:51:55	查看 修改 删除
5	52	3	false	2016-08-22 20:27:44	2016-11-09 09:51:56	查看 修改 删除
5	53	4	false	2016-08-22 20:27:44	2016-11-09 09:51:56	查看 修改 删除
5	54	5	false	2016-08-22 20:27:44	2016-11-09 09:51:59	查看 修改 删除
5	55	6	false	2016-08-22 20:27:44	2016-11-09 09:52:01	查看 修改 删除

10秒  
完成单表  
CURD

# 组件微服务

LeechV2.0代码生成助手

首页

创建工程

设计工程

基础工程

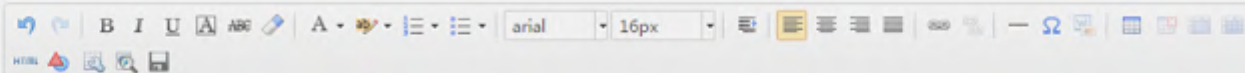
其它需求

文件内容

配置文件

页面设计

设计页面



表单设计器 - 清单

控件列表

使用教程

单行输入框

多行输入框

下拉菜单

单选框

复选框

宏控件

进度条

二维码

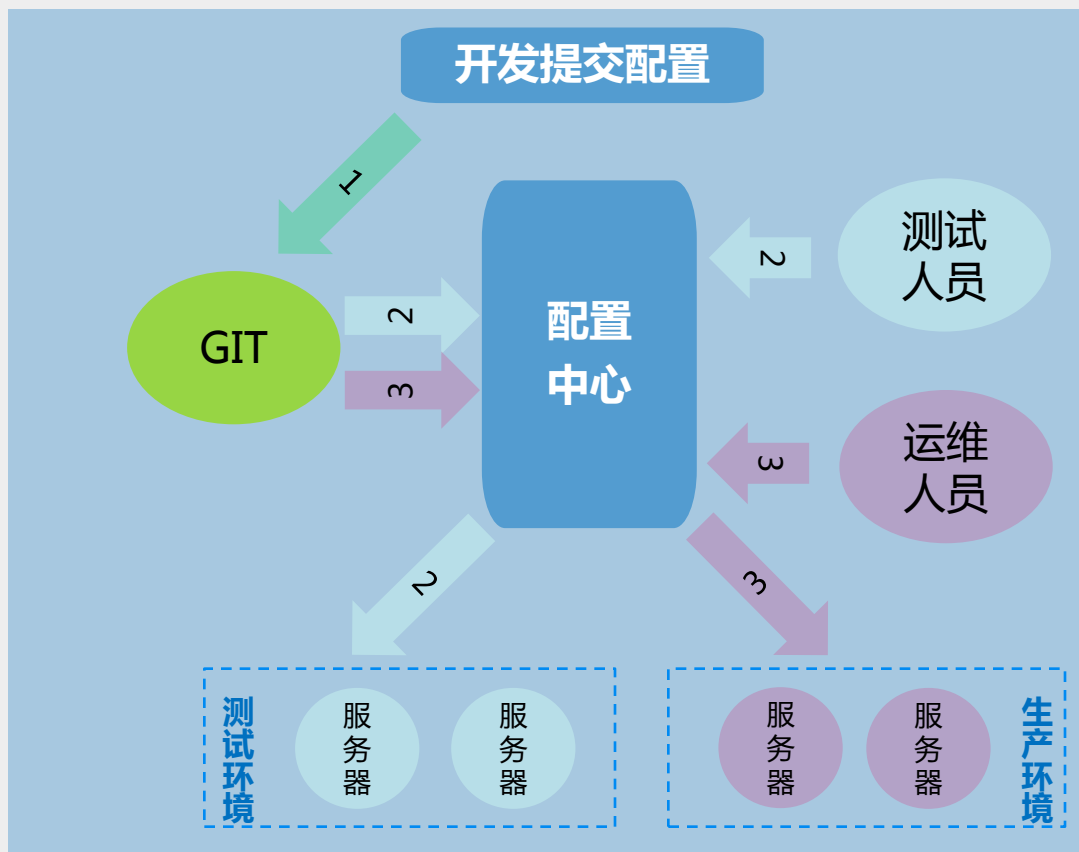
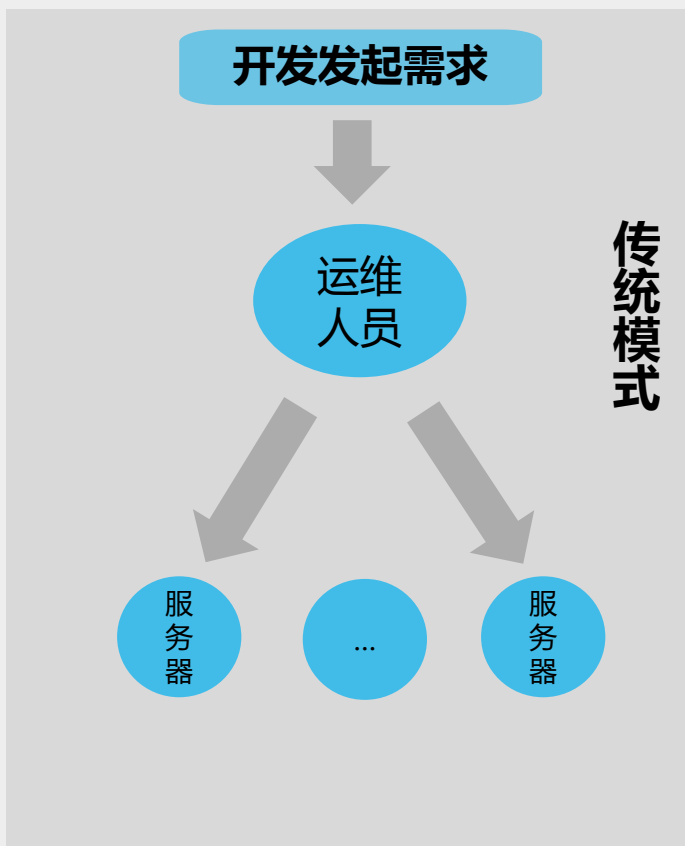
列表控件

