NJSD

中国（南京）软件开发者大会

China (Nanjing) Software Developers Conference

2016

灵犀客户端技术架构与研发经验分享

黄明登 科大讯飞 2016年4月

内容大纲

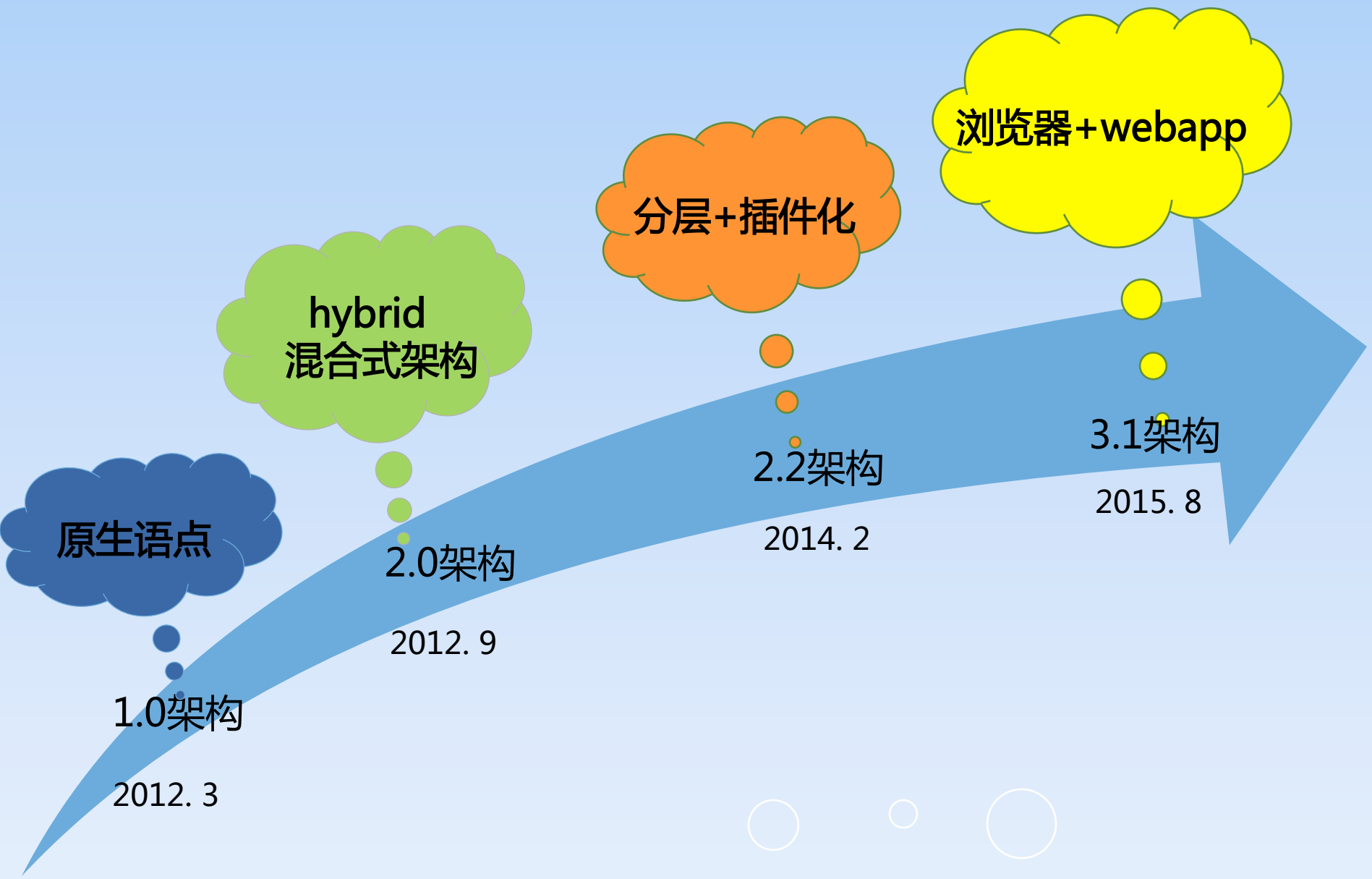
1. 客户端架构演化史
2. 历史架构简介
3. 当前架构介绍
4. 后续发展规划
5. 研发实践经验分享

灵犀是什么

灵犀语音助手由科大讯飞和中国移动联合出品，采用全球最先进的语音识别技术且拥有丰富本土化服务，识别准确，唤醒迅速，特别针对中文口音进行识别优化

目前是国内市场占有率第一的中文语音助手，总用户数1亿+

灵犀客户端架构演化


浏览器+webapp
分层+插件化
hybrid
混合式架构
原生语点
3.1架构
2015. 8
2.2架构
2014. 2
2.0架构
2012. 9
1.0架构
2012. 3

内容大纲

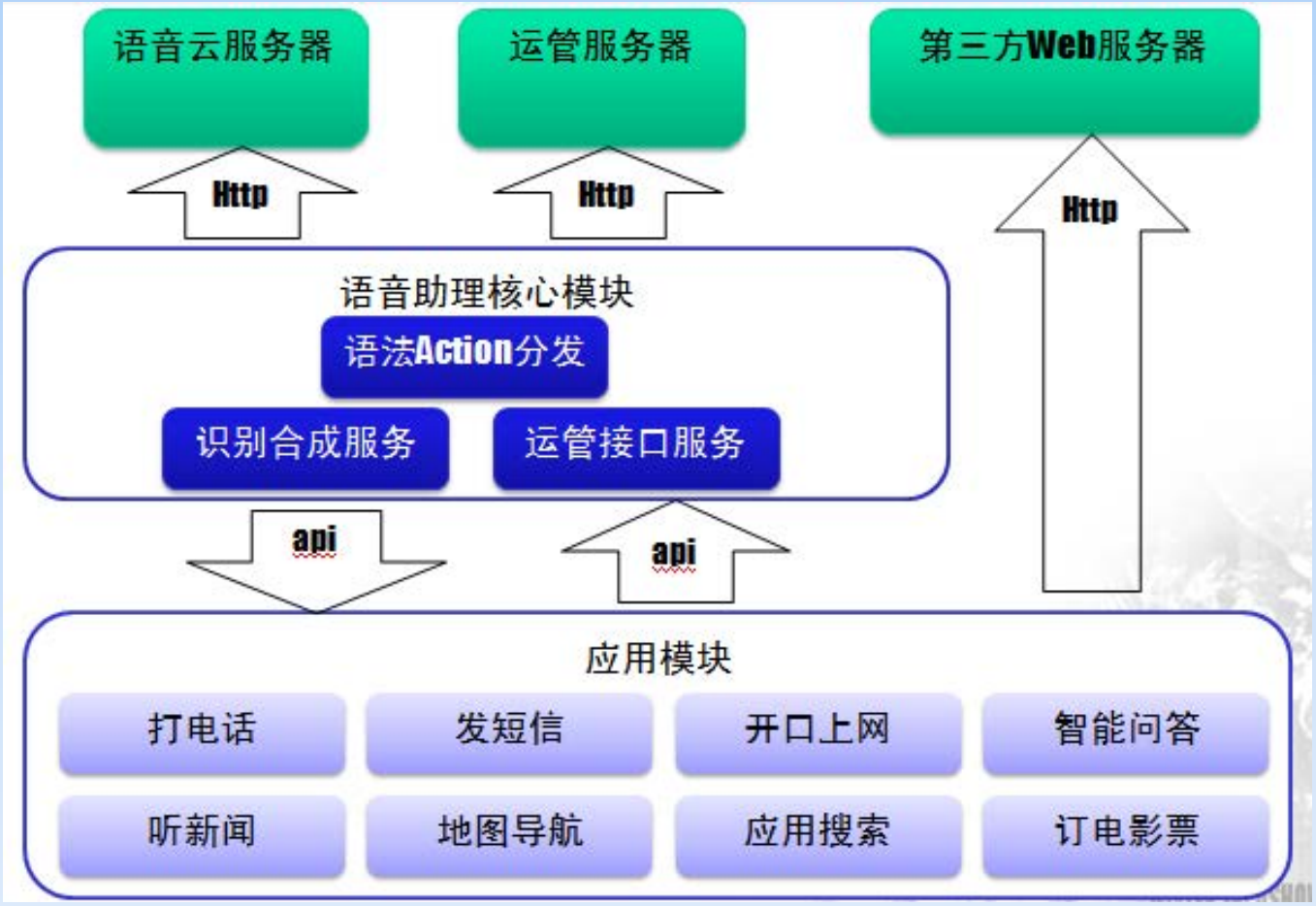
1. 客户端架构演化史
2. 历史架构简介
3. 当前架构介绍
4. 后续发展规划
5. 研发实践经验分享

