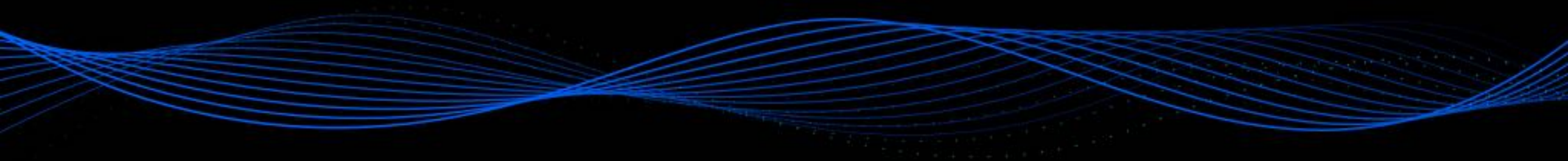
58同城Android客户端框架演进与实践

赵路平



目录


PART 01
框架演进历程

演进图

2012

快速占领市场，速度

纯Native

2013

快速发展，迭代快

Hybrid模式

2014

回归体验，满足灵活

部分动态化

2015

AllInApp，大平台，并行开发

插件框架

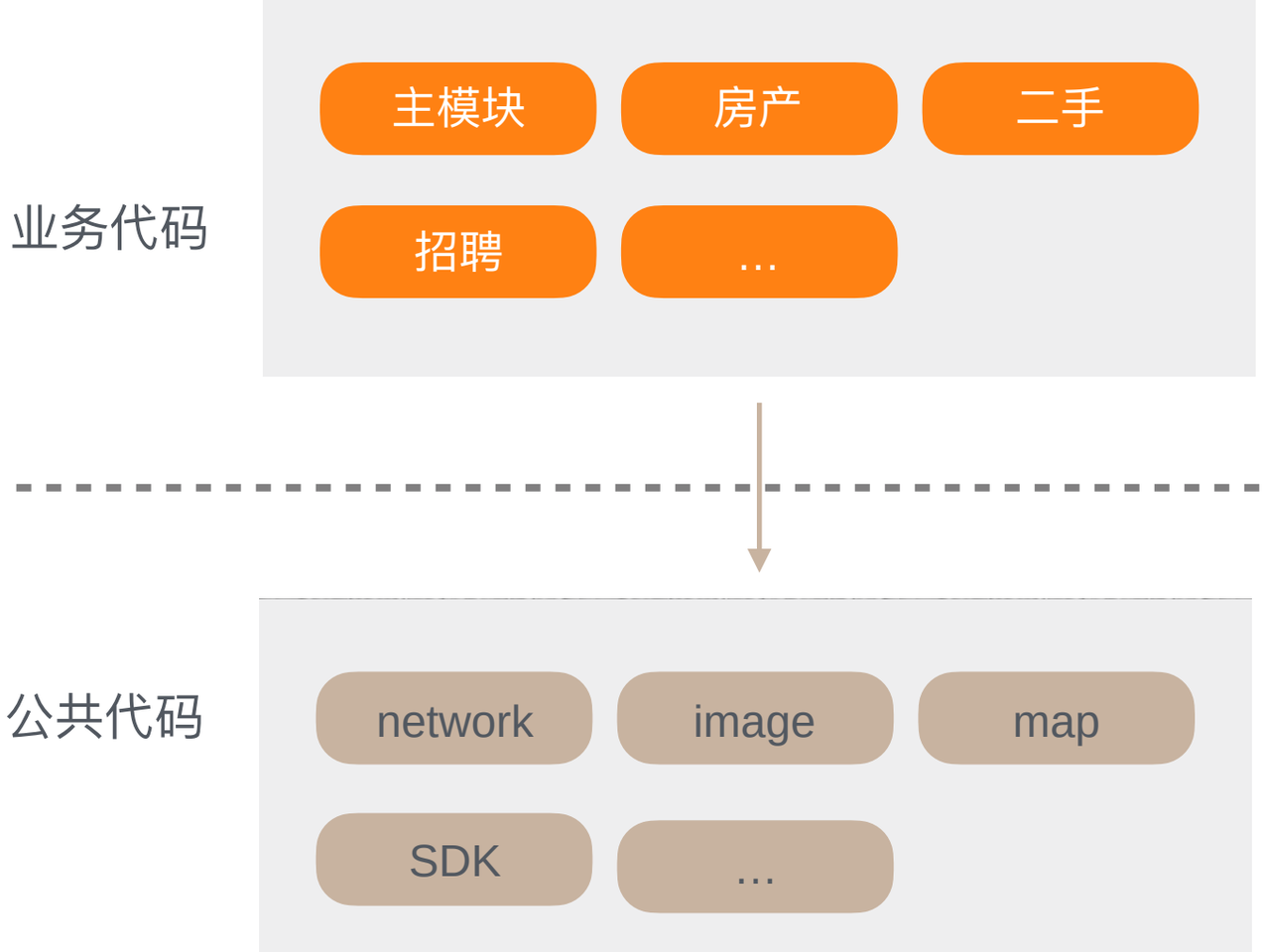
2016

可维护性，开发效率

Walle框架成型

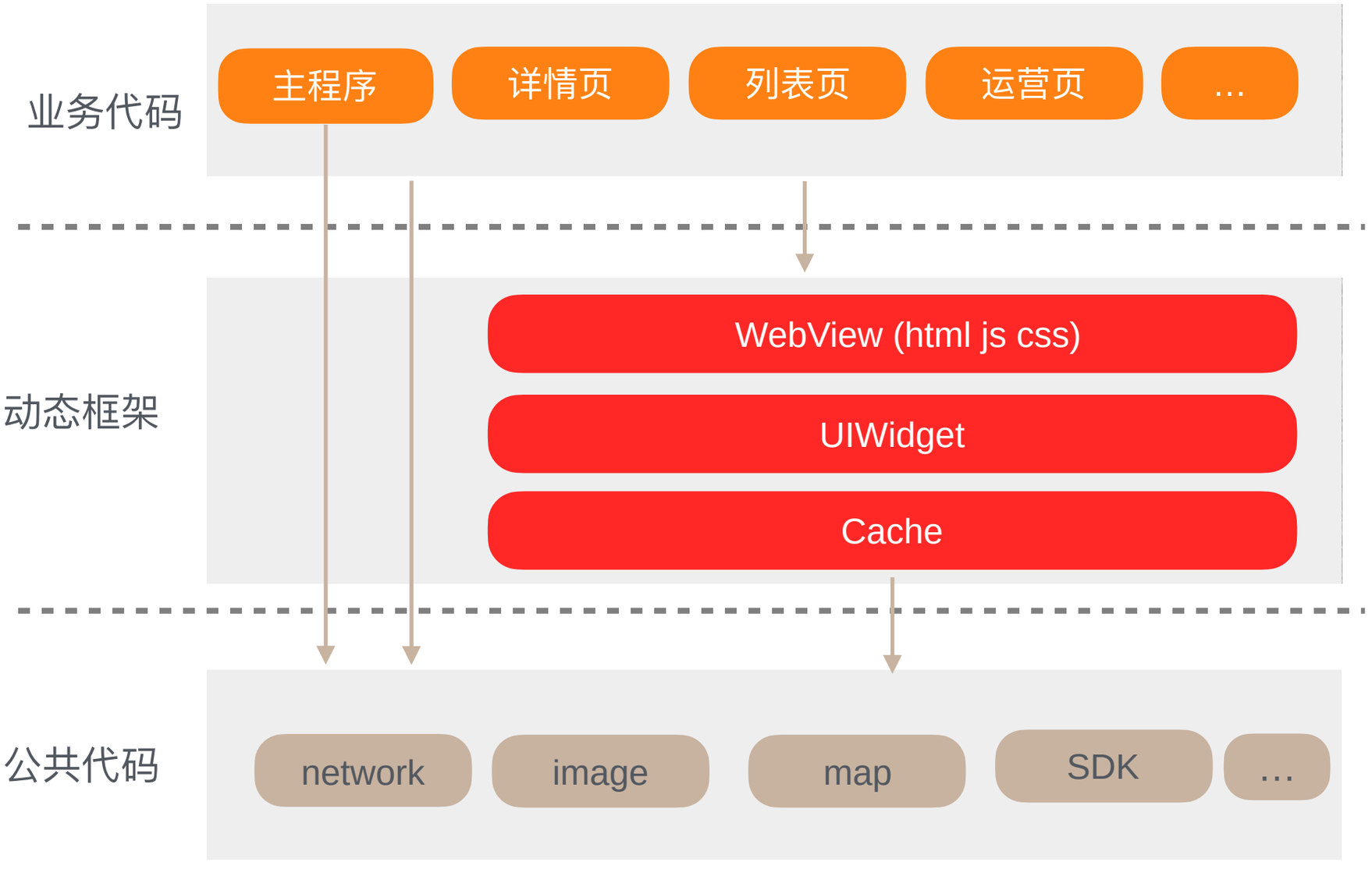
纯Native

- 模块拆分
- 基础组件
- 快速搭建框架



Hybrid模式-框架

- 业务迅速迭代
- 线上修复能力强



Hybrid模式-实践

✦加载页面速度慢?

