大搜车 ReactNative 开发集成流程体系演变

芋头@大搜车无线团队

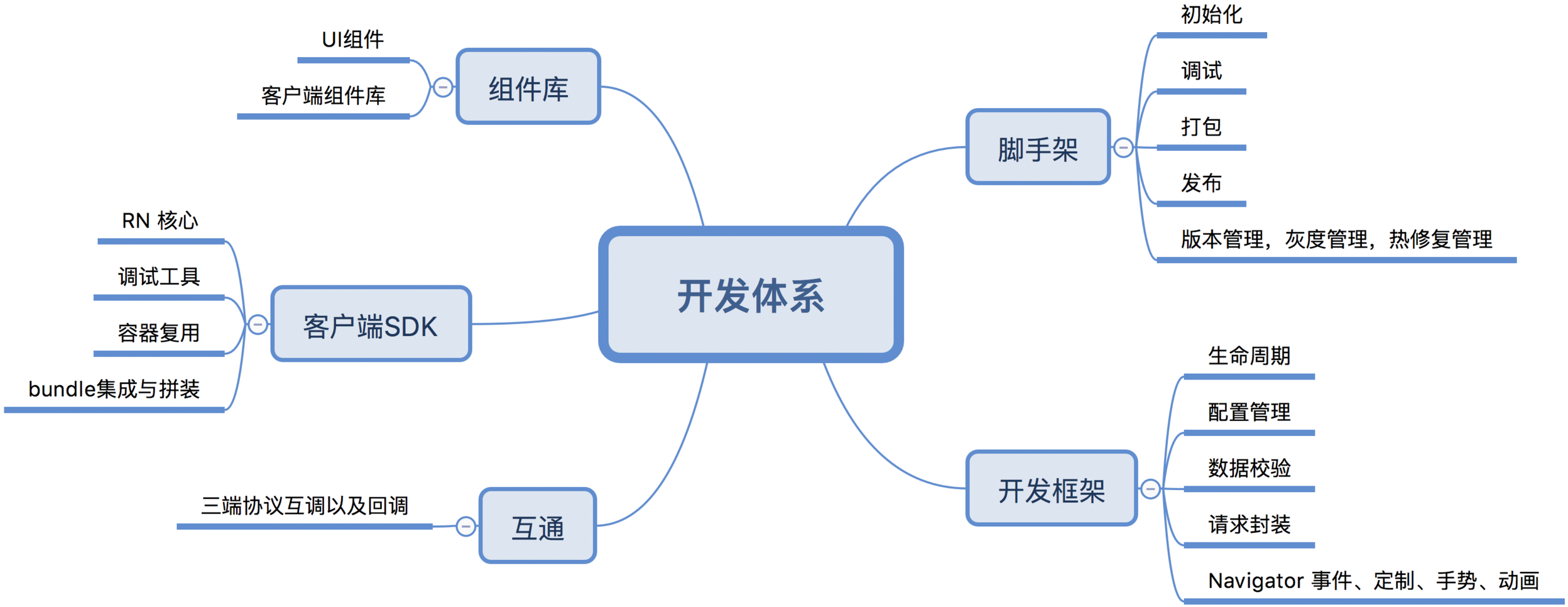
一、场景

场景

- 重业务逻辑，轻营销。更重要的是可控性而不是灵活。
- 核心解决客户端人力膨胀问题，主要由 Native 开发，需尽量尊从其习惯。
- 公司内多个 app，以及合作方的 app，业务繁多。
- 完全耦合进 Native，模糊三端概念，互通互调。

战略级，没有退路

三、体系简介


开发体系
组件库
UI组件
客户端组件库
客户端SDK
RN 核心
调试工具
容器复用
bundle集成与拼装
互通
三端协议互调以及回调
脚手架
初始化
调试
打包
发布
版本管理，灰度管理，热修复管理
开发框架
生命周期
配置管理
数据校验
请求封装
Navigator 事件、定制、手势、动画

