讯飞输入法Android架构演进与实践

程坤
科大讯飞

大纲

Part1

架构演进历程

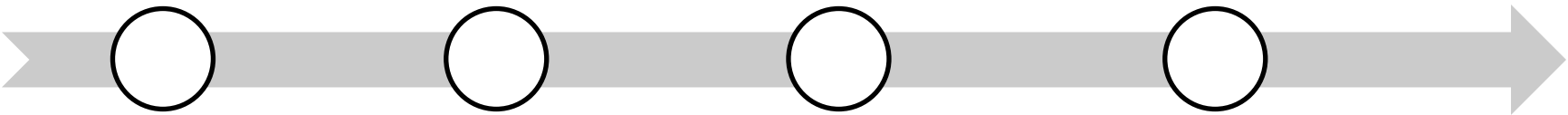
Part2

组件化架构实践

Part1

架构演进历程

架构演进概览



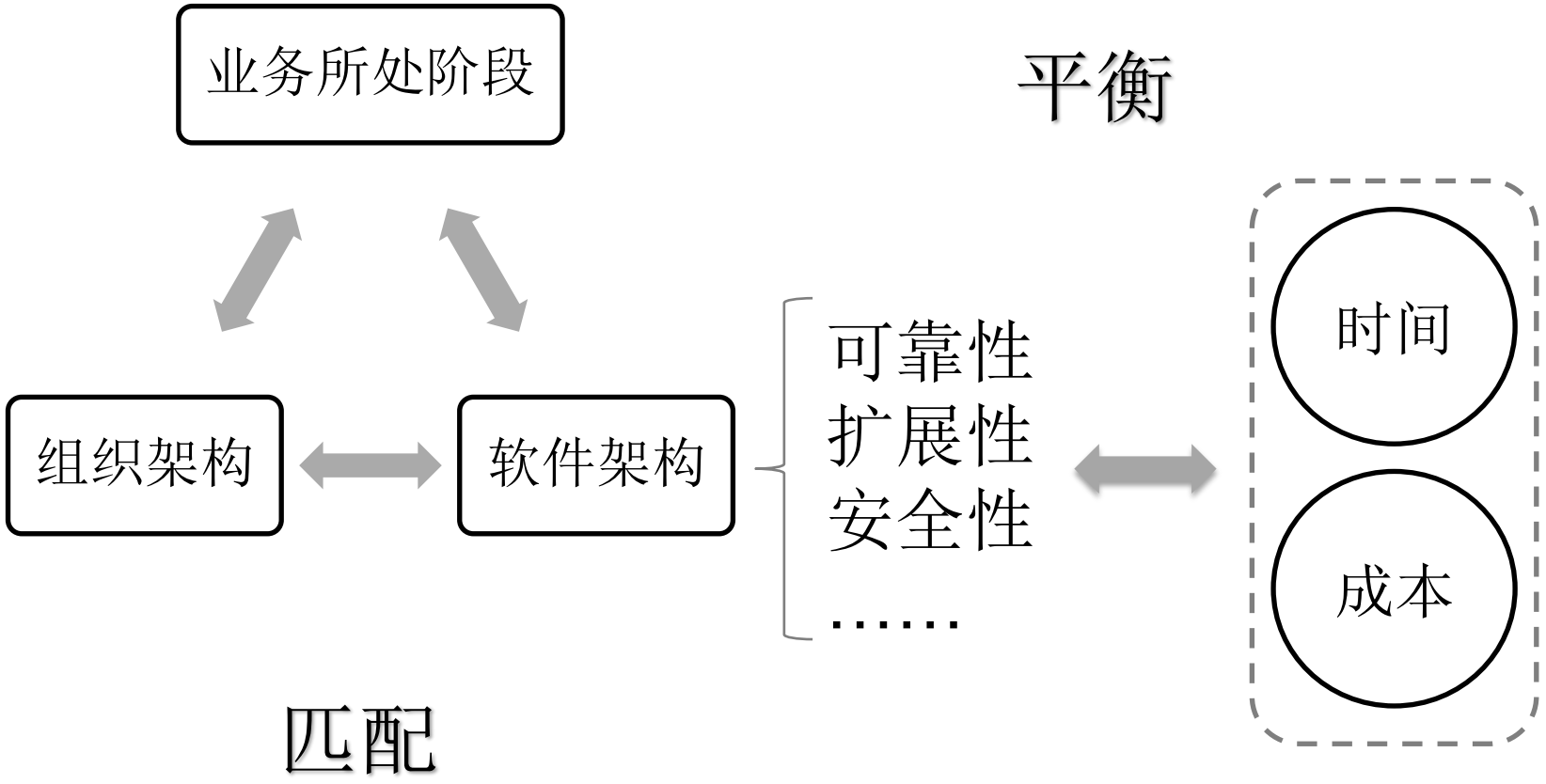
2010.7
简单MVC

2012.3
简单分层

2014.12
多进程

2015.12
组件化

架构如何演进



产品初期

挑战：

- 时间紧，从无到有只有三个多月
- 无参考，很多功能都是首创
- 人员少，客户端开发仅有2人

语音云

2010.7 启动