客户端1.0—产品形态

宫格模式

对话模式

1.0总体架构



主体分为应用模块和核心处理模块，应用模块负责各业务功能实现与用户交互处理，核心模块提供语音能力支撑与语义结果分发，满足产品在初始阶段地语音交互功能

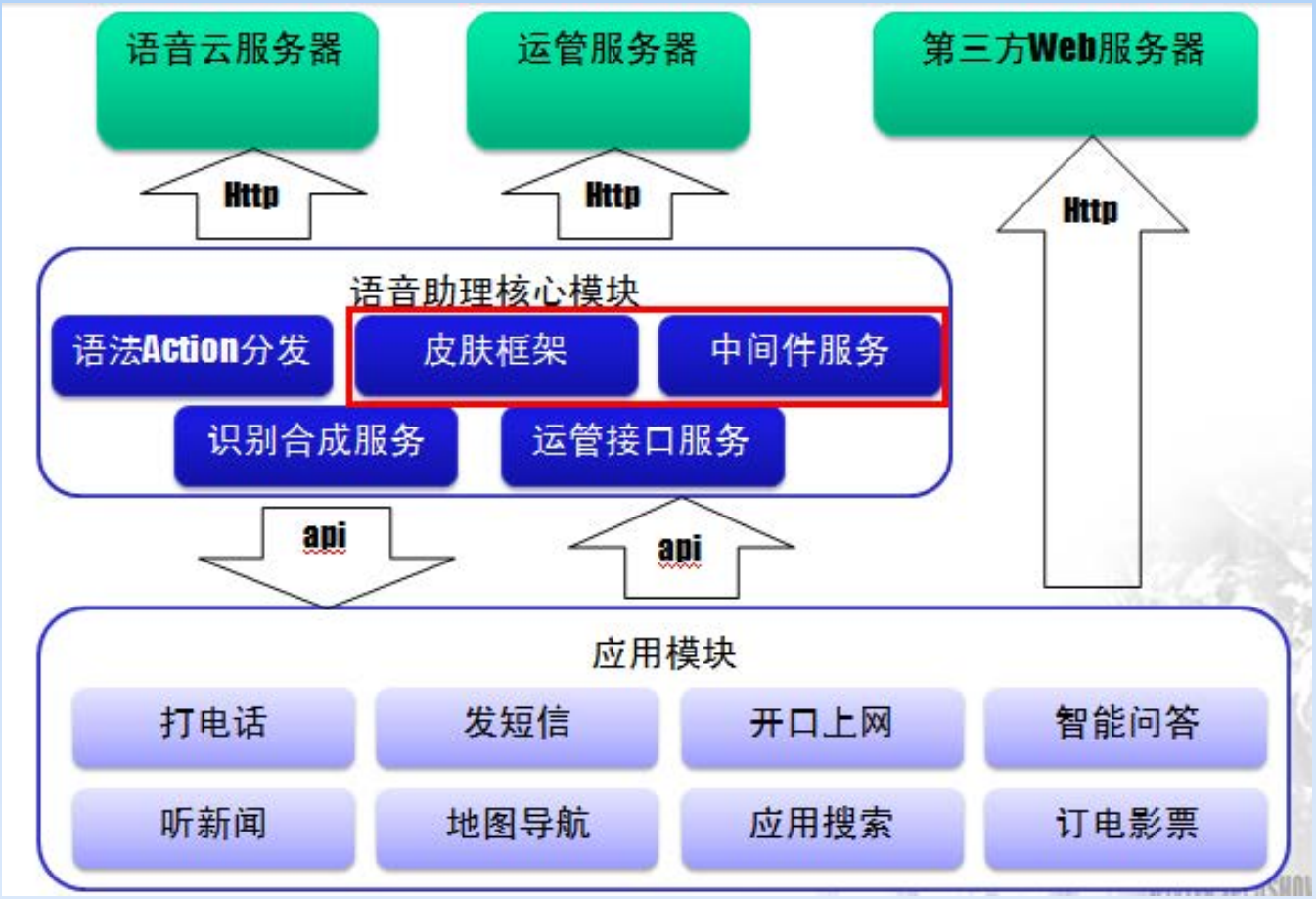
客户端2.0—产品形态

青春版皮肤

奥运版皮肤

与1.0相比，最主要的就是实现了动态换肤功能

2.0总体架构



在1.0架构的基础上增加了“皮肤框架”和“中间件服务”两大组件

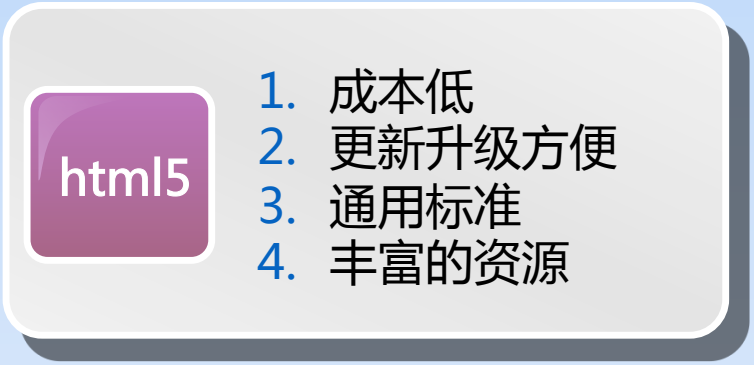
中间件服务—Hybrid框架

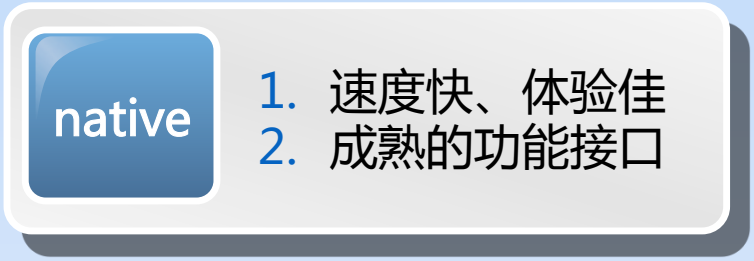
web层为采用标准的网页开发技术所实现的应用界面及交互逻辑

中间层封装系统webview控件，负责网页的解析和展现，封装手机系统的原生能力，提供网页代码与本地代码之间的交互机制

为什么要使用混合式技术架构

- html5技术与native技术优势对比





我们的考虑

- 产品在不同平台的差异性较小，UI层面没有强烈的酷、炫要求
- 产品平台化的发展策略，要求快速开发与更新业务