三、开发与集成

最初版

当时核心需要解决的问题

- **实现热更新**
- **版本锁定**
- **三端互通**
- 业务解耦，业务之间互相独立
- 发版时带上最新版的 bundle

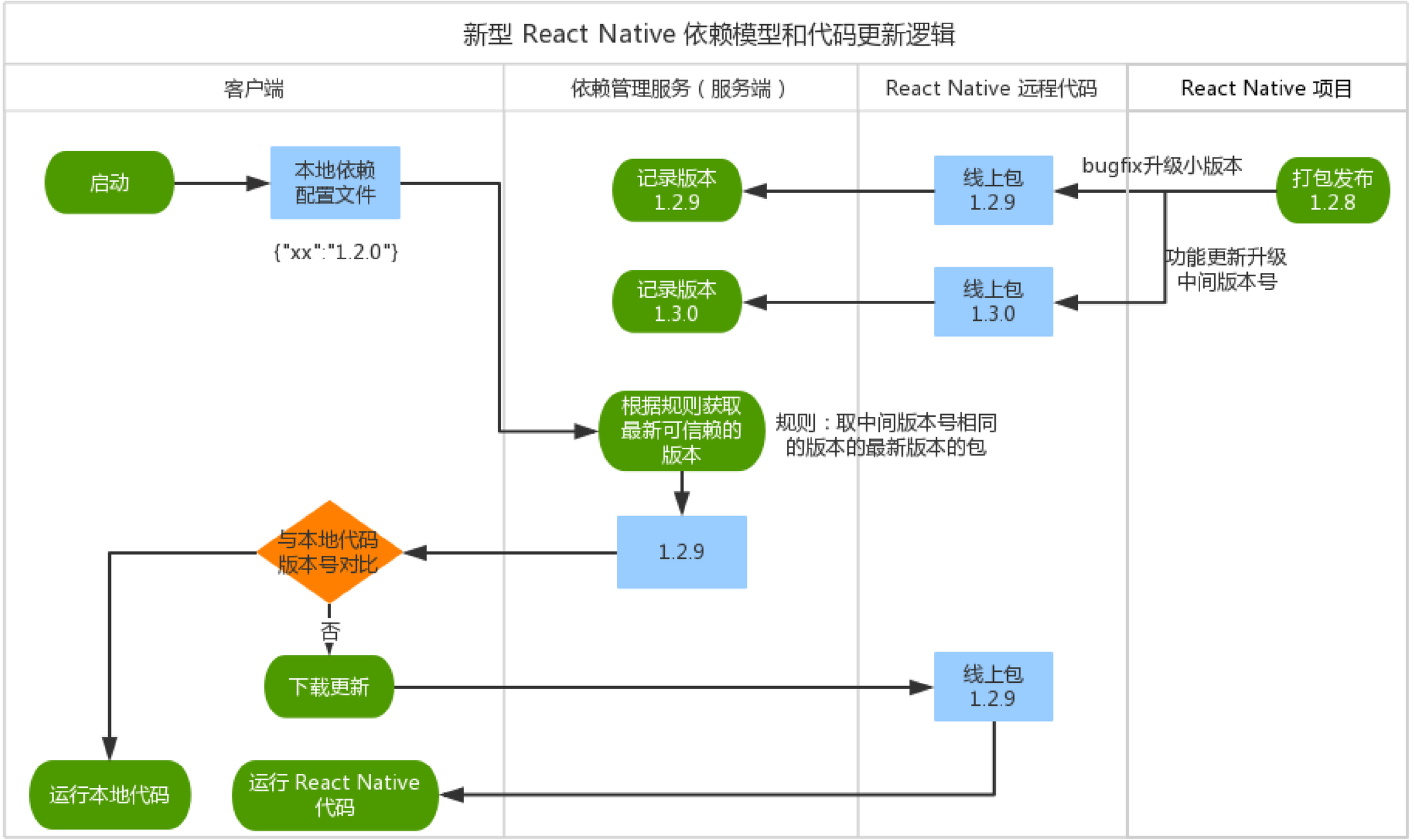
版本锁定

- 大量的跨端依赖，端与RN的版本可能不兼容。
- RN 官方包 需要可以随时升级，因为不稳定因素很多。
- SDK 可以不断迭代功能，而不影响线上业务。

SRN-HUB 版本管理服务

最初思路

- 在服务端判断请求来自哪个版本的app
- 服务端维护一个关系：哪个app的哪个版本可以更新哪个版本的bundle
- 根据这个关系，返回是否需要下载某个 bundle
- 太复杂的架构不是好架构，要对使用者尽量透明


新型 React Native 依赖模型和代码更新逻辑
客户端
依赖管理服务（服务端）
React Native 远程代码
React Native 项目
启动
本地依赖
配置文件
{"xx":"1.2.0"}
记录版本
1.2.9
线上包
1.2.9
bugfix升级小版本
打包发布