2010.10 发布会

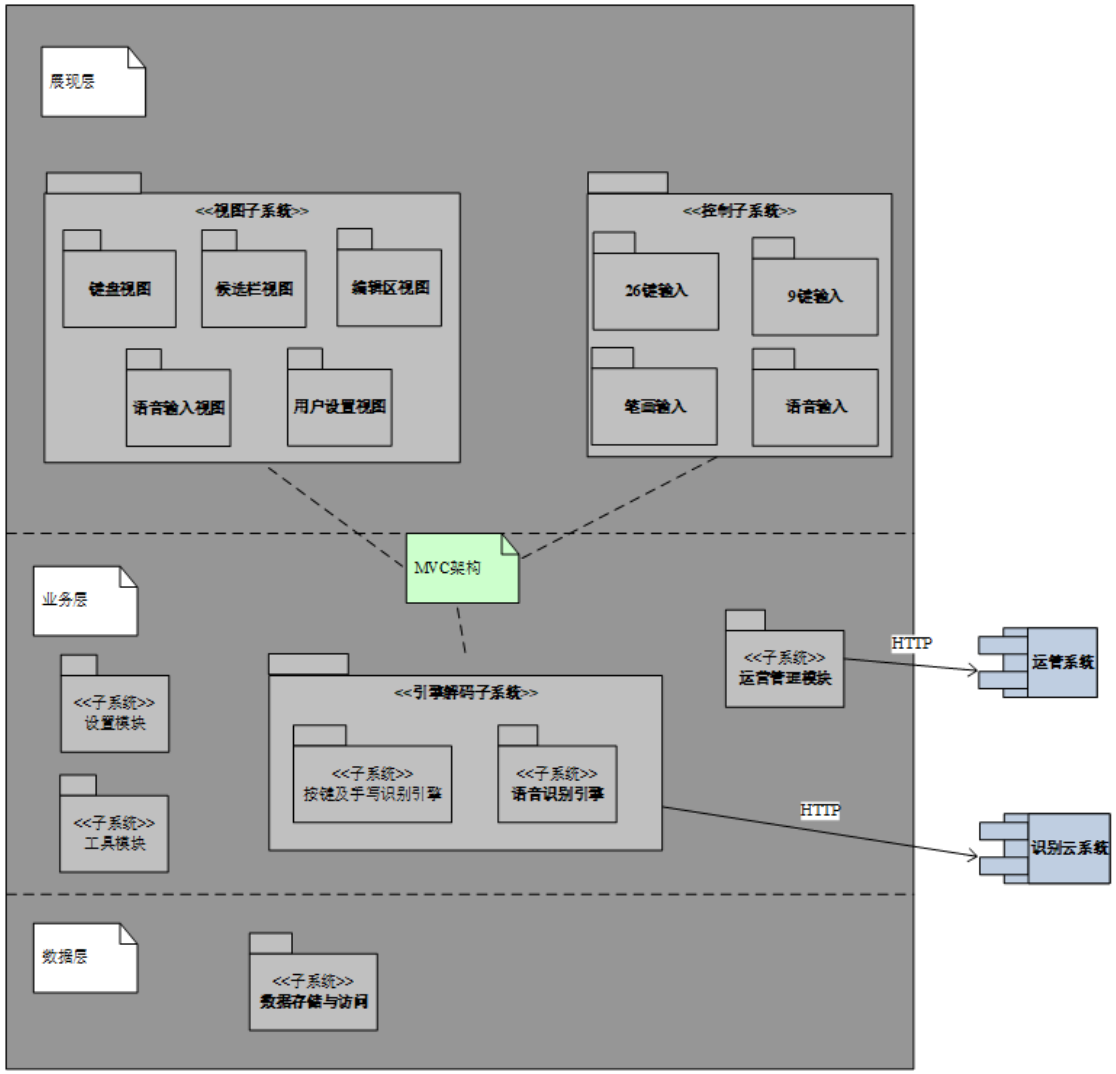
简单MVC

关注点

- 快
- 稳定

经验

- 尽量复用



产品快速发展前期

业务

- 补齐功能
- 优化效果

团队

- 开发4人
- 扁平化组织

问题

- 各写自己的代码
- 代码重用性差
- 新功能开发成本高
- 维护起来困难

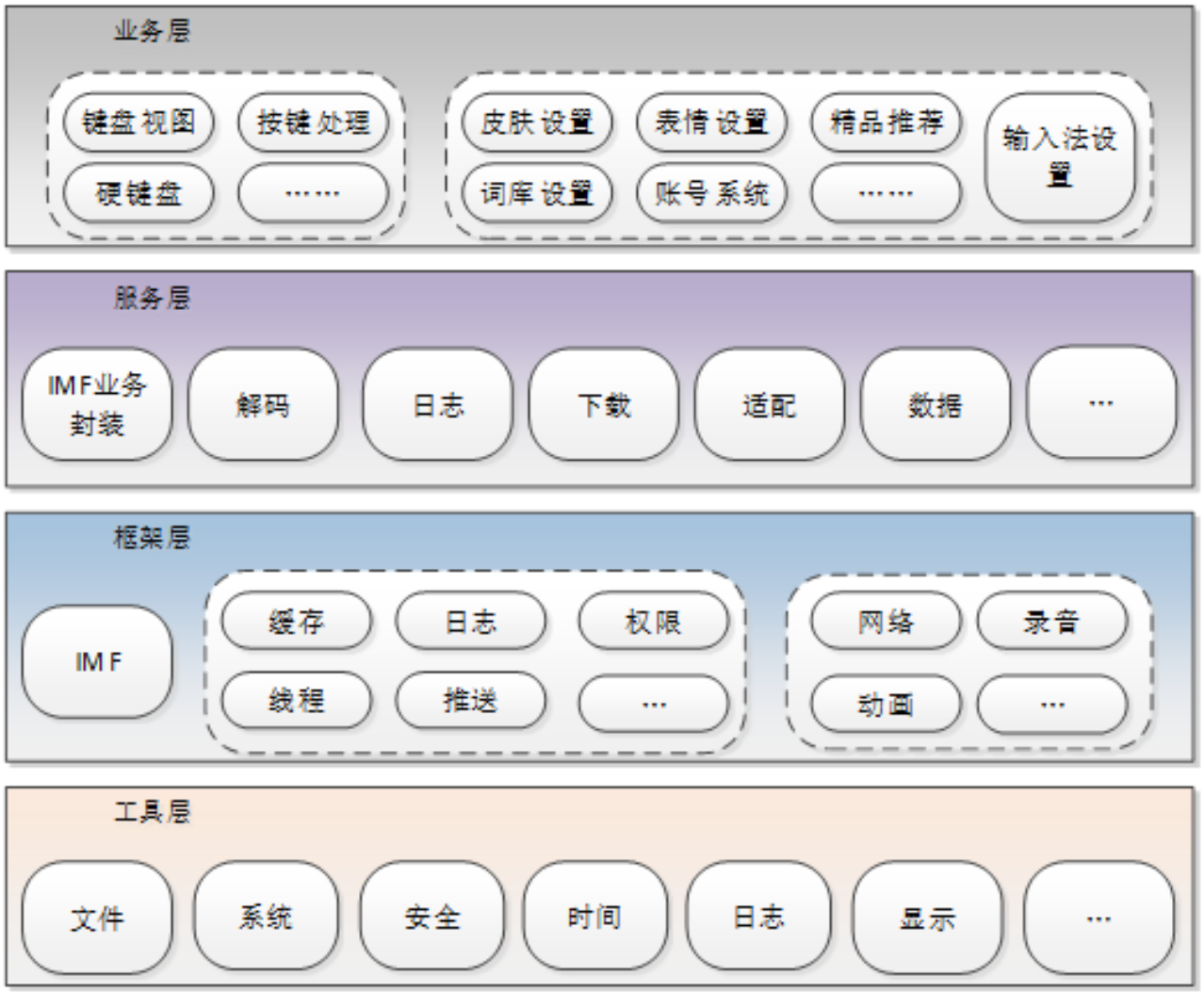
分层架构

关注点

- 开发效率
- 提供复用

经验

- 梳理和优化
- 封装和推广



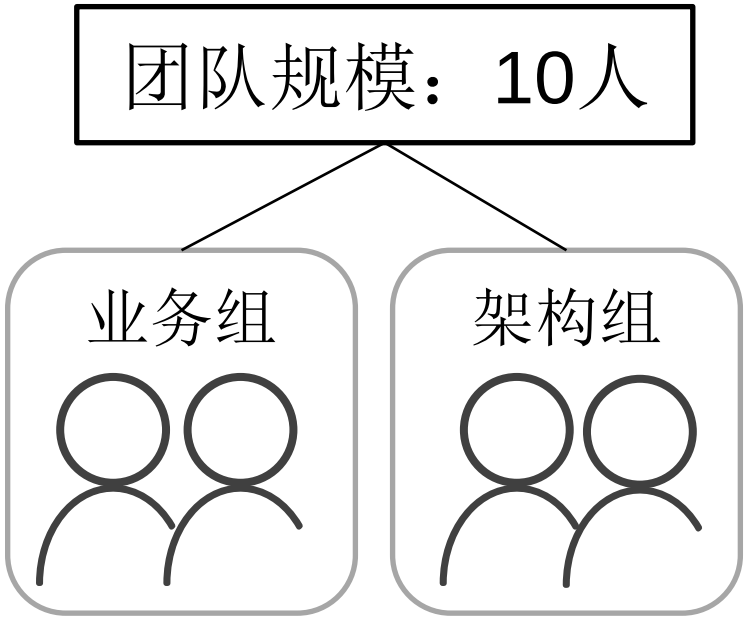
