GMTC 全球移动技术大会
GLOBAL MOBILE TECH CONFERENCE

滴滴的组件化实践与优化

滴滴出行 / 李贤辉

聚焦前沿技术　传递实践经验

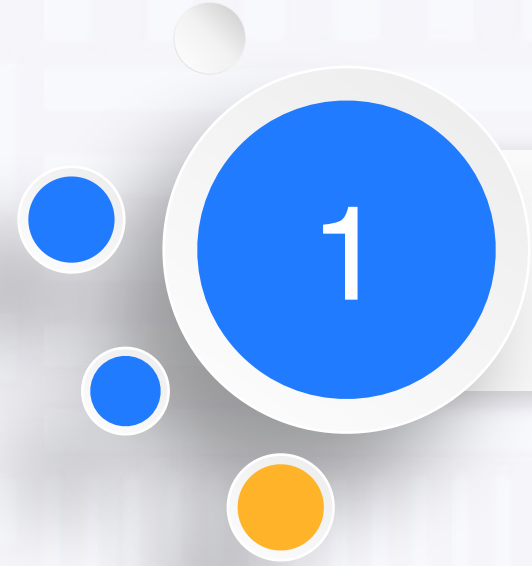
主办方 Geekbang. 极客邦科技　InfoQ

自我介绍

- 李贤辉
- 2013.5~至今 滴滴出行乘客端 iOS 开发
- 2.0~4.3.8 版本

目录

- 组件化
- 专项技术
- 问题与思考



组件化

- √背景
- √问题
- √目标
- √组件化方案
- √组件化效果

组件化-背景

1 个Project

1

70

大于 70 个开发人员数量，

分布在北京、上海、杭州

7 条业务线

7

70

70 万行代码

组件化-组件化前的问题

开发体验

1、代码冲突多。

2、即使开发小功能，也需要全部编译，开发效率低。

迭代速度