- 灵活、多变的运营需求

综合以上考虑，决定在Android灵犀中尝试hybrid技术架构。于是在调研了当时几种主流的hybrid框架后，参考phoneGap框架实现了灵犀的中间件框架

遗留问题—webview内存泄露

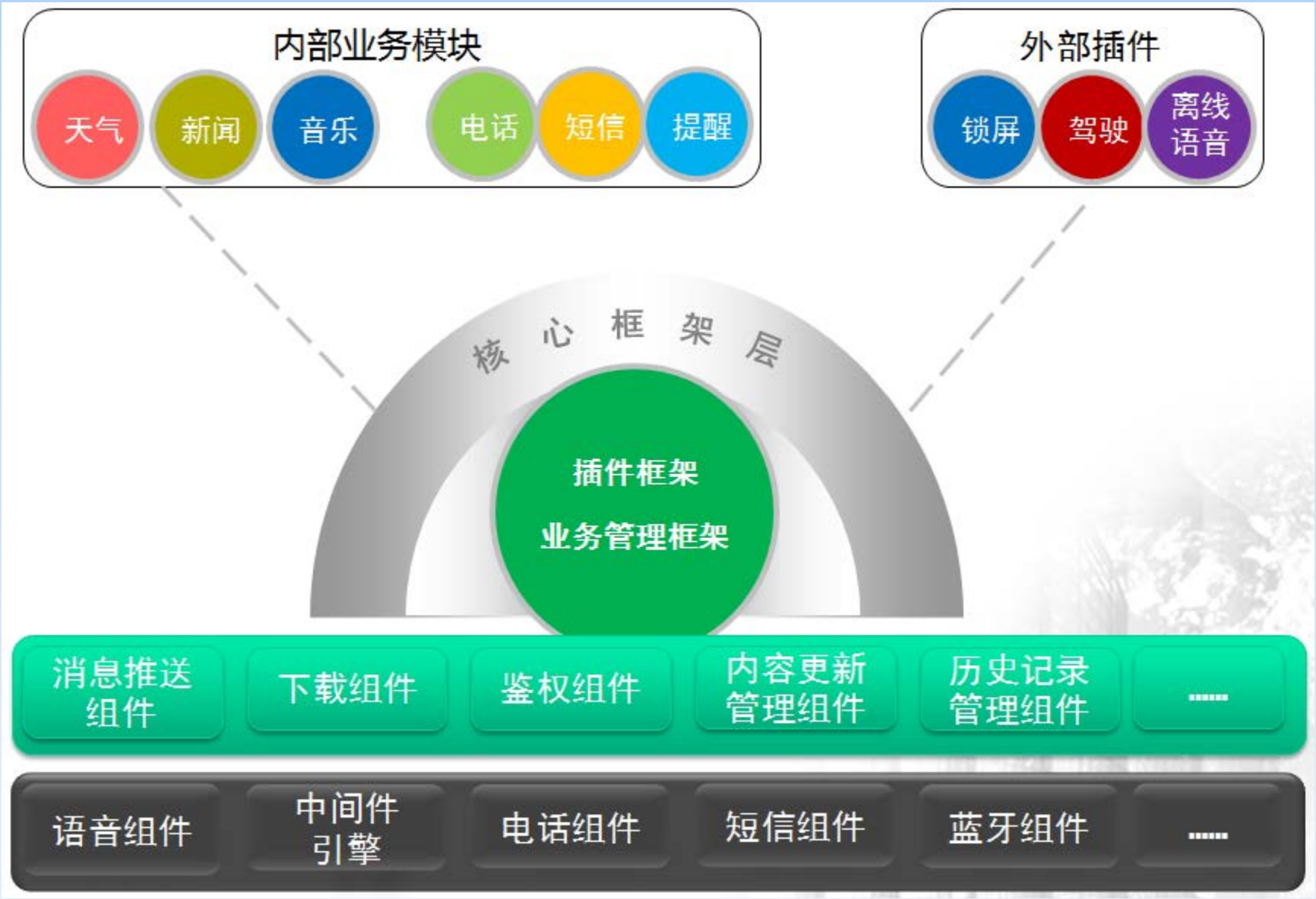
- webview存在内存泄露情况，出现在external/webkit/Source/WebKit/android/WebCoreSupport/UrlInterceptResponse.cpp，该问题至今未修复
- 影响：客户端内存一直增长，使用多次之后内存将变得很大，从而被系统强制关闭程序
- 终极解决方案：采用多进程的架构设计，这个在后面章节有详细描述

客户端2.2—产品形态

频道列表

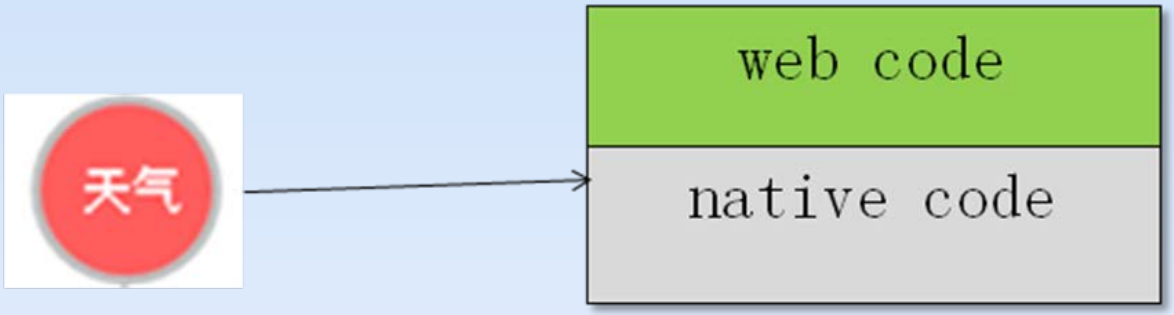
插件中心

2.2总体架构


内部业务模块
天气
新闻
音乐
电话
短信
提醒
外部插件
锁屏
驾驶
离线语音
核心框架层
插件框架
业务管理框架
消息推送组件
下载组件
鉴权组件
内容更新管理组件
历史记录管理组件
.....
语音组件
中间件引擎
电话组件
短信组件
蓝牙组件
.....