产品快速发展中期

产品迭代

- **功能越加越多**
- **代码越来越复杂**

形成制约：
性能和稳定性

团队结构调整



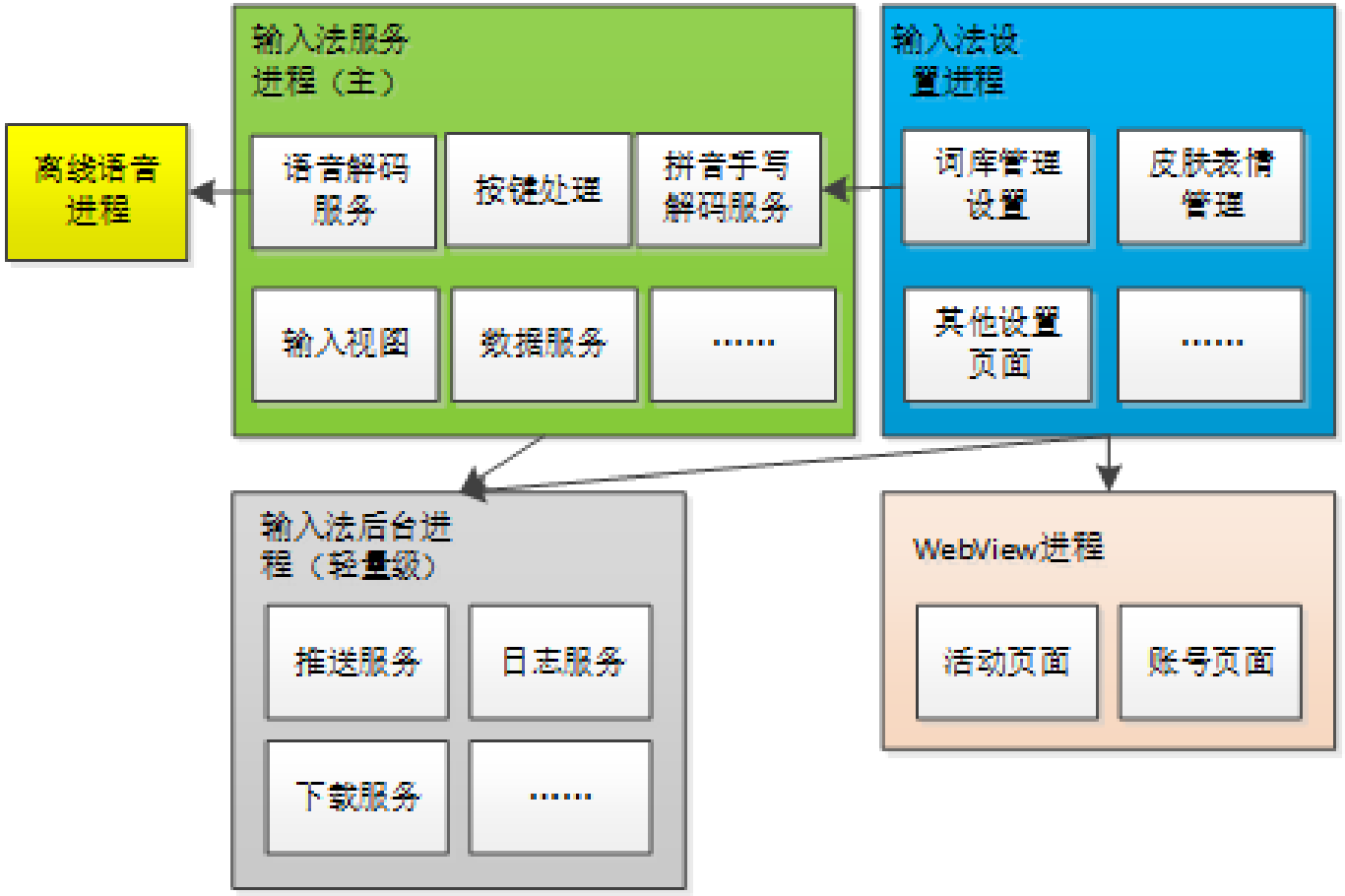
多进程架构

关注点

- 即用即走
- 隔离
- 轻量

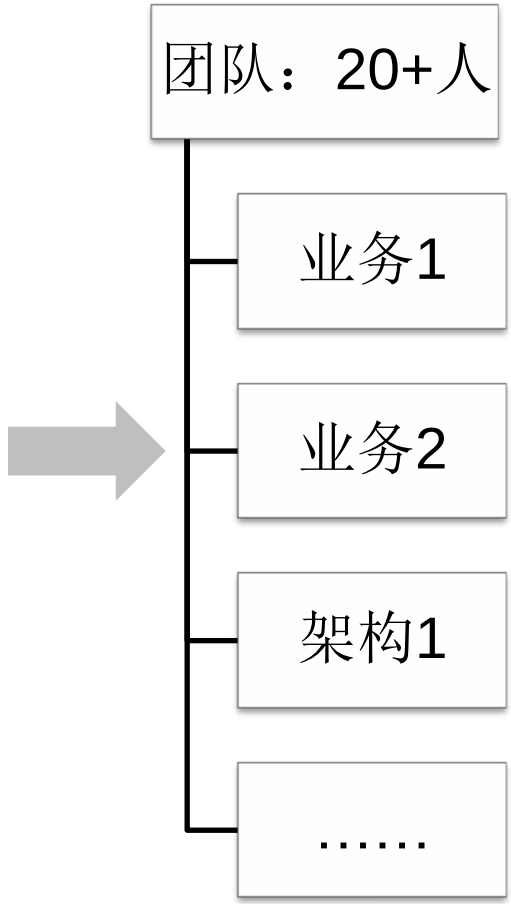
经验

- 摒弃单例
- 简化调用



产品快速发展后期

业务：
- 业务分工更细
- 不断进行新尝试
- 快速市场验证



架构上支持更快的产品迭代

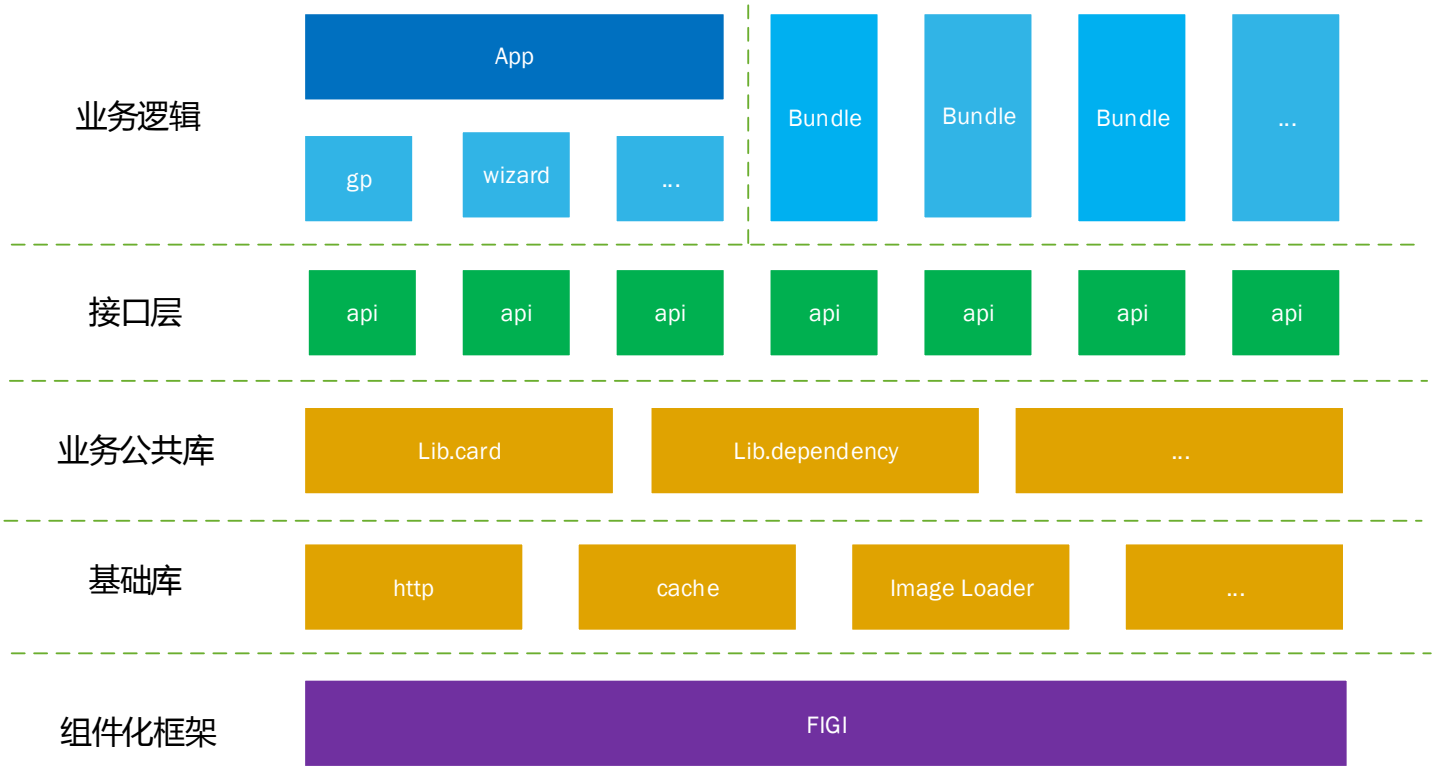
组件化架构