1、集成难度大，尤其是出租车、快车冲突较多。

2、任何修改，都全量回归，经常晚上 2 点以后。

版本风险

源码集成，增加或者回退某个功能，都需要修改较多代码，风险可控性低。

组件化-目标

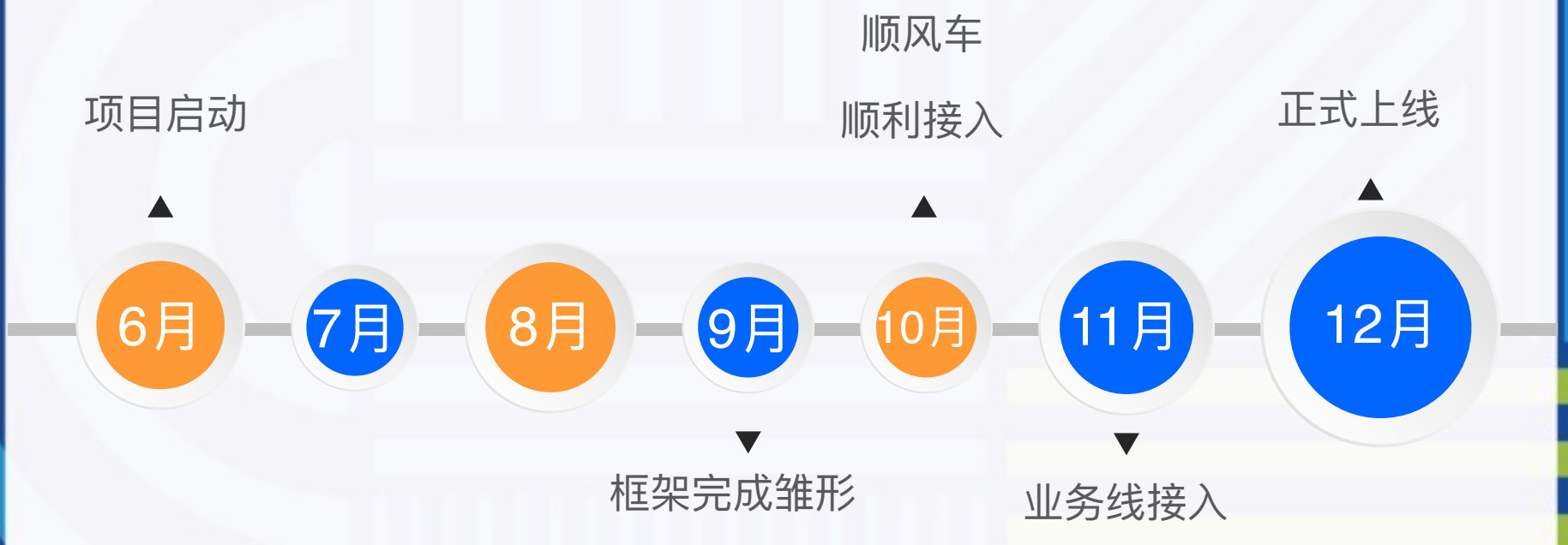
目标一

提供通用技术组件和业务组件，实现代码分而治之，方便未来重构。

目标二

改善产品迭代体验，改善协同发版。

组件化-TheOne进度



组件化-出租车业务线

组件化-独立代码仓库

组件化-业务线或SDK开发集成

独立的项目中

开发测试

项目测试后通过

Jenkins 用 tag 集成

集成后测试

组件化-滴滴出行 App

组件化-问题解决情况

开发体验

1、按团队拆开不同代码库，最多十几个人一个项目。

2、组件一般在自己项目里开发。

迭代速度

1、集成容易：大家各自生成 tag ，放在集成包里。

2、功能开发和测试都变快了。

版本风险