内部业务模块

- 业务模块彼此独立，每个业务都有自己的逻辑组成，但大体架构保持一致，即采用web+native的方式：web技术实现界面展现与交互逻辑，native部分负责业务逻辑处理。
- 以天气业务模块为例进行说明：



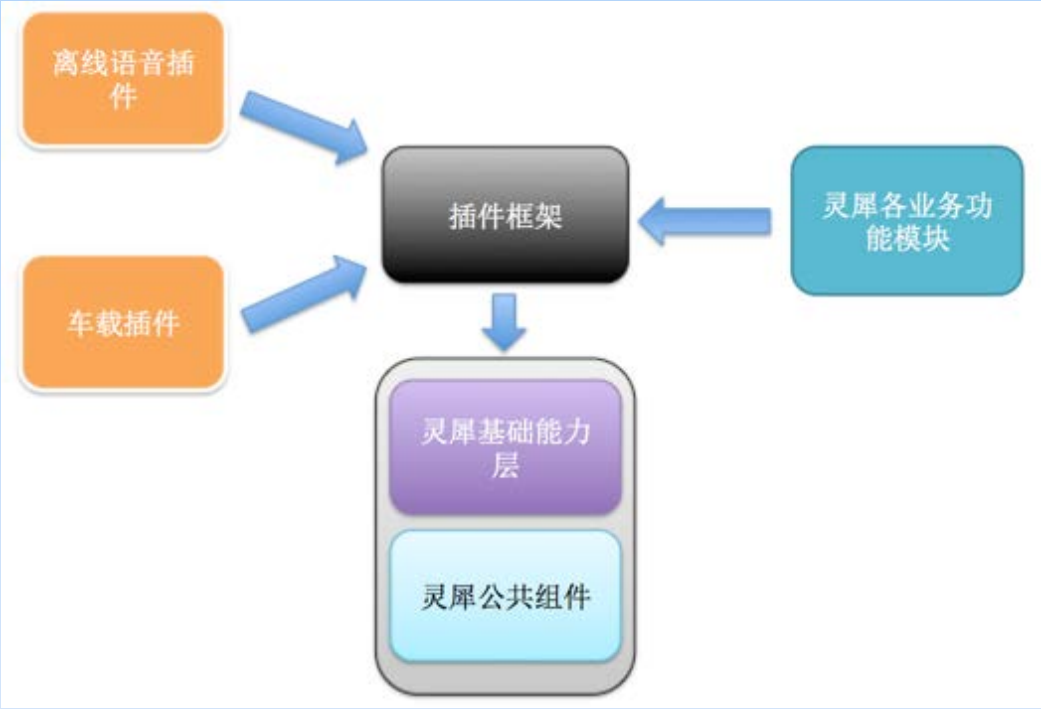
业务模块架构

- web层主要负责天气视图的绘制，界面事件的响应处理，以及根据状态对界面进行刷新
- native层主要负责语音结果的处理，网络协议解析，界面数据组装以及播报、分享功能的实现

插件框架

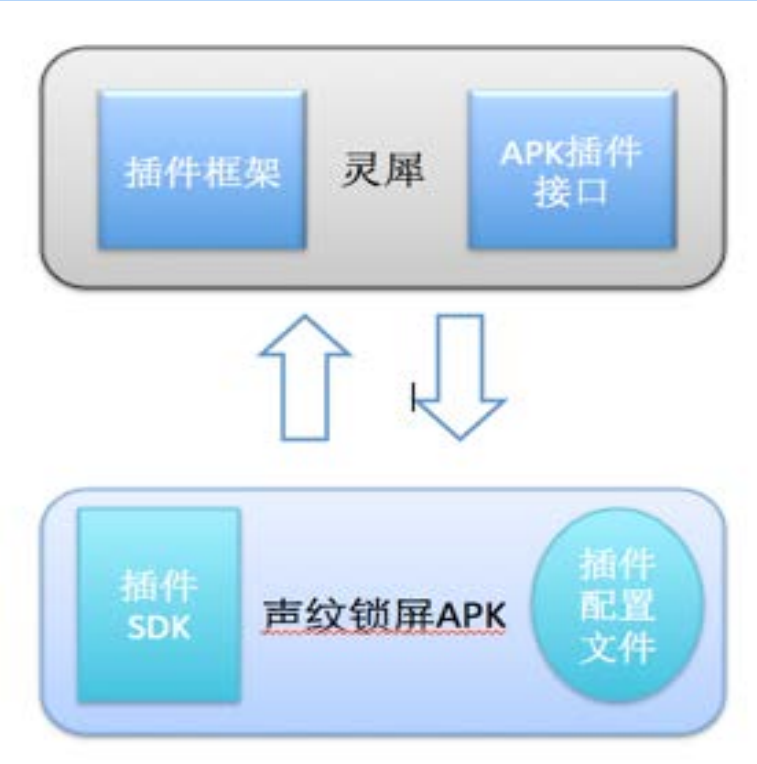
1、生命周期管理：管理各个插件的生命周期
2、插件能力管理：对插件和主程序、插件之间的调用交互进行统一管理
3、插件实体：插件模块的逻辑体现，包括插件的逻辑实体、资源实体对象，插件对外提供的能力接口

Jar包插件



依赖灵犀程序，不能独立运行。此类插件的生命周期完全依赖插件框架的管理调度。框架负责控制插件的安装、卸载、更新，通知插件当前的生命周期状态，提供灵犀主程序各种基础能力接口供插件内调用

apk插件



此类插件的安装、卸载都由系统应用程序管理器统一控制，插件框架只能通过监听应用程序变化的广播事件来判断插件的安装状态

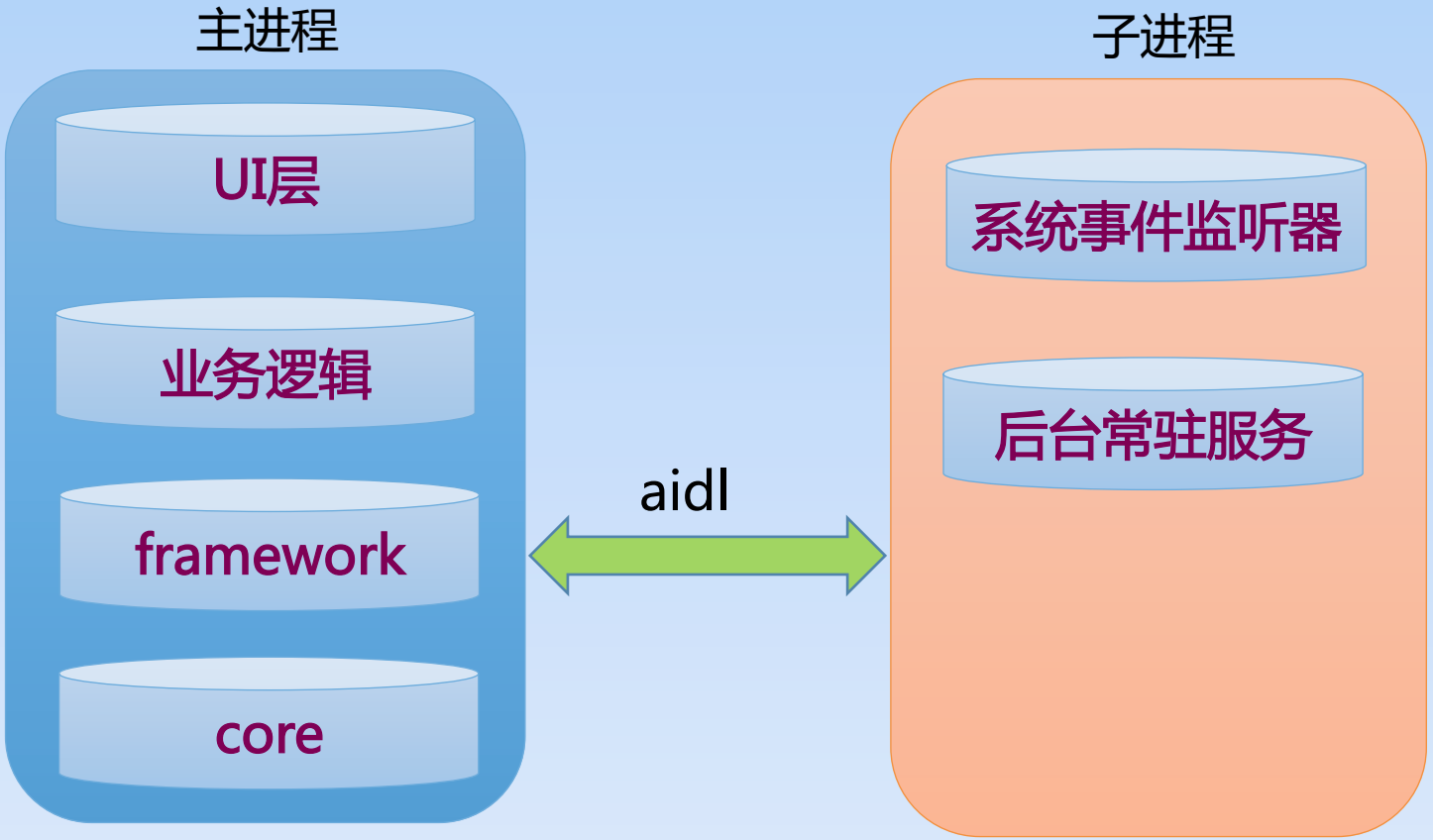
APK插件需要集成灵犀提供的插件SDK，在APK内部添加插件配置文件。主程序插件框架可以查询到集成了插件SDK的应用程序，通过SDK获取应用内插件配置的相关信息