1.2.8
功能更新升级
中间版本号
记录版本
1.3.0
线上包
1.3.0
根据规则获取
最新可信赖的
版本
规则：取中间版本号相同
的版本的最新版本的包
1.2.9
与本地代码
版本号对比
否
下载更新
线上包
1.2.9
运行本地代码
运行 React Native
代码

统一 Router

- 没有复杂的存储规则，所有规则都在版本号里。
- ${major}.${feature}.${patch}
- 版本号升级根据发布类型自动管理。
- 依赖声明在 app 工程内，由 app 自己决定依赖，而不是三方管理依赖。

问题

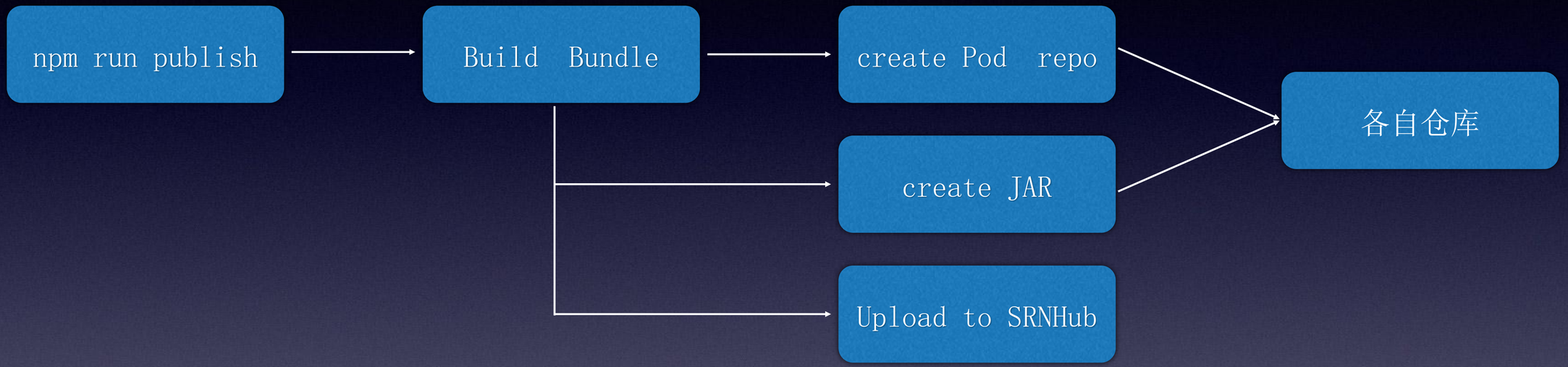
- 发布前需要RN开发修改APP内的依赖文件，这个文件和 podfile ，以及maven的依赖不在一起，客户端难以适应。
- bundle 包有环境的概念，在开发流程里不断切换环境的时候需要不断改依赖文件，很麻烦。
- 测试环境一律走覆盖式热更新，但是上线前会忘记修改依赖。
- 热更新被滥用，规则成为摆设。
- 打包时需要拉取依赖的bundle集成到本地，拖慢打包过程。