表格控件

一起参与...

确定

# 流程化配置



提升操作效率、有效控制风险

# 平滑发布

The image shows a software interface for traffic control. On the left, a window titled "添加流量管控" (Add Traffic Control) displays a table of IP addresses and their weights. A blue box labeled "1 选定应用机器" (Select application machine) points to the IP list. A red circle highlights the "ms-demo" application selection, with a blue box labeled "2 阻断请求 (流量)" (Block request (traffic)) pointing to it. A blue box labeled "3 设置规则" (Set rules) points to the "HTTP限流" (HTTP throttling) section. A red circle highlights the "调节权重" (Adjust weight) dropdown, which is currently set to "关闭流量" (Close traffic) with a weight of 0. On the right, a "修改应用规则" (Modify application rules) dialog box is open, showing configuration fields for ContextPath, API version, and JSON rules. Below these fields is a table for defining logical rules with columns for logical operations, keywords, operators, and matching values. At the bottom right of the dialog are "关闭" (Close) and "提交更改" (Submit changes) buttons.

**1 选定应用机器**

**2 阻断请求 (流量)**

**3 设置规则**

添加流量管控

应用选择: ms-demo

HTTP限流

IP地址	权重	版本
7.110:8080	90	2.0.0
202.7.133:8080	90	2.0.0
10.202.7.143:8080	90	2.0.0

调节权重: 关闭流量 0

修改应用规则

ContextPath: \*/express

API版本号: 3.0

既存规则Json: ["@", "\$distcode", ["17", "18"]]

新建规则:

- And优先级比Or高-->A and B or C and D就是 (A and B) or (C and D)。
- In中的内容请用[英文逗号<, >]分开。
- 单/双引号代表字符串。