关注点

- 并行开发
- 动态更新

经验

- 坑多慎入



Part2

组件化架构实践

为什么要重复造轮

较早

- Atlas/ACDD
- DynamicLoadApk
- DynamicApk
- Small

较新

- VirtualAPK
- Replugin

输入法业务有其独特性

填坑之路

- 兼容性
- 启动性能
- 多进程
- 组件更新
- 工程结构
- ……

兼容性

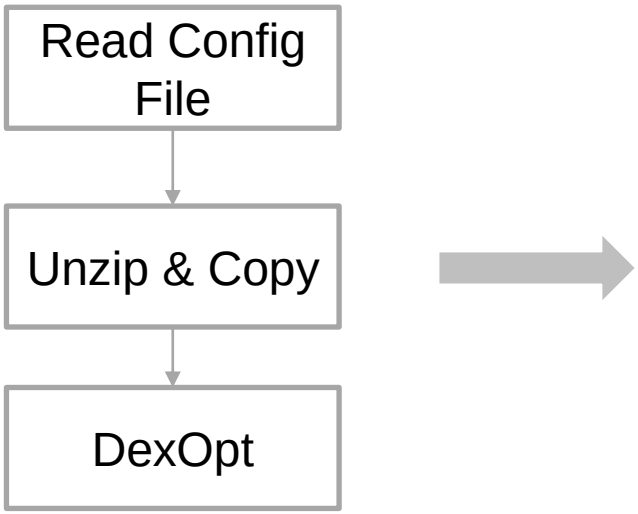
- ActivityThread
- Instrumentation
- Resources
- ContextImpl