2.0 版

核心解决

- **将 bundle 编译成 Native 模块**
- **bundle 抽离公用模块**
- 扫码调试等辅助开发工具

app 发布流程

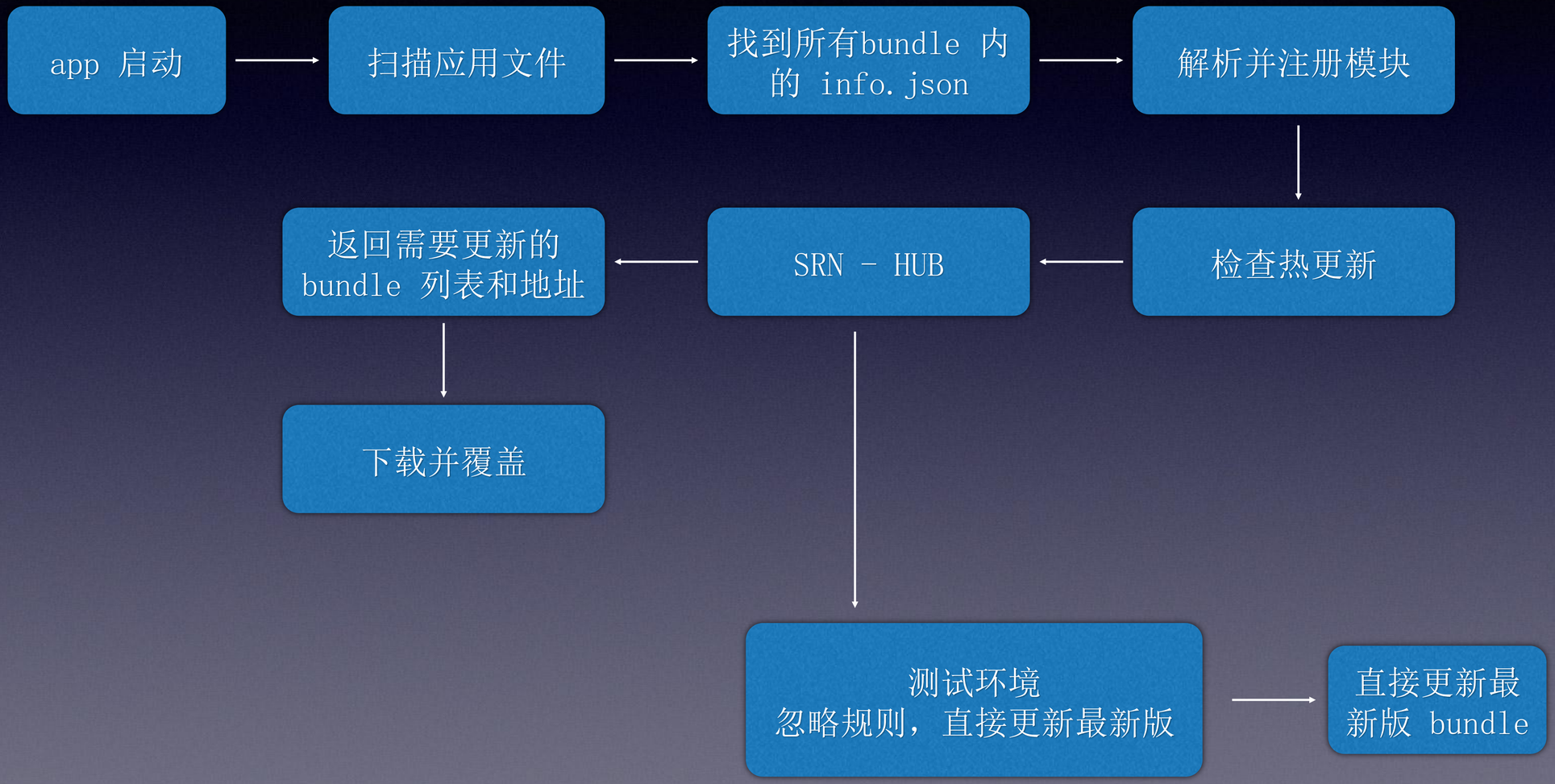


然后在原生工程里声明依赖及版本

```
def static androidRnDependency() {
    def androidDependency = [
            'com.souche.rn:DFCDirectRent:0.5.0-SNAPSHOT',
            'com.souche.rn:DFCWXAccountManager:0.2.1-SNAPSHOT',
            'com.souche.rn:dfc_groupingstaff:0.2.8-SNAPSHOT',
            'com.souche.rn:dfc_report:0.4.0-SNAPSHOT',
            'com.souche.rn:dfc_weibao:0.3.1-SNAPSHOT',
            'com.souche.rn:dfc_weizhang:0.2.1-SNAPSHOT',
            'com.souche.rn:RNClueSetting:0.4.0-SNAPSHOT',
            'com.souche.rn:xfd_platform:0.3.6-SNAPSHOT',
            'com.souche.rn:dfc_coupons:0.6.2-SNAPSHOT',
            'com.souche.rn:SCCFastAuction:0.2.0-SNAPSHOT',