双进程架构

1、解决hybrid架构中由于webkit内存泄露、内存占用大的问题

2、解决程序在后台经常被系统关闭而导致的消息推送、业务事件监听（提醒、来电、短信）失效的问题

双进程架构

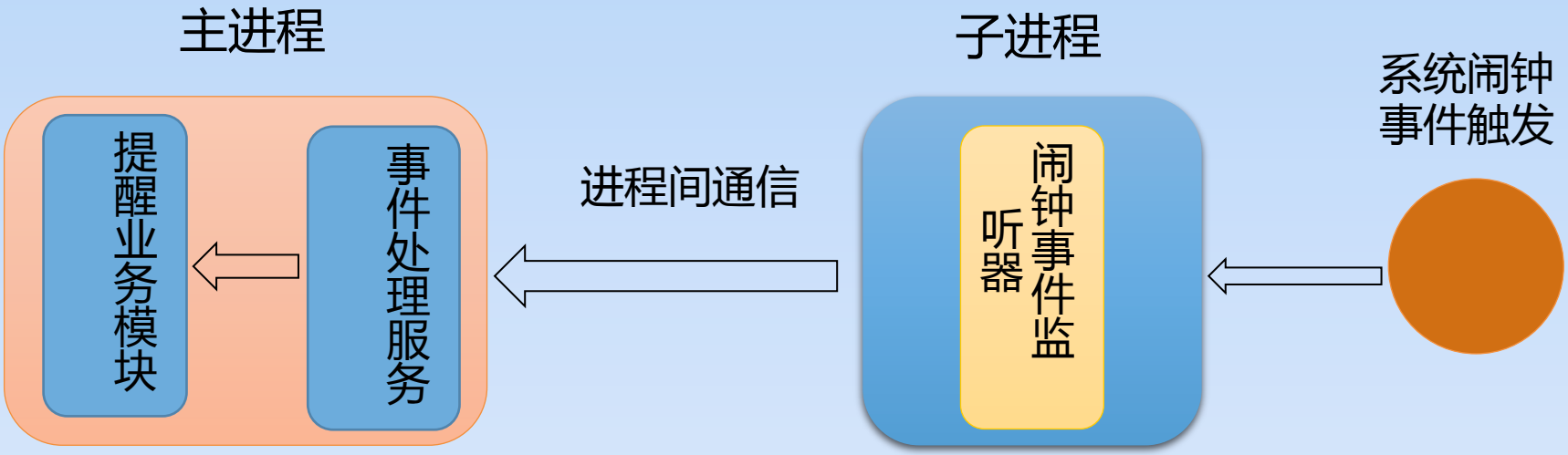


将系统事件监听逻辑和一些需要在后台常驻的服务移植到独立的小进程，这是一个非常轻量级的进程，只负责事件监听等基础服务，不处理具体业务逻辑。收到事件后通过进程间通信接口将事件发送给主进程的相应模块来处理

主进程在主界面退出后即刻自行关闭，彻底释放内存。同时为了不影响下次界面的启动速度，主进程在关闭后又在后台自动重启

双进程架构

子进程和主进程交互（以提醒为例）：



子进程通过监听系统事件（开机、网络变化、亮屏暗屏等）、设置START_STICKY标识、设置为前台服务等一系列手段来提升在后台常驻的能力

内容大纲

1. 客户端架构演化史
2. 历史架构简介
3. 当前架构介绍
4. 后续发展规划
5. 研发实践经验分享

客户端3.1—产品形态

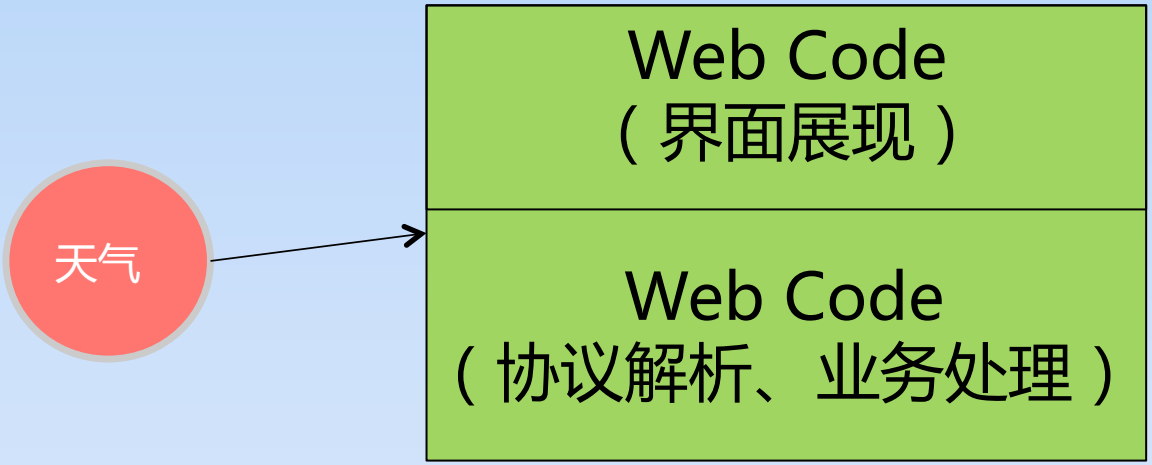
在2.2的基础上引入了浏览器框架，首页采用信息流卡片式结构，整个产品类似于语音助手+浏览器