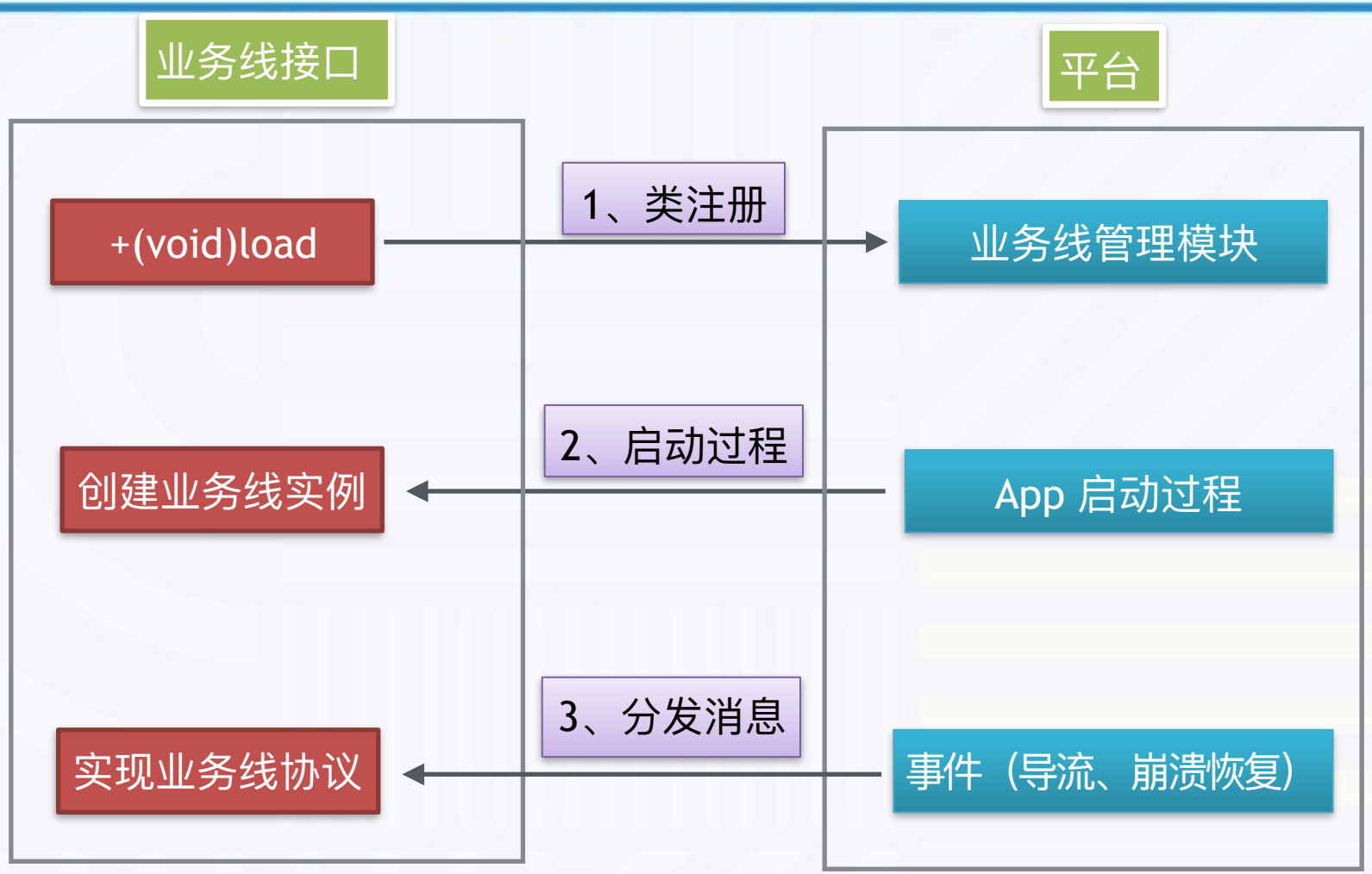
1、用 tag 的方式集成，集成和回退都容易。

2、发版前一天晚上 9 点后出最终包。

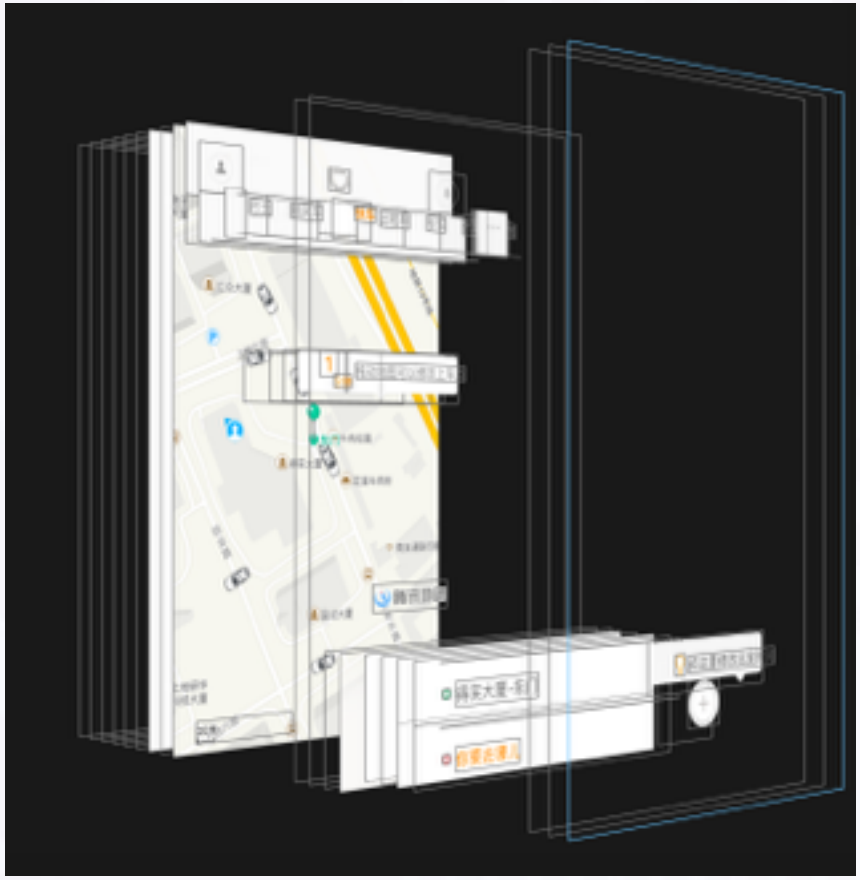
专项技术

- √ 业务线接入
- √ 页面结构
- √ 页面导航
- √ 地图模块
- √ 灰度开关

专项技术-业务线接入方式-解耦



专项技术-页面结构



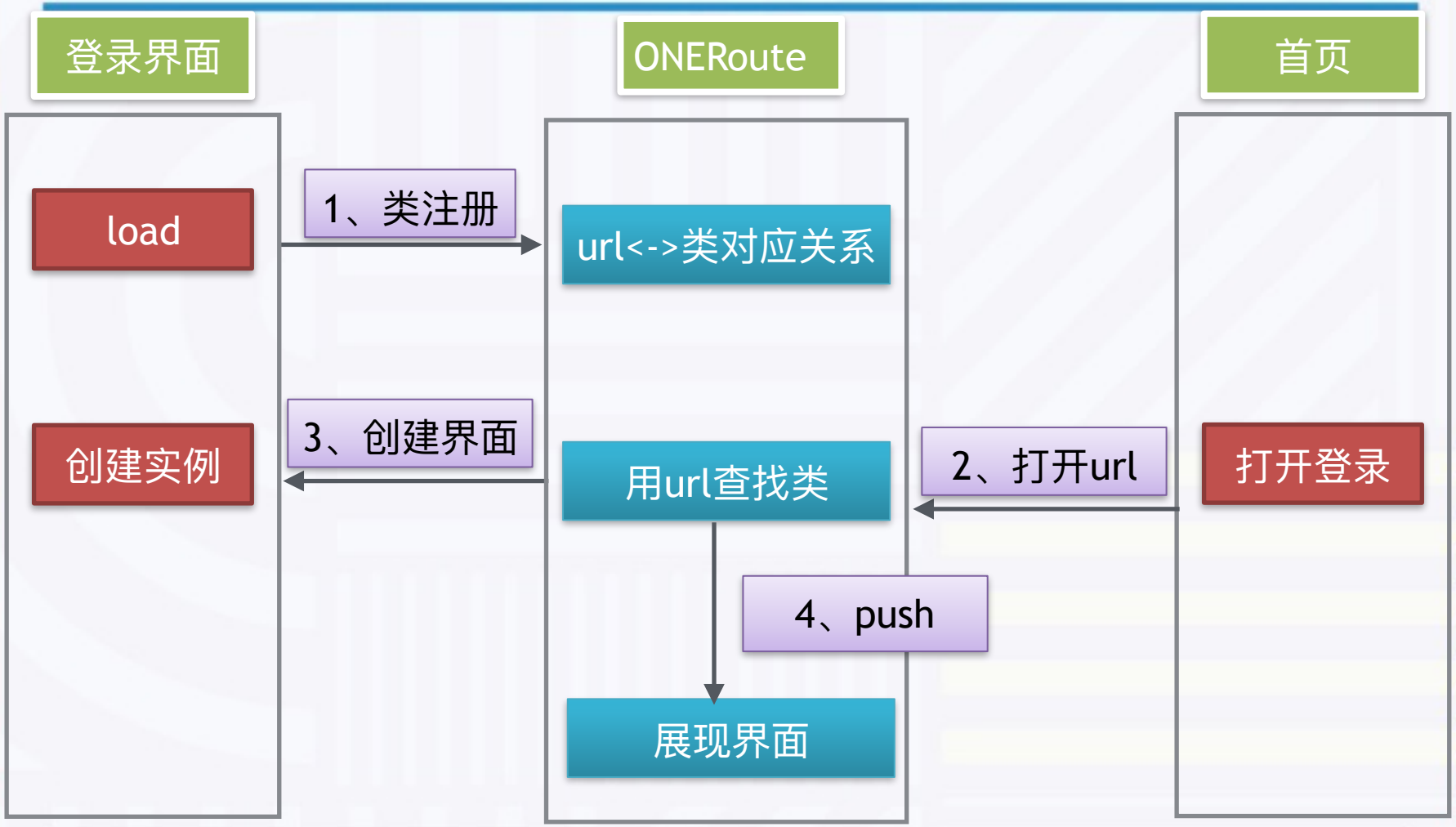
01 首页在组件里

02 业务线首页是子页面

03 业务线首页从系统的VC派生

04 hitTest

专项技术-界面导航管理组件调用示例



专项技术-界面导航管理组件优势

便捷

任何代码都可以打开页面，无需获取页面栈。

H5打开

框架层面支持 H5 打开 Native 界面。

页面解耦

页面间解耦，在首页，平台只需要知道具体的业务线首页的 url 即可，不需要知道具体的类。