            'com.souche.rn:dfc_wallet_rn:0.4.2-SNAPSHOT',
            'com.souche.rn:SCCCarSourceFilter:0.2.0-SNAPSHOT',
            'com.souche.rn:RNCarBrandSelect:0.1.6-SNAPSHOT',
            'com.souche.rn:RNCitySelect:0.1.3-SNAPSHOT',
            'com.souche.rn:grab_car:0.1.13-SNAPSHOT',
    ]
    androidDependency
}
```

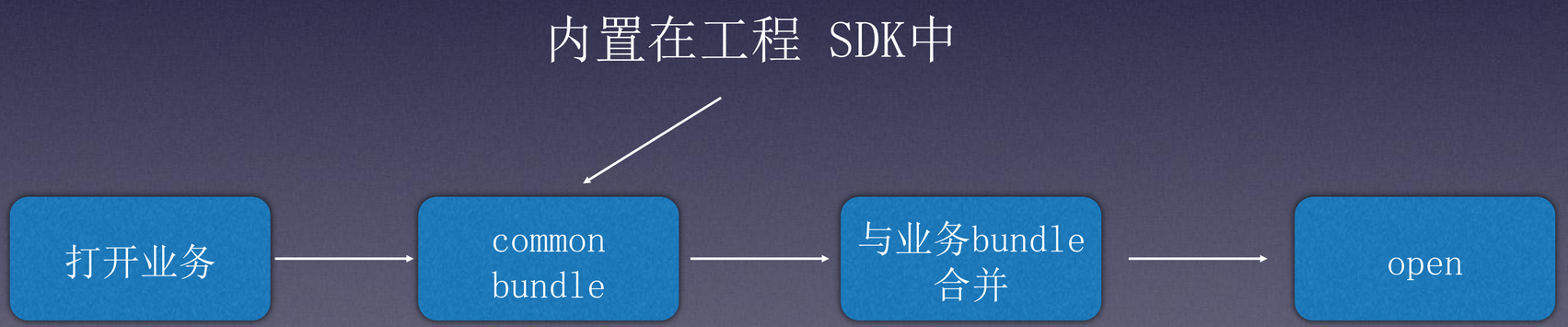
```
#RN

pod 'SCCReactNative', '2.0.5'
pod 'DFCDirectRent', '0.3.2'
pod 'DFCWXAccountManager', '0.2.1'
pod 'dfc_groupingstaff', '0.2.8'
pod 'dfc_report', '0.3.0'
pod 'dfc_weibao', '0.3.1'
pod 'dfc_weizhang', '0.2.1'
pod 'RNClueSetting', '0.4.0'
pod 'xfd_platform', '0.3.6'
pod 'dfc_coupons', '0.6.2'
pod 'SCCFastAuction', '0.2.0'
pod 'dfc_wallet_rn', '0.4.2'
pod 'SCCCarSourceFilter','0.2.0'
pod 'RNCarBrandSelect','0.1.6'
pod 'RNCitySelect','0.1.3'
pod 'grab_car', '0.1.13'
end
```