逻辑运算	关键词	运算符	匹配值	
		>		+
And		=		-

关闭 提交更改

扩宽发布窗口、快速试错

# 自动化部署

## 测试环境

- 系统发布
- 系统管理
- 系统设置
- 密码管理

系统 子系统清单

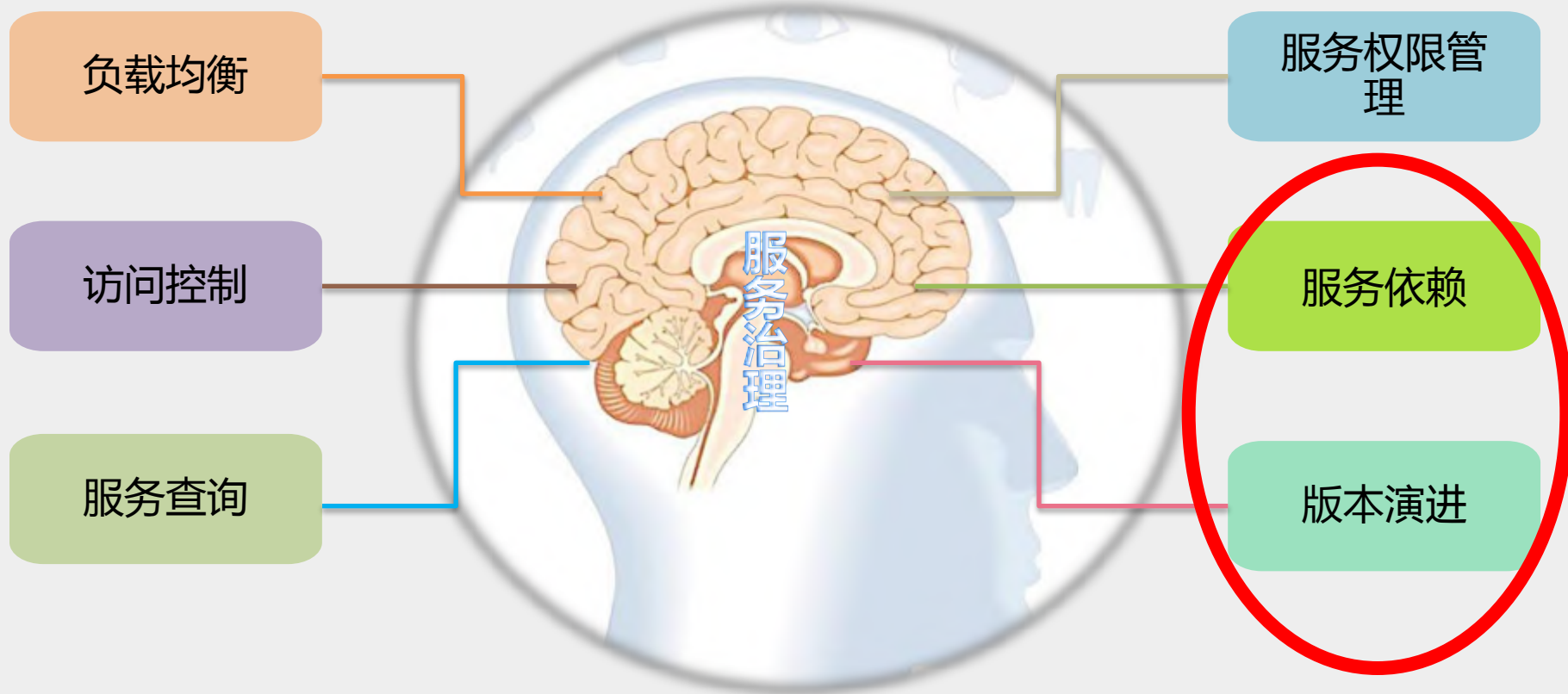
系统 : INC-SGS-CORE

C	<b>ccsp-app-interface</b> 接口系统	可操作环境 : sig环境
C	<b>COMMON-APPMANAGER</b> 资源服务,common-appmanager	可操作环境 : sig环境
G	<b>GATEWAY-API</b> api网关	可操作环境 : sig环境
G	<b>GATEWAY-LUA</b> 外部网关-LUA	可操作环境 : sig环境
G	<b>GATEWAY-LUA-LVS</b> 短链接网关	可操作环境 : sig环境
G	<b>GATEWAY-SYNC</b> 同步(内部)网关微服务,gateway-sync	可操作环境 : sig环境
G	<b>GEO-UNITAREA</b> 位置微服务,geo-unitarea	可操作环境 : sig环境
G	<b>GEO_LOCATION</b> 位置微服务,geo-location	可操作环境 : sig环境
G	<b>GOVERNANCE</b> 服务治理平台,governance	可操作环境 : sig环境