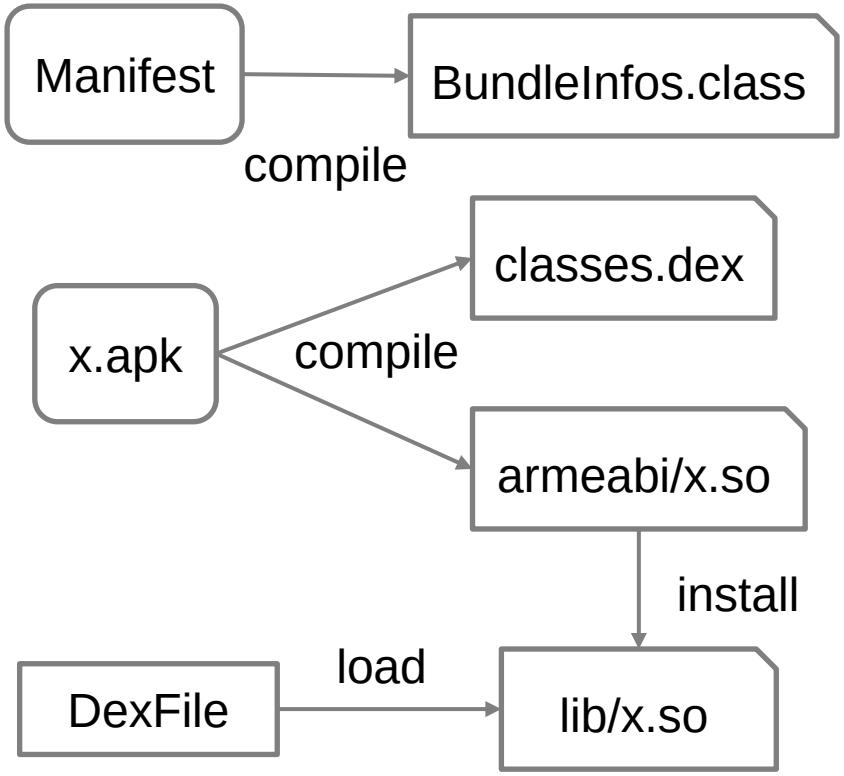
参考Replugin（进行中）

减少Hook点

启动性能

启动时I/O

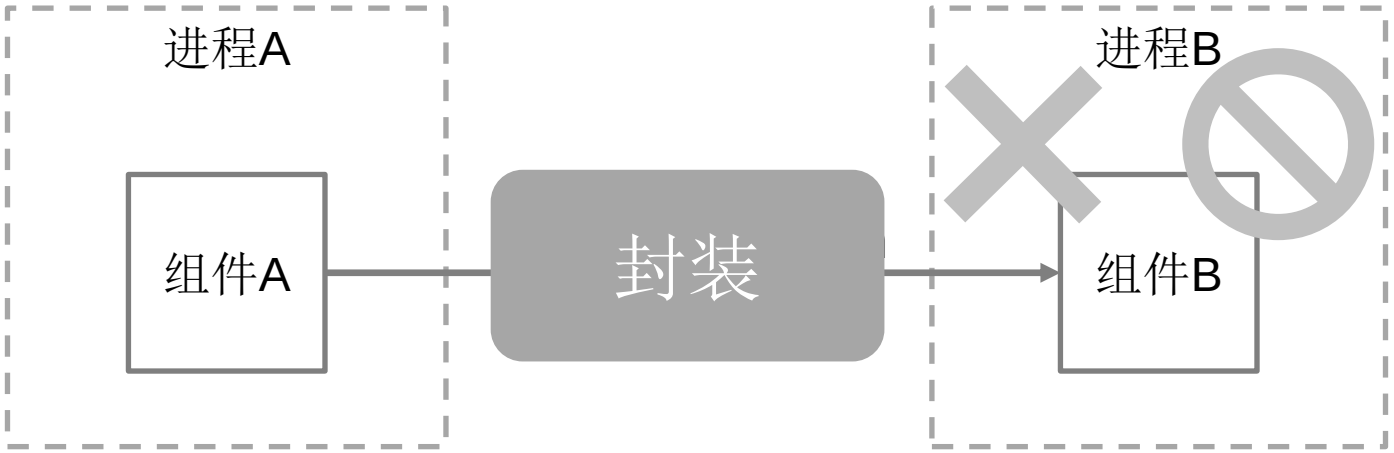




问题：

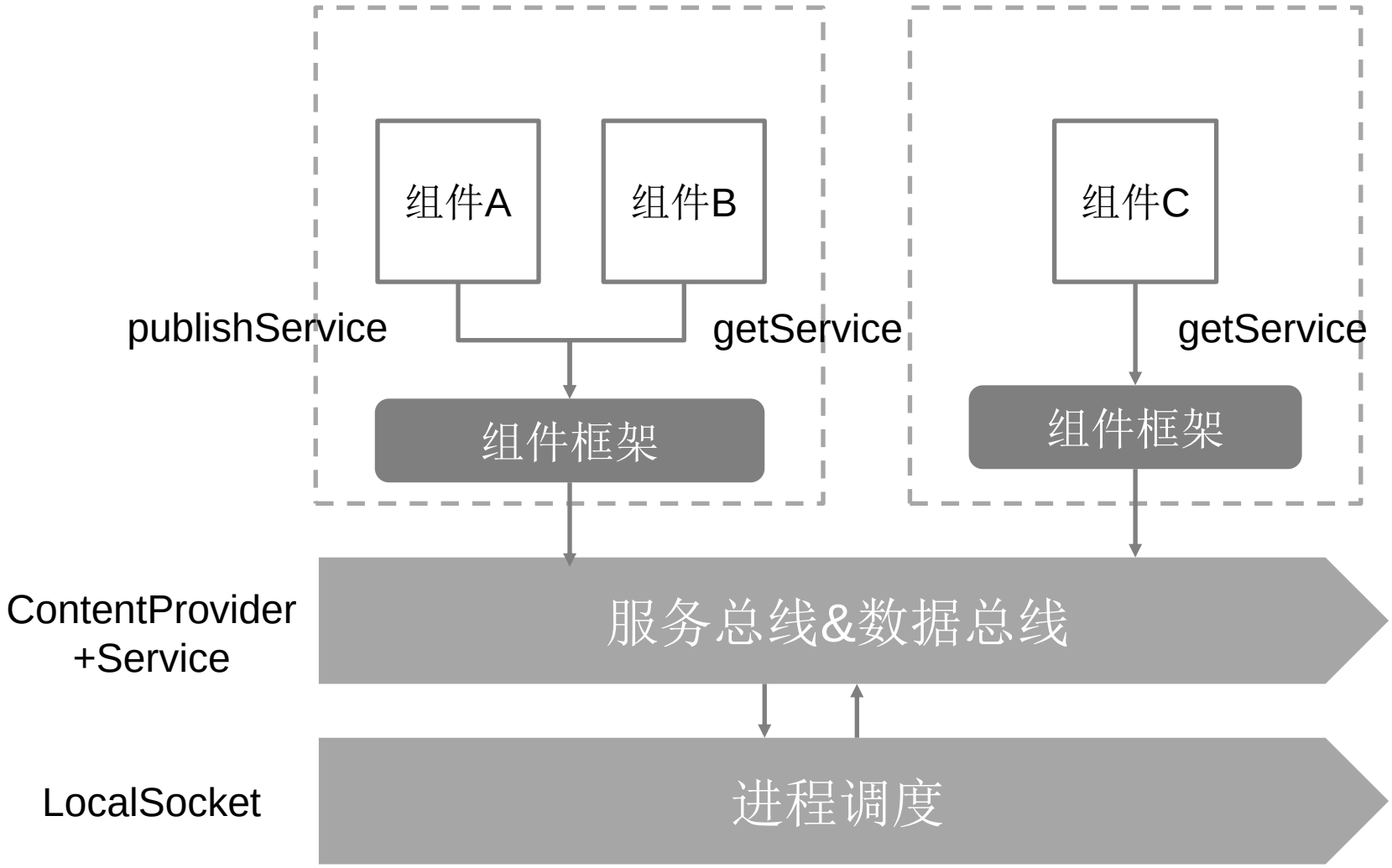