app 启动后流程


app 启动
扫描应用文件
找到所有bundle 内的 info.json
解析并注册模块
检查热更新
SRN - HUB
返回需要更新的bundle 列表和地址
下载并覆盖
测试环境
忽略规则，直接更新最新版
直接更新最新版 bundle

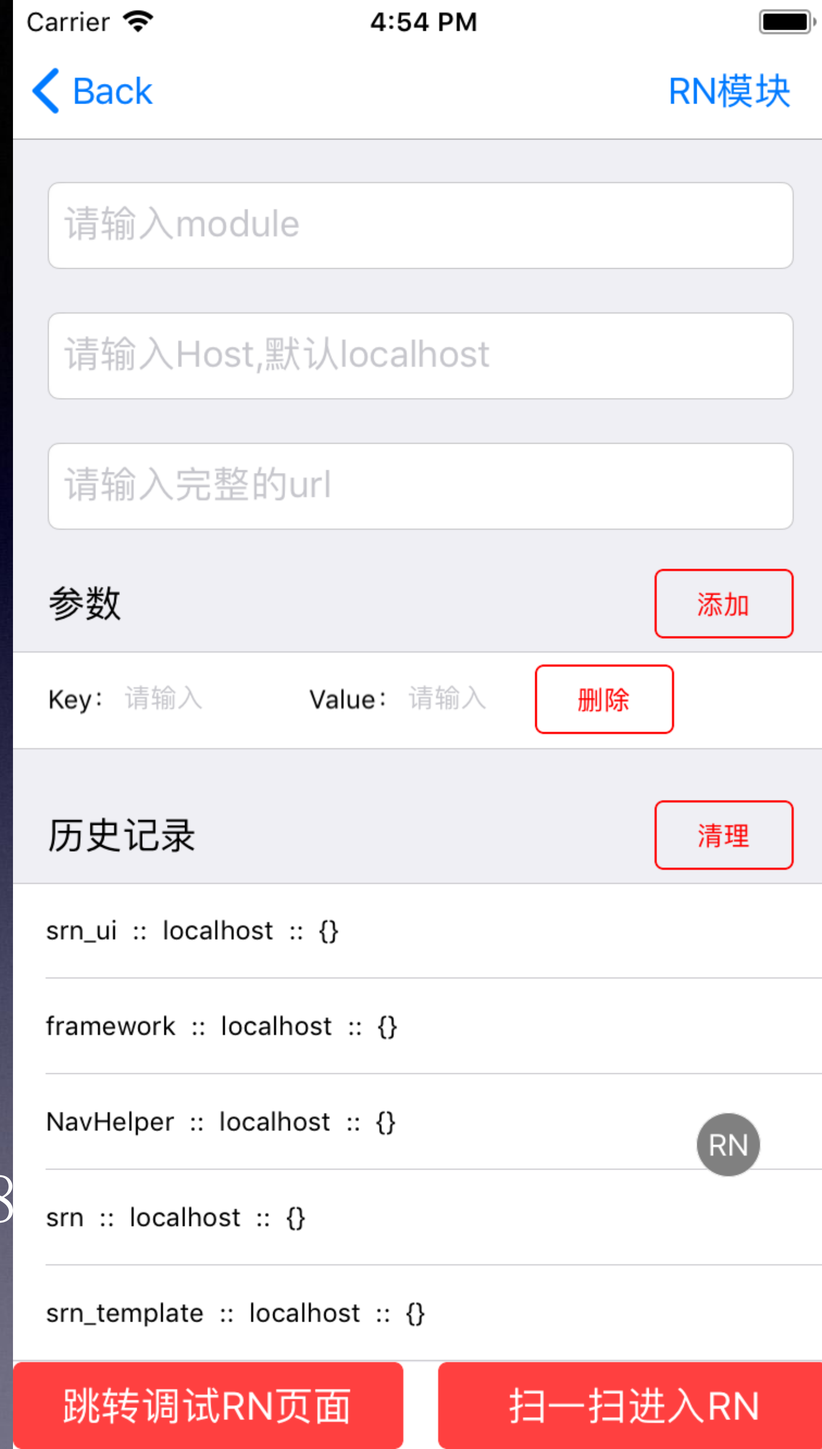
bundle 拆分


发布
分析依赖
剔除黑名单的依赖
打包成业务bundle


内置在工程 SDK中
打开业务
common bundle
与业务bundle合并
open

扫码调试

```
{
    “bundleName”:”workspace”,
    “bundleProps”:{},