专项技术-共享地图背景

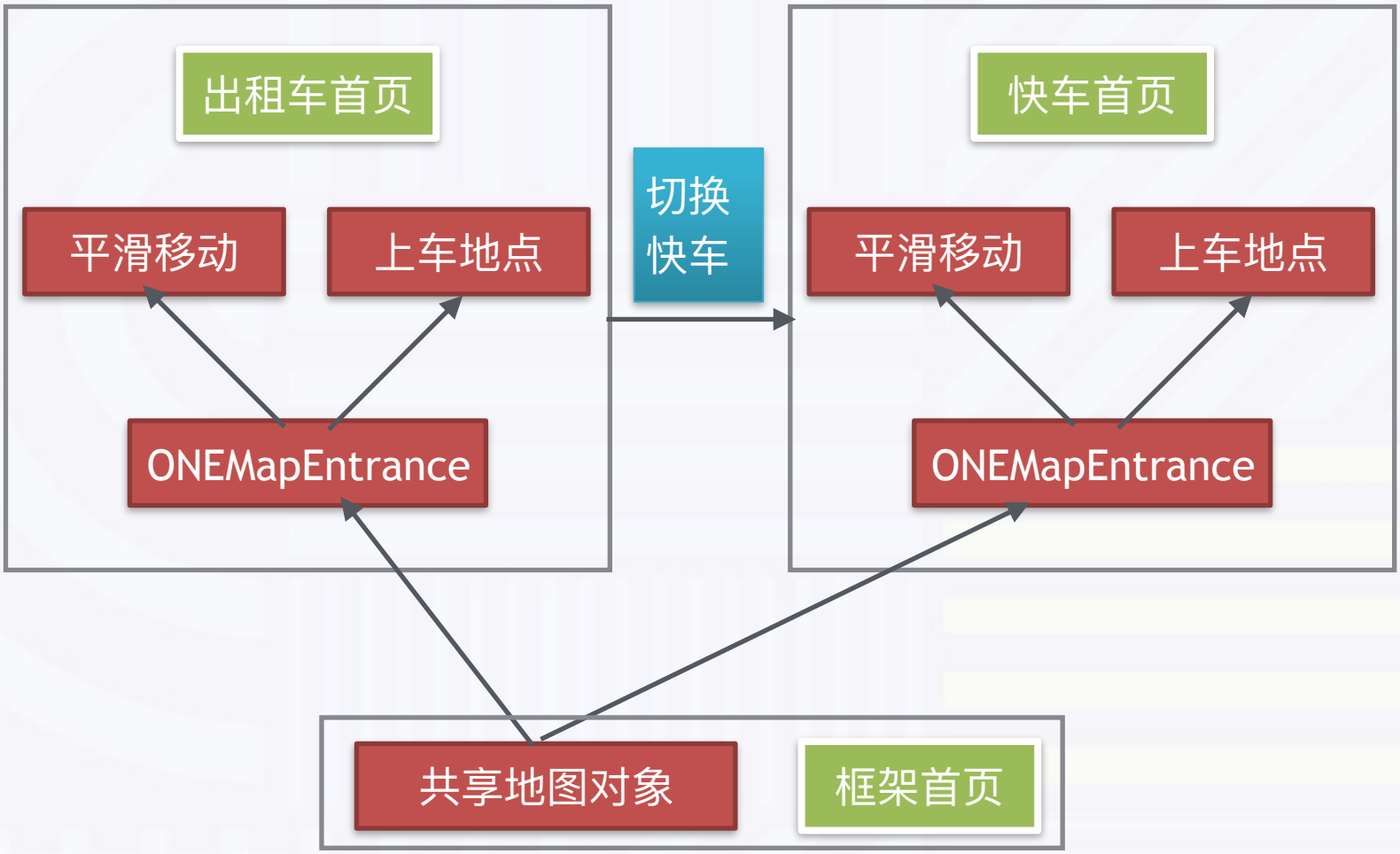
01 只有一份地图

02 切换平滑

03 内存占用

04 挑战：地图只有一个回调

专项技术-业务可扩展的共享地图的解决方案



专项技术-地图方案的优势

业务隔离

业务间隔离，平滑移动、上车地点等业务模块完全隔离，降低代码复杂性。

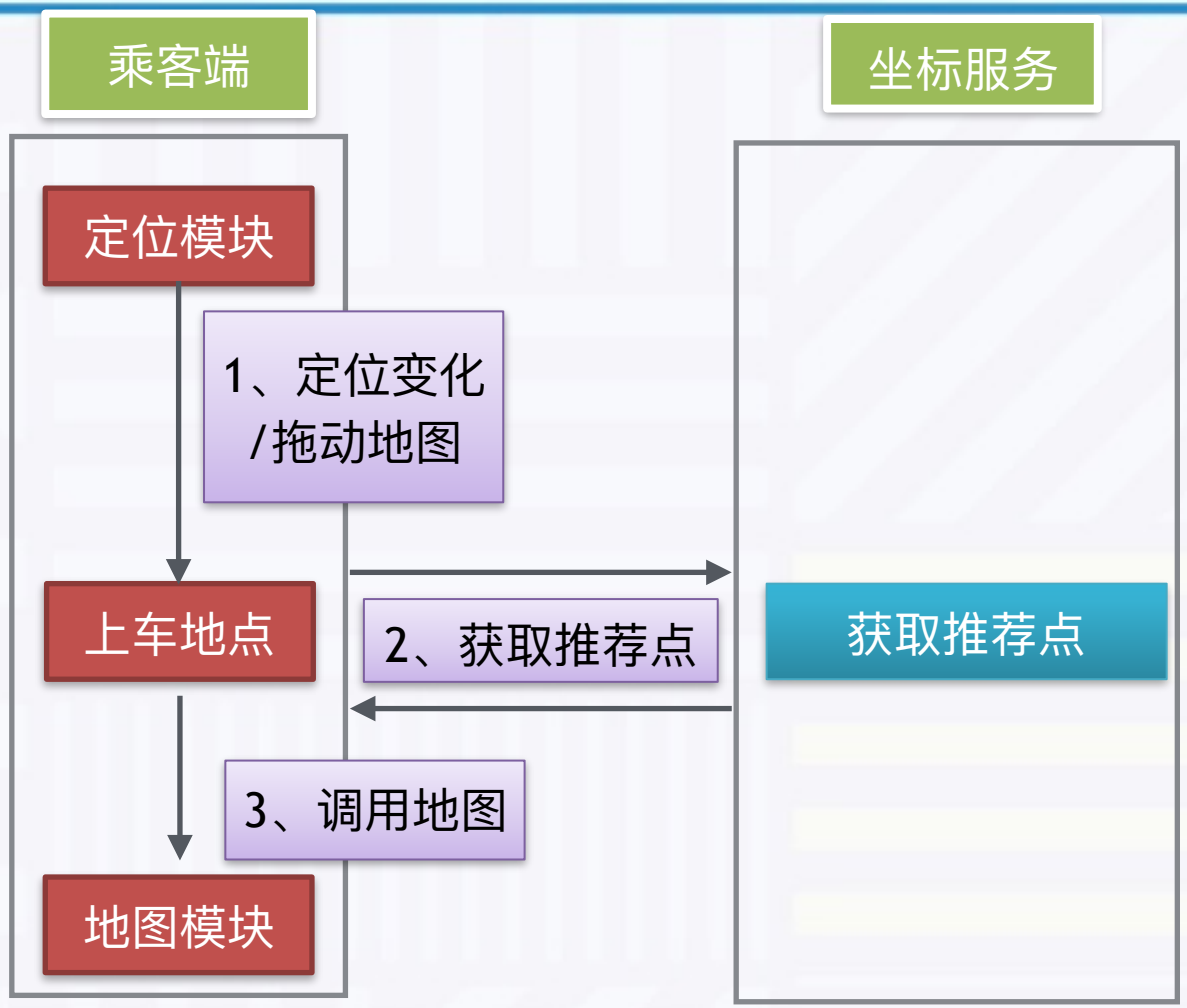
切换清除

MapEntrance 全封装，页面切换时，可以删除原页面所有内容，MapEntrance 类似于画布。

底层更新

所有的地图操作都通过 MapEntrance，上层业务避免了直接和地图交互，方便未来底层的更新。

专项技术-地图-上车地点



专项技术-地图-平滑移动


司机端
坐标服务
乘客端
上报模块
2、上报数据
保存坐标
获取坐标
3、获取数据
平滑移动