- 键盘启动变慢
- 空间不足导致崩溃

多进程

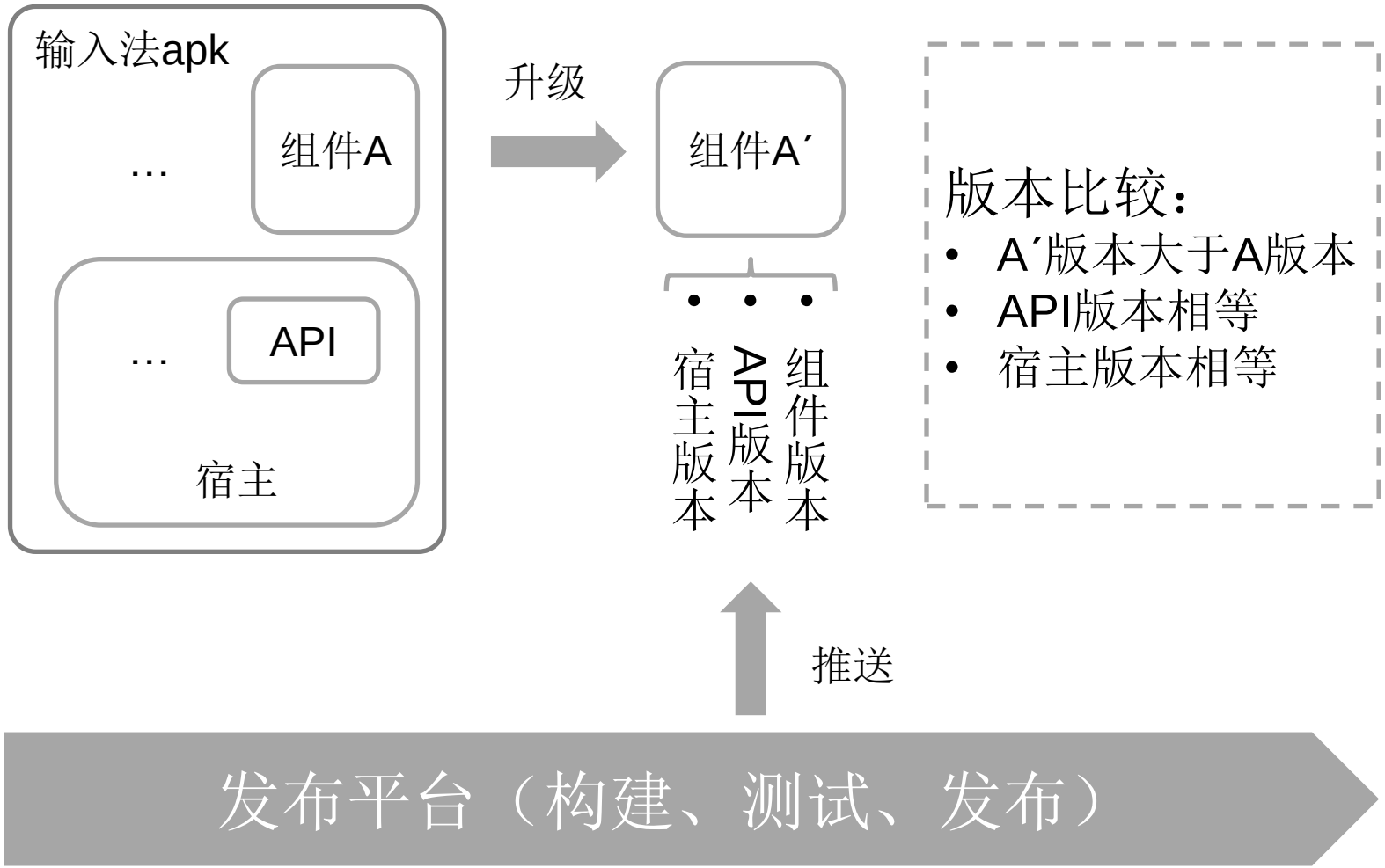


- 像用本地能力一样使用跨进程的能力
  - 接口转换、进程拉起、重连和状态恢复
- 保证各组件能力能正常运行
  - 组件自适应选择运行进程