3.1总体架构


webapp
天气
新闻
音乐
Native模块
电话
短信
提醒
外部插件
锁屏
驾驶
离线语音
核心框架层
浏览器框架
插件框架
业务管理框架
消息推送组件
下载组件
鉴权组件
内容更新管理组件
历史记录管理组件
.....
语音组件
中间件引擎
电话组件
短信组件
蓝牙组件
.....

webapp

- 信息类业务模块完全web化，还是以天气模块为例说明

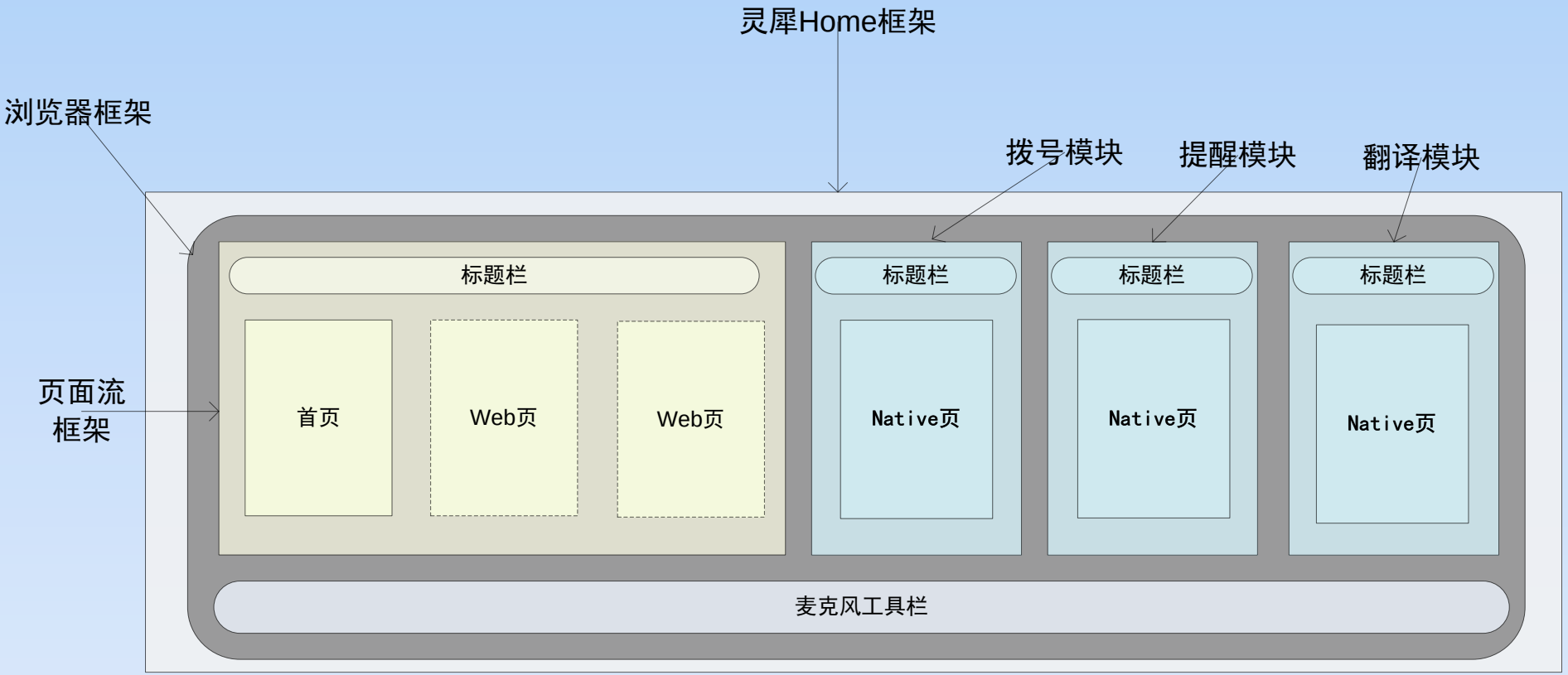


- 从UI展现到数据请求、协议解析以及业务逻辑处理都是使用web技术来实现，对于需要使用到的基础能力（如播报、定位等）由浏览器框架提供

webapp

- 业务模块完全web化后，与底层客户端完全解耦，很容易移植到其它平台，比如iOS版灵犀、h5版灵犀

浏览器框架



提供全局统一的麦克风控件，管理web页面流堆栈和历史记录，支持web页面与native页面之间的相互切换

灵犀首页技术路线选择

原生方式（手机百度、猎豹浏览器）：
优点：性能和加载速度较快，支持离线
缺点：卡片动态调整难以实现

web方式（qq浏览器、360浏览器）：
优点：卡片动态调整容易
缺点：加载速度慢，离线支持不好

鱼和熊掌无法兼得？

解决方案

web package方案：
原理：
html、js、css以及资源文件打成一个压缩包，下载到本地后再加载；压缩包携带版本号，根据版本号比较是否有更新
优点：
更新过程完全可控，技术风险低；初始资源包可以预置在客户端中，初次加载速度快
缺点：无法做到文件级别的更新