# 服务治理

服务治理：业务微服务管控中心



# 直观感受

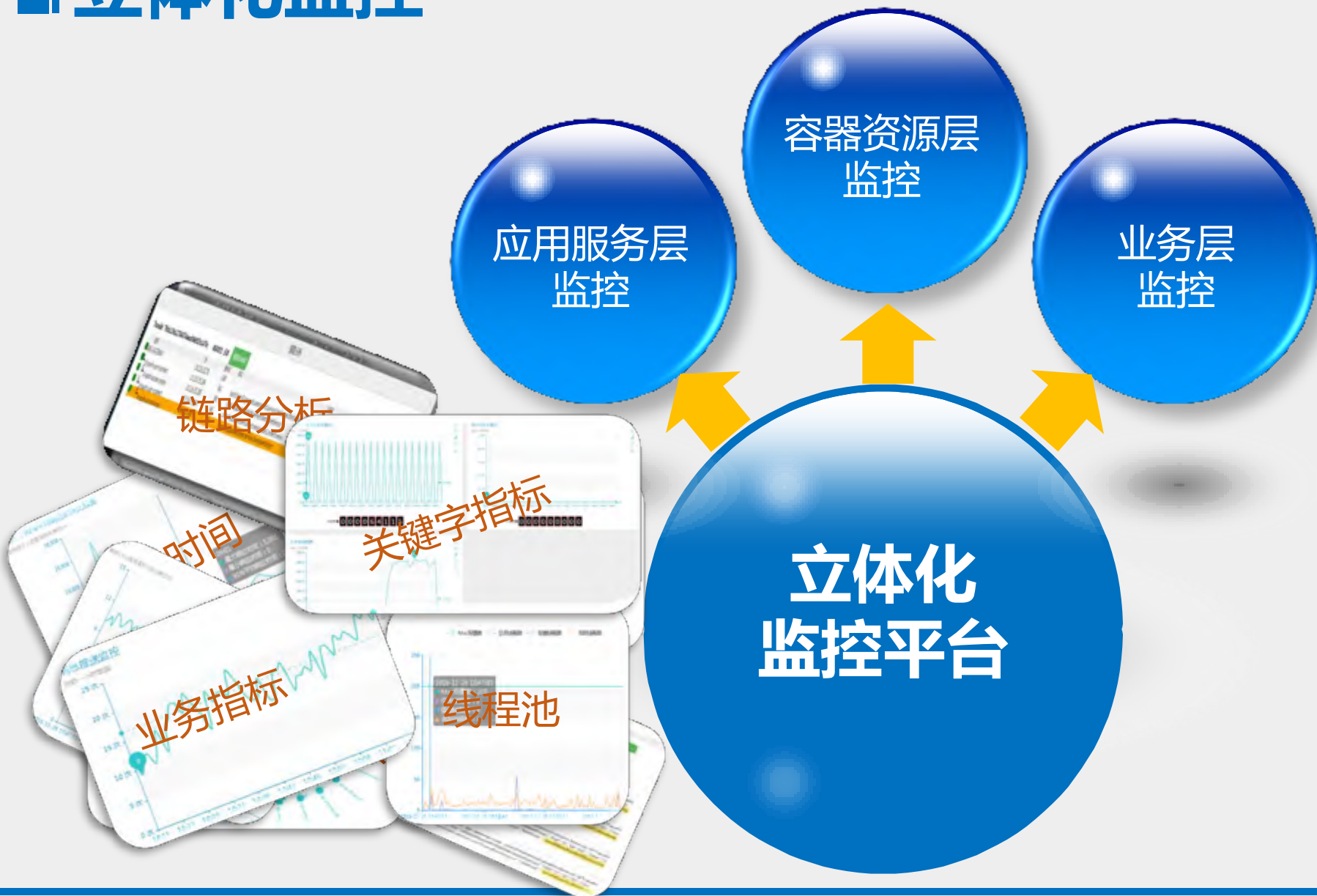
## 服务依赖



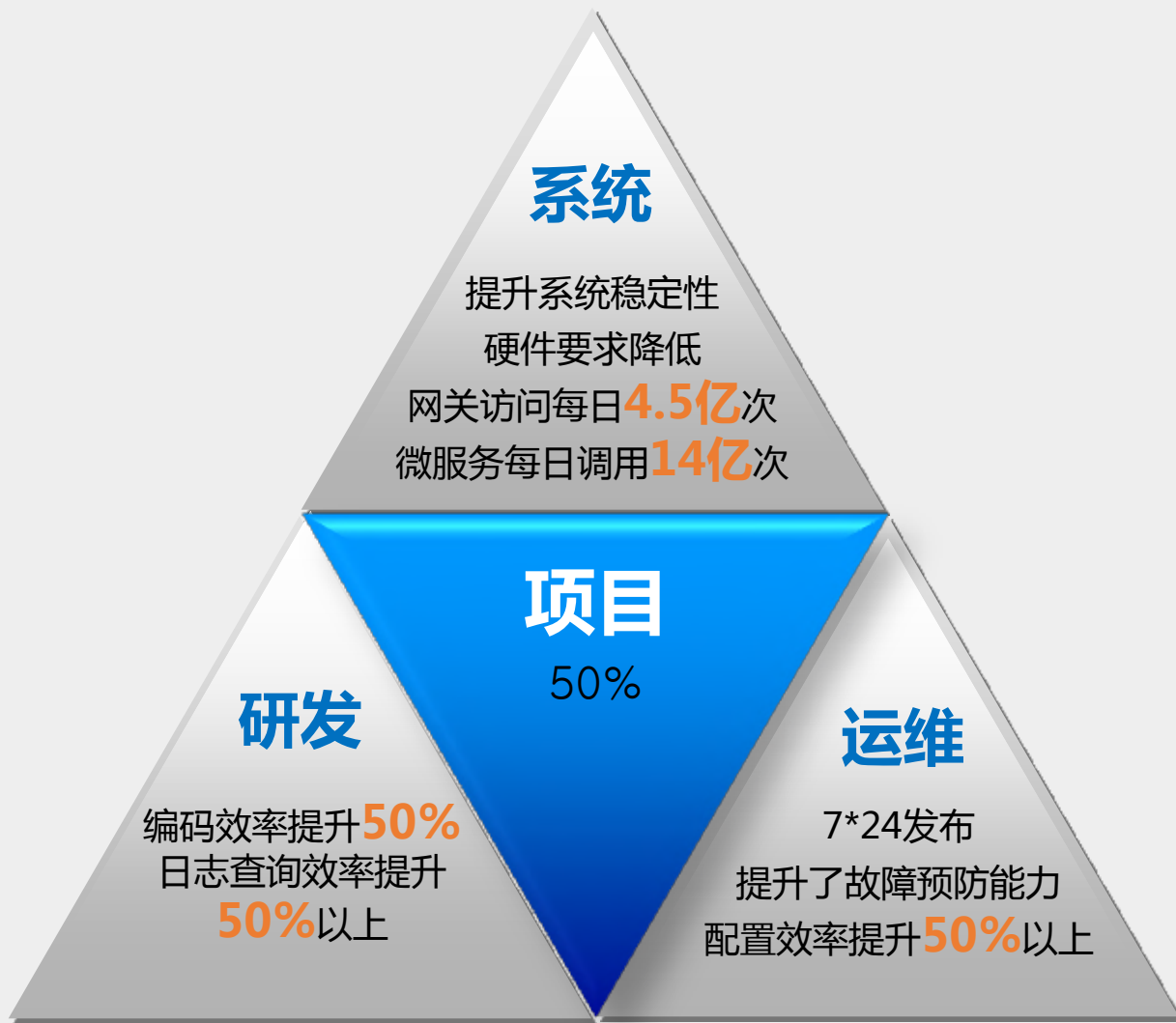
## 版本演进



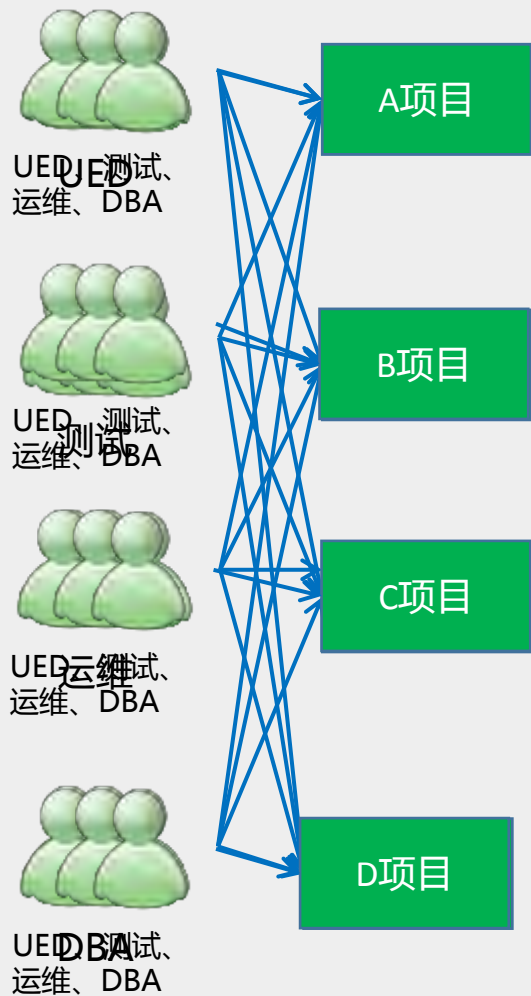
# 立体化监控



# 实际使用效果



# 敏捷和项目管理



微服务架构同时还是一个**组织原理**的体现，这个原理就是**康威定律 ( Conway's Law )**，Melvin Conway在1968年指出：“Any organization that design a system(defined broadly) will produce a design whose structure is a copy of the organization's communication structure”，翻译成中文就是：**设计系统的组织，其产生的设计和架构等价于组织间的沟通结构**。Dan North对此还补充说：“Those system then constrain the options for organization change”，**简言之，这些系统在建成之后反过来还会约束和限制组织的改变**。

大量小团队带来的管理难题？

敏捷

- 纪律性
- 自组织学习型团队
- 鼓励沟通
- 做减法

项目管理

- 大量的文档
- 严谨的流程
- 协调资源
- 风险把控

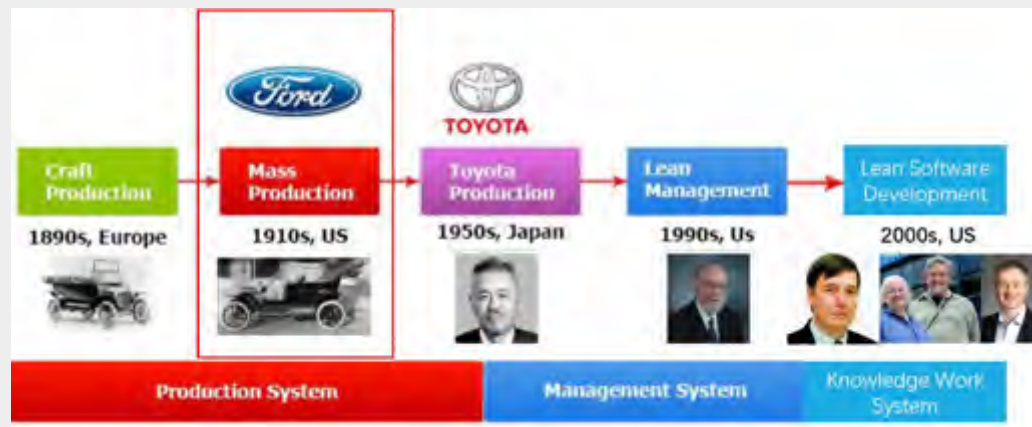
# 自然进化



自然是使用一种低成本试错的方式在演进。

微服务架构的妙处就在于它符合自然演进规律，提供低试错成本的演进基础。

精益



**感谢聆听**