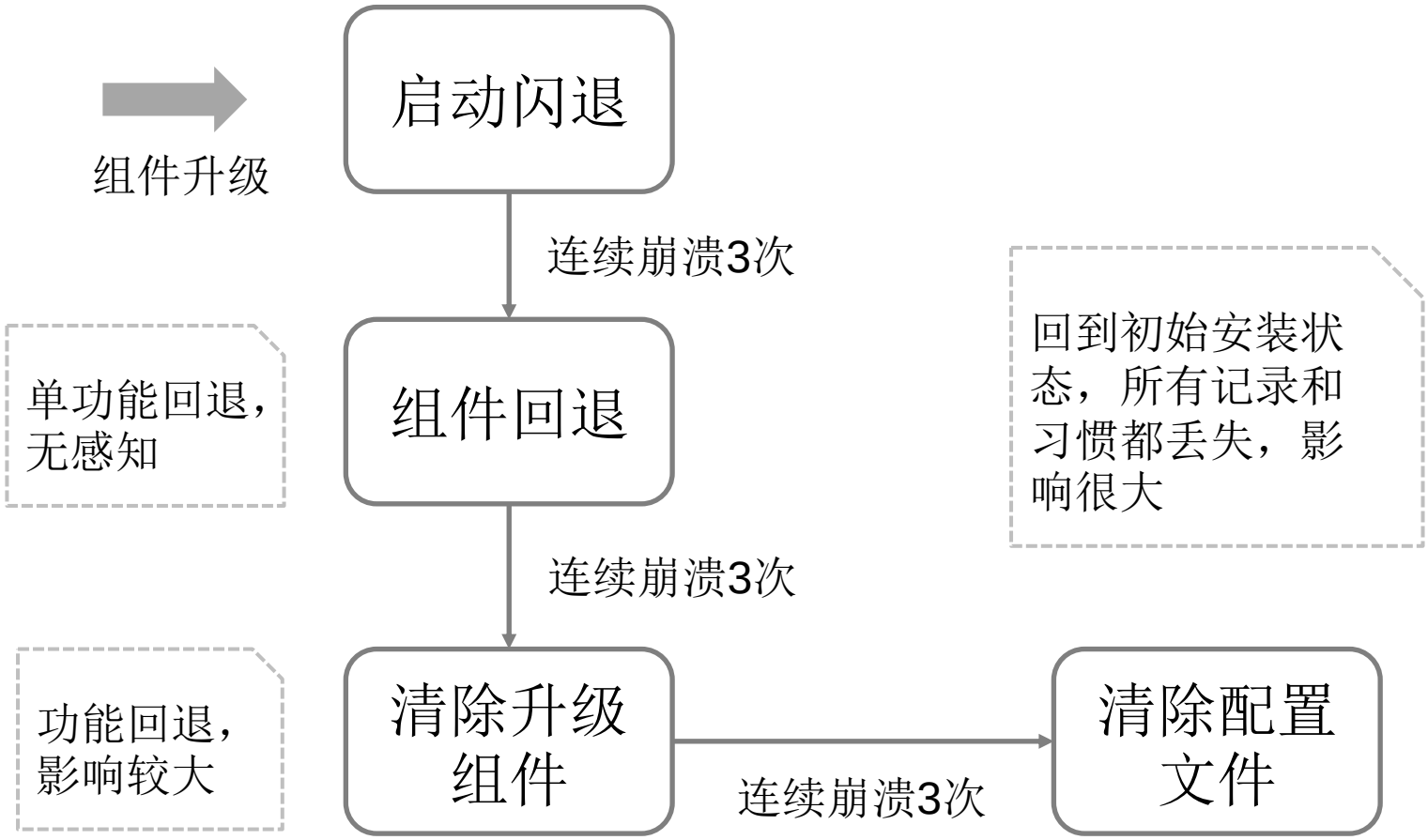
多进程


组件A
组件B
组件C
publishService
getService
getService
组件框架
组件框架
ContentProvider
+Service
服务总线&数据总线
LocalSocket
进程调度