1、采集数据
4、调用地图
定位模块
地图模块

apollo 是用来帮助各种类型系统实现流量控制、灰度发布和AB test的系统，用以控制产品和技术风险。

专项技术-灰度-灰度方式

内部外部

内部：员工，外部：非公司员工；功能一般从内部员工体验开始。

城市

以城市维度拆分，例如：北京或者上海的用户，功能方面会有差异。

百分比

以一定比例放开，一般是逐步放开，观察功能是否正常，以及后台相关的崩溃。

专项技术-灰度-使用场景

新功能

新功能：如 3D Touch & Spotlight。

功能改版

功能改版：侧边栏改版、登录模块改版

技术风险

有技术风险模块：新日志模块经历若干版本才稳定。

专项技术-灰度-侧边栏灰度实践

新版

旧版

01 独立VC

02 入口处切换

专项技术-灰度-针对网络的无缓存灰度

场景

网络流量统计

风险

如果代码出问题，导致 App 不能下载任何东西，则更新的灰度策略都无法下发。

方案

默认关闭，每次都使用服务器上最新值，如果出问题，用户最多重启一次 App。

专项技术-灰度-总结

启动确定

启动时决定，默认情况下，我们仅获取一次开关状态，甚至用变量来缓存启动时开关状态。

全流程

全流程控制，保证当开关关闭时，所有流程都不生效。

及时清理

及时清理，当功能已经完全确定后，把不需要的代码都清理掉。

问题与思考

- √瘦身
- √启动速度
- √降崩溃
- √模块优化

问题与思考-瘦身

瘦身方案	周期	当前进展	风险
资源清理与压缩	短期	分析报告集成pipeline	无
无用函数	短期	demo完成	
图片资源服务器缓存	短期	开发中	风险不高，需要大家支持
跨平台化ReactNative	中期	试驾试点	FE/RD需要熟悉另外一套开发语言；性能可能会有降低
图片转换为WebP/BPG	中期	调研	图片加载性能降低；解析库增加200K
ProtoBuf瘦身	中期	技术调研	动态调用，性能可能有降低
组件化	中期	支付、评价在开发中	业务线产品层面的差异性，平台的抽象性，业务线配合度
去除异常处理no-exception	中期	技术调研	有一定代码风险
插件化动态加载	长期	技术调研	需要RD调整技术栈

问题与思考-启动速度

1. 分析耗时。
2. 业务线内部优化，包括+(void)load方法。
3. 每次启动仅加载当前业务线。

问题与思考-降低崩溃

监控

持续关注、自动监控、使用热修复修复严重崩溃。

分类

根据崩溃栈和符号表，自动分类到各个业务线。

总结

每周分析已经发生过的崩溃，团队内定期分享。

问题与思考-模块优化

下沉

把业务线依赖的内容下沉，这些模块要求稳。

上升

平台业务上升，对业务线无影响，这些模块求快。

Q & A

GMTC 全球移动技术大会
GLOBAL MOBILE TECH CONFERENCE

THANKS

聚焦前沿技术　传递实践经验

主办方 Geekbang. 极客邦科技　InfoQ