    “bundleAddress” :”http://192.168.105.16:808
}
```



遗留问题

- 发布脚本时仍然会让你选择发到哪个环境，其实对于原生包来说，没有环境的概念，而是用版本号格式表达。
- 热更新导致发布不可控，产品一直想要热更新，业务太多，app内容频繁发生变化。
- 测试不方便，不知道当前是什么版本的业务，也控制不了要测试的是哪个版本的业务，测试通过后需要修改依赖。

3.0 版

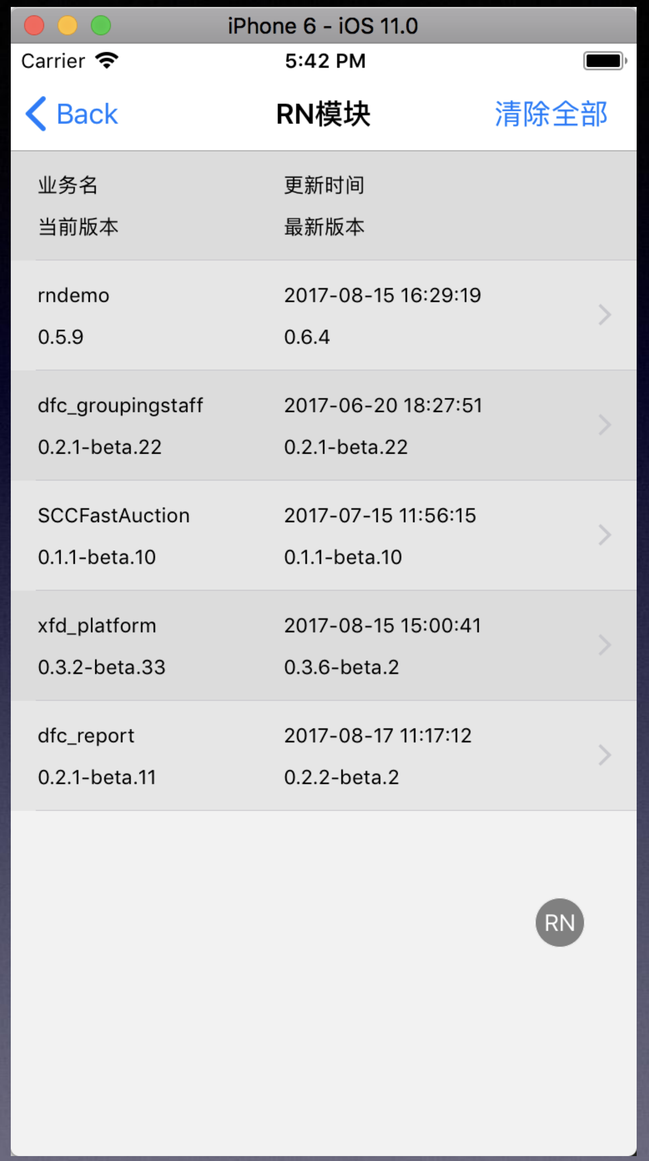
- 遵照客户端理念，去掉包环境的概念
- 关闭热更新，去掉版本号依赖规则，增加热修复
- 新的测试工具，可以回朔任何业务的任何版本
- 分离测试工具与业务 SDK 的耦合
- 容器复用机制