灵犀情况：卡片动态更新是一个低频需求（1个月1、2次），资源包压缩后大小在100k左右

更新机制：后台更新
主界面启动的时候检查更新，资源包下载完成后下次启动主界面时生效

内容大纲

1. 客户端架构演化史
2. 历史架构简介
3. 当前架构介绍
4. 后续发展规划
5. 研发实践经验分享

近期规划

插件化2.0

- 支持内部功能模块apk化：完全解耦，动态更新
- 支持外部独立apk免安装运行：提高业务转化率

代码热修复

- 在线bug修复

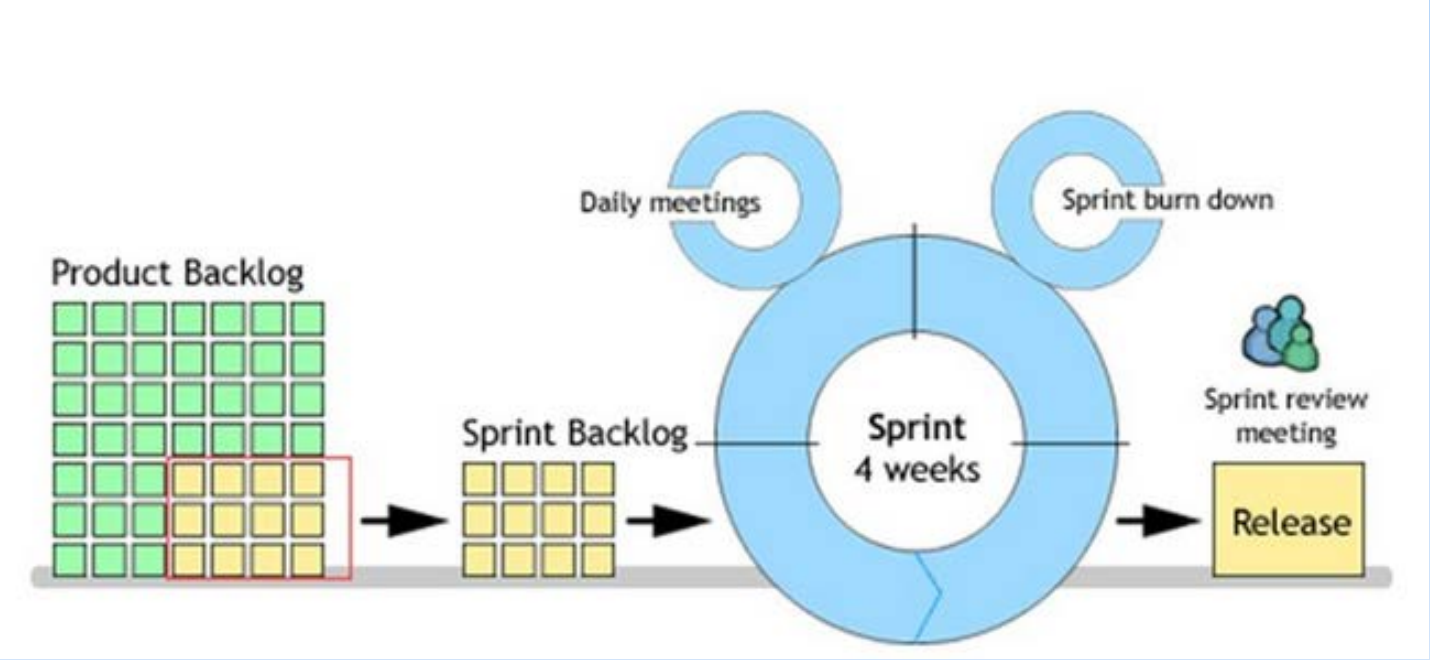
质量数据采集与监控

- 采集基础性能数据：cpu、内存、流畅度、响应时间、ANR等
- 在线监控与告警

内容大纲

1. 客户端架构演化史
2. 历史架构简介
3. 当前架构介绍
4. 后续发展规划
5. 研发实践经验分享

敏捷实践



灵犀目前的节奏：1个Sprint周期为1个月，发布两个版本，每天进行晨会，电子看板，系统每天自动发布燃尽图，每日构建，Sprint周期结束后进行Sprint Review