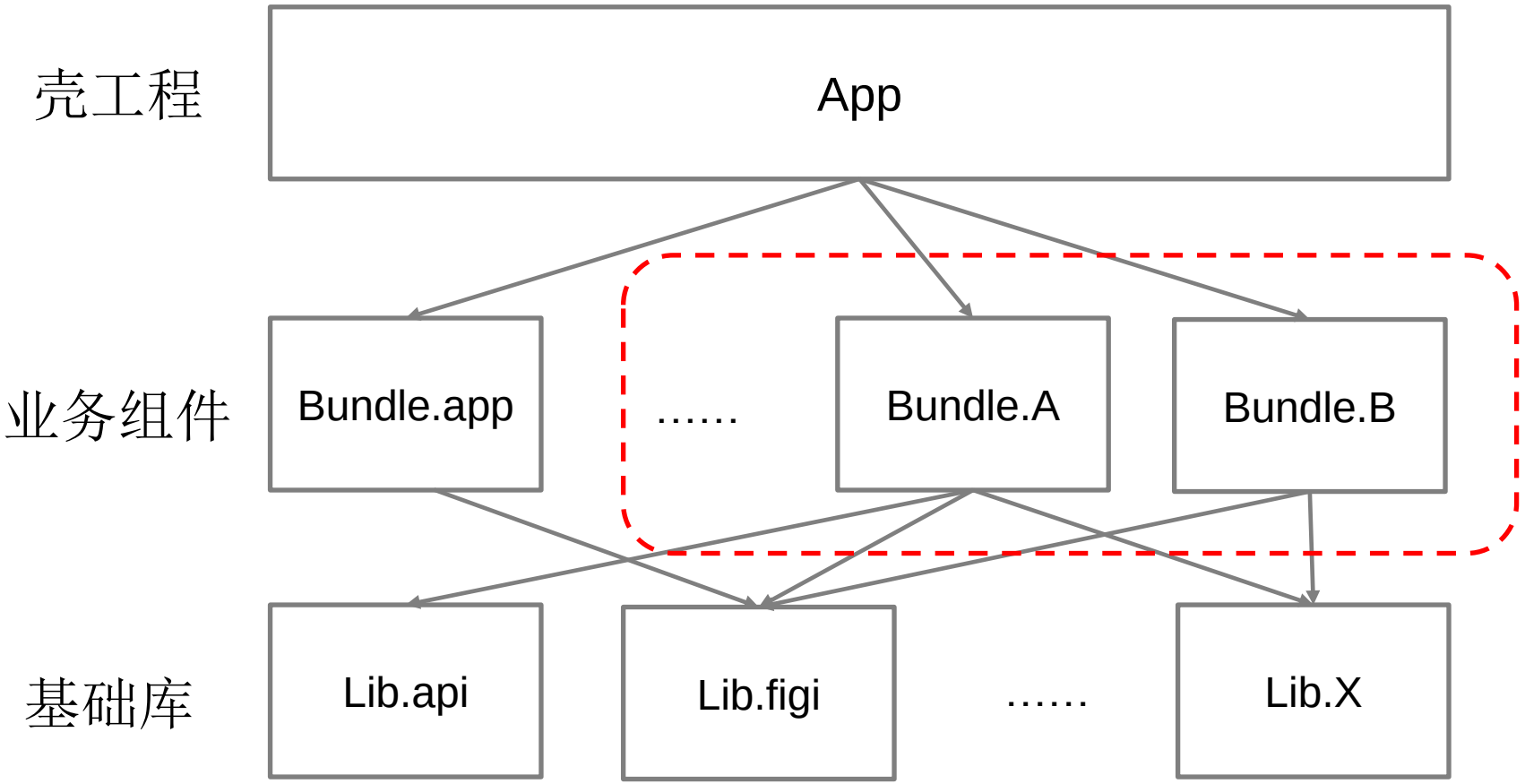
组件更新


输入法apk
组件A
…
API
…
宿主
升级
组件A´
宿主版本
API版本
组件版本
版本比较：
• A´版本大于A版本
• API版本相等
• 宿主版本相等
推送
发布平台（构建、测试、发布）

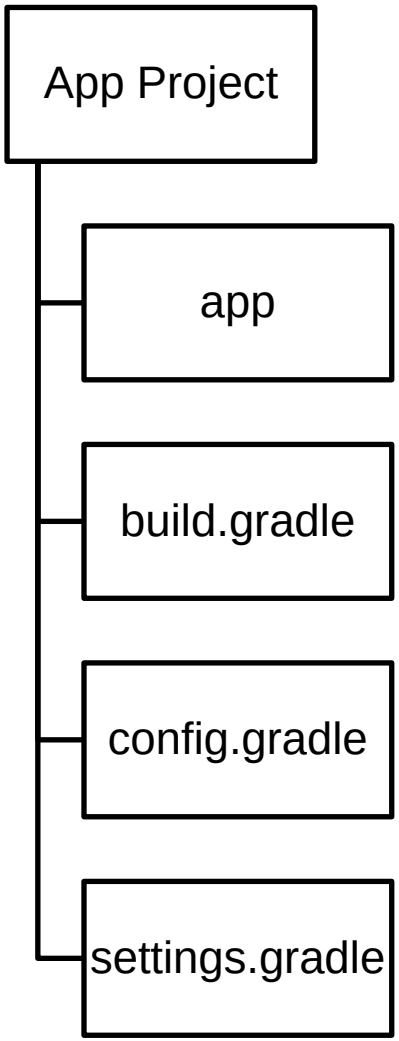
组件更新


组件升级
启动闪退
连续崩溃3次
单功能回退，无感知
组件回退
回到初始安装状态，所有记录和习惯都丢失，影响很大
连续崩溃3次
功能回退，影响较大
清除升级组件
连续崩溃3次
清除配置文件

工程结构


壳工程
App
业务组件
Bundle.app
……
Bundle.A
Bundle.B
基础库
Lib.api
Lib.figi
……
Lib.X

工程结构



特点：

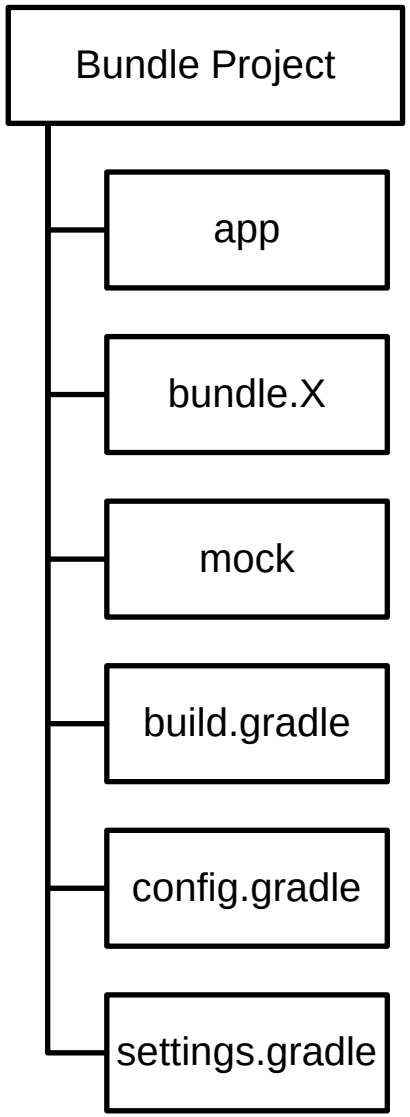
- 无任何代码，仅有脚本及配置文件
- 仅有唯一分支，一般不做修改
- 仅集成打包、集成调试时使用
- 不使用repo、git submodule

组件/库列表：

- 仓库地址
- 本地路径（可缺省）
- 版本号
- 集成模式（三种）

gradle plugin

工程结构



特点：

- 可独立编译调试
- 打包产物有：测试apk、组件apk和aar
- 产物上传到Nexus私服

- 仓库地址
- 包名、版本号、进程、加载模式
- 依赖库列表（地址、版本）
- 开发模式（三种）

正在进行的工作

组件持续拆分

业务兼容处理

我们的路还很长

增加监控纬度

数据分析优化

结束

欢迎交流