热修复

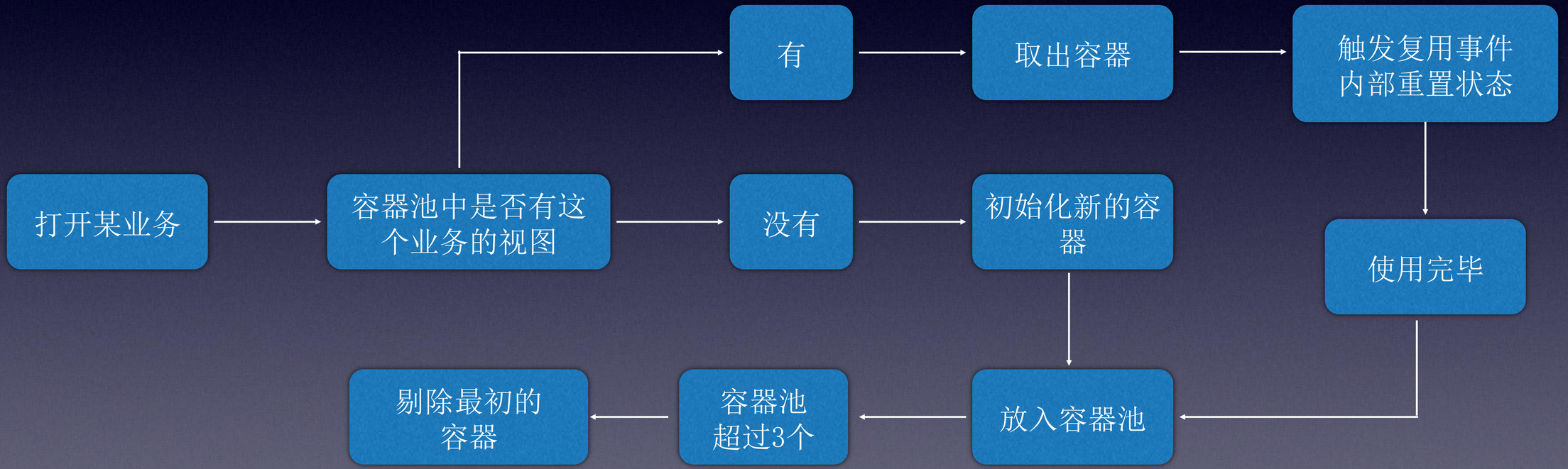
- 不按照版本号规则，因为不需要严格的版本锁定。
- 直接在后台定义需要把哪个版本的业务 bundle 更新成哪个版本的 bundle 的规则。
- 热修复需要抄送副总裁和各个大佬才能发布。

测试工具

不依赖更新规则
手动选择测试包



容器复用


打开某业务
容器池中是否有这个业务的视图
有
取出容器
触发复用事件
内部重置状态
使用完毕
没有
初始化新的容器
放入容器池
容器池超过3个
剔除最初的容器

“脱离场景谈架构都是耍流氓。”

– 路人甲

Thanks

芋头

大搜车无线架构团队

Nodejs 中间件、客户端、前端、RN

个人开发者

《颜文字输入法》《二十一点睡前故事》

《前端乱炖》《设计搞》

芋头

浙江 杭州

扫一扫上面的二维码图案，加我微信