版本内测与试上线

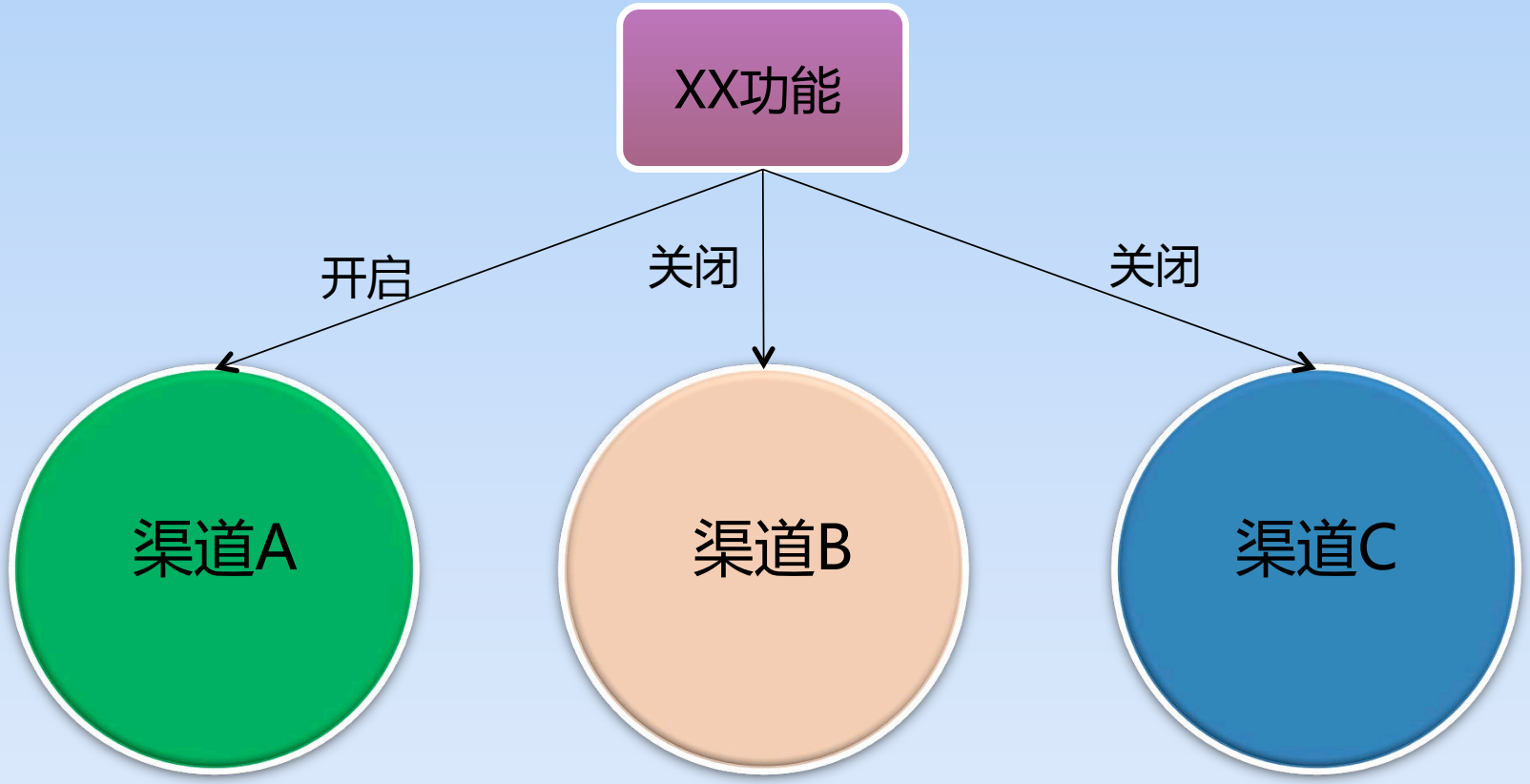
内测

版本发布前一周组织内测，范围：粉丝QQ群。目前的效果：每次有效参与人数有不少，但是反馈需求或建议较多，反馈bug较少

试上线

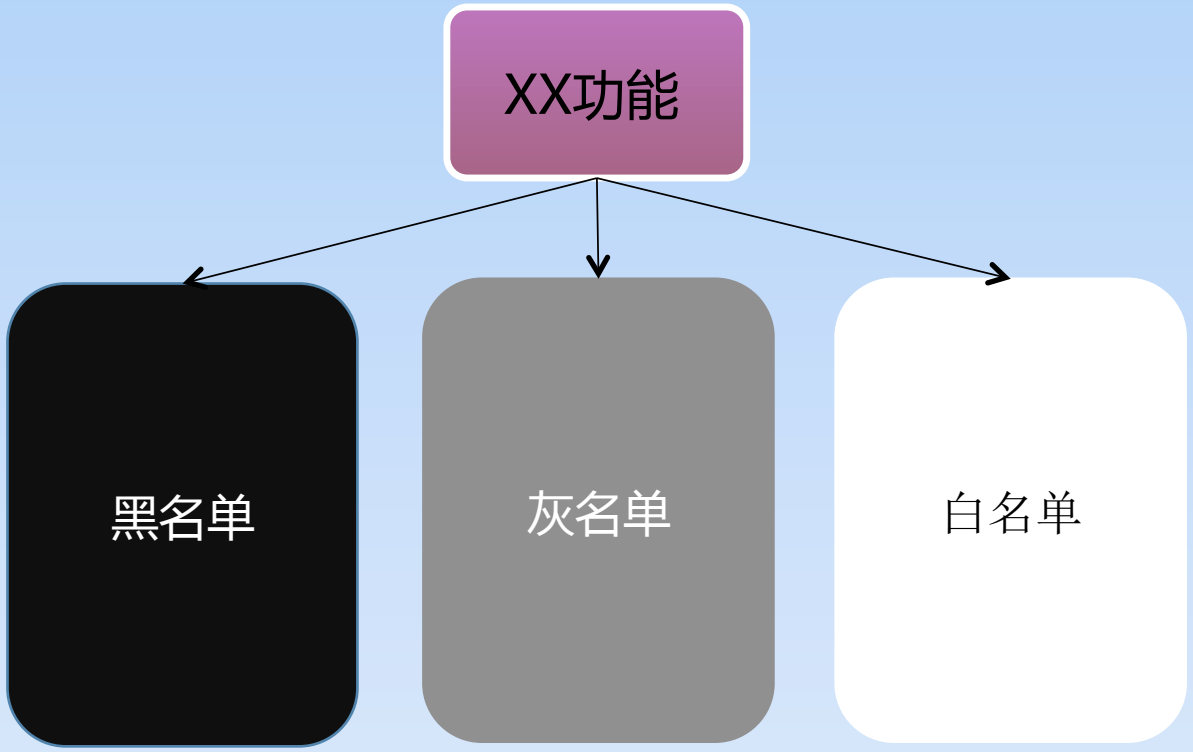
版本发布前三天，在部分渠道上先进行发布。主要目的是观测版本发布后的现网实际效果和可能暴露的问题。试上线三天没有发现严重问题，则在全渠道正式上线。

功能灰度发布—按渠道



对于风险较大的功能，在技术方案设计的时候就需要考虑灰度发布的策略，上线后可以只在部分渠道开启，待观察一段时候再扩大范围

功能灰度发布—按机型



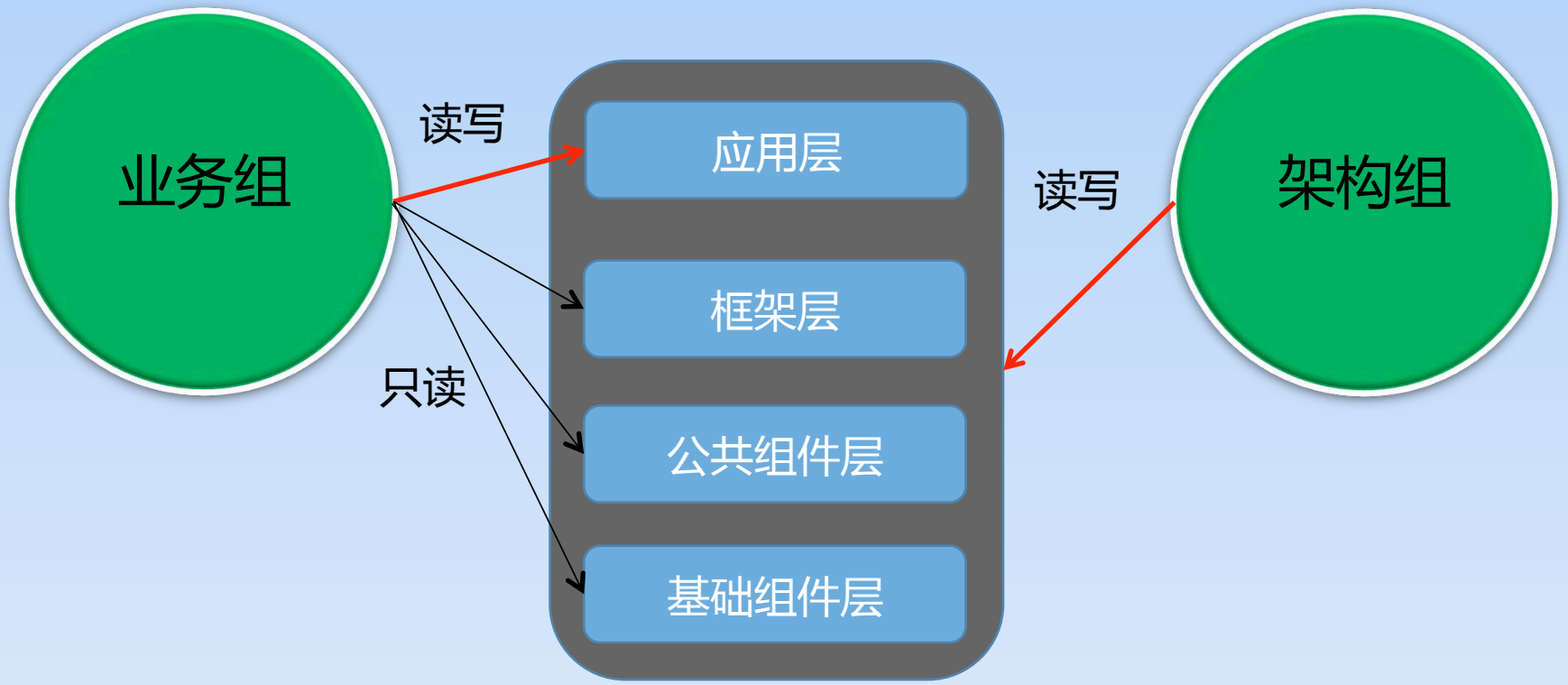
由于Android手机机型众多，灵犀中的电话、短信、蓝牙、识别、锁屏功能受到各手机的软、硬件限制，不能完全支持，故采用灰度控制方案

机型黑名单：功能默认关闭，用户不可见

机型灰名单：功能默认关闭，用户可见，可以手动开启

机型白名单：功能默认开启，用户可见，可以手动关闭

权限管理



按照架构层级进行控制，业务组同学主要负责业务开发，因此对应用层代码拥有读写权限，对其它层则只有只读权限；架构组同学则拥有全部权限。这种方式在一定程度上加强了对底层代码的保护，降低了代码修改对整个架构带来的冲击

Hybrid实践分享—分辨率适配：XCSS

```
.main_item_icon{
    vertical-align: middle;
    width: 54dp|720:120px;
    height: 54dp|720:120px;
    padding-top: 4dp|720:10px;
    padding-bottom: 4dp|720:10px;
    margin-left: 6dp|720:13px;
    margin-right: 8dp;
}
```

- dp：与分辨率成正比的尺寸单位
- 可以为某特定分辨率单独指定尺寸
- 中间件框架负责解析此文件，并生成标准的CSS样式表

Hybrid实践分享—慎用Canvas

- 内存占用大（所有手机）
- 渲染效率低（大部分手机）
- 存在兼容性问题，绘制不出来（少部分手机）

优先使用css，尽量少用Canvas，除非没有其他选择

Hybrid实践分享—JS模块化开发

- require.js：目前业界比较主流的、一款非常轻巧的第三方开源库，支持js文件的模块化异步加载
- sea.js：淘宝团队研发的开源类库，比require.js更加灵活和强大，支持js文件的同步加载

欢迎关注讯飞技术团队微信公众号