1. 缓存（html、js、css、image）

2.3

ContentProvider

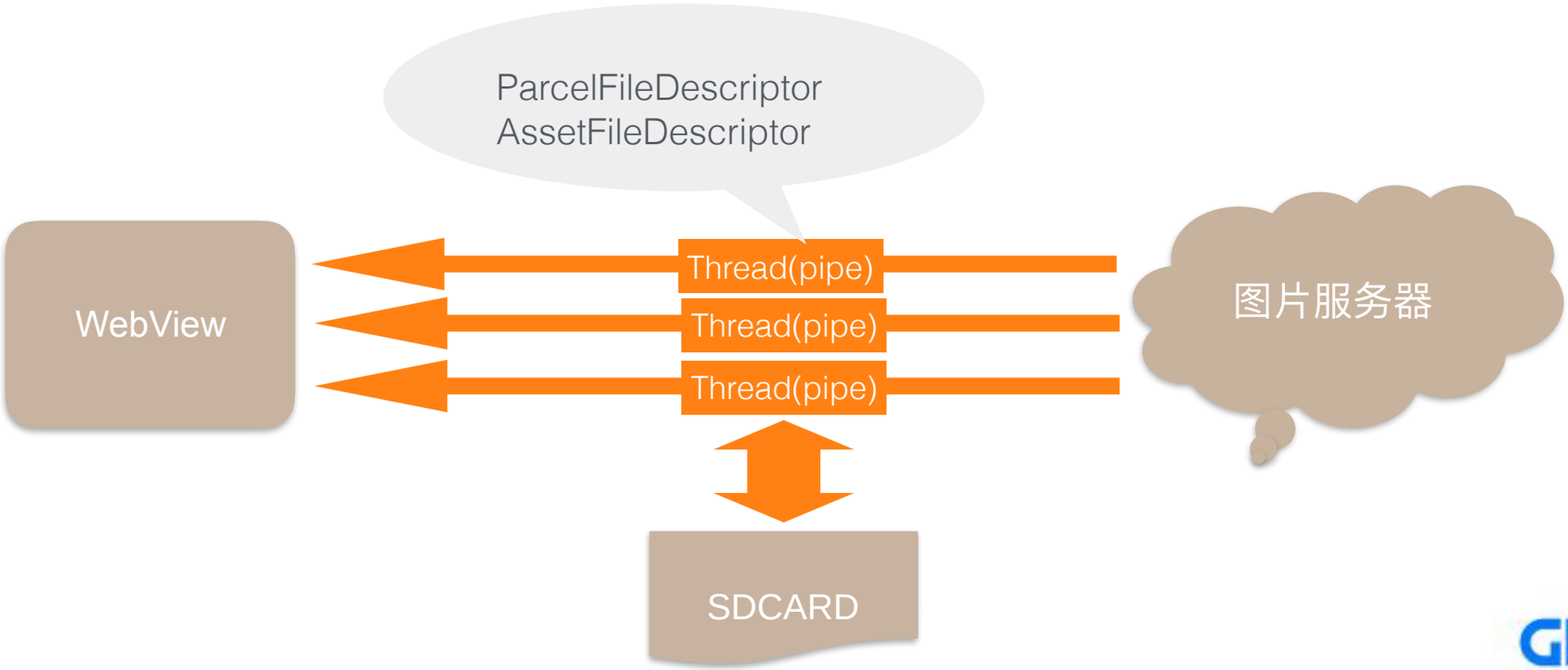
3.0

shouldInterceptRequest
(WebView view, String url)

5.0

shouldInterceptRequest
(WebView webView, WebResourceRequest webResourceRequest)

2. 并行加载



Hybrid模式-实践

✦ 如何管理缓存文件（版本号、超时间）？

1. 文件名?

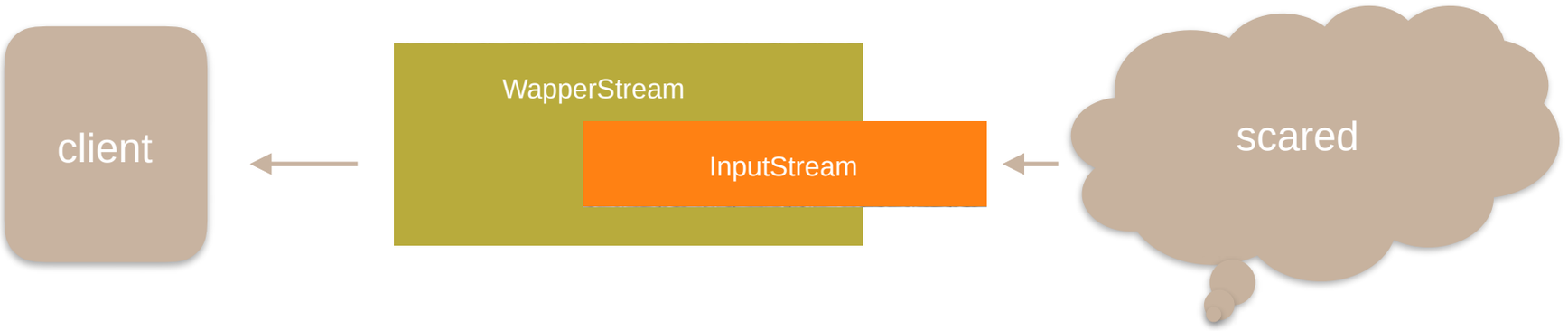
2. 数据库表?

key	value
version	0.0.1
time	2017-6-9

3. 放在文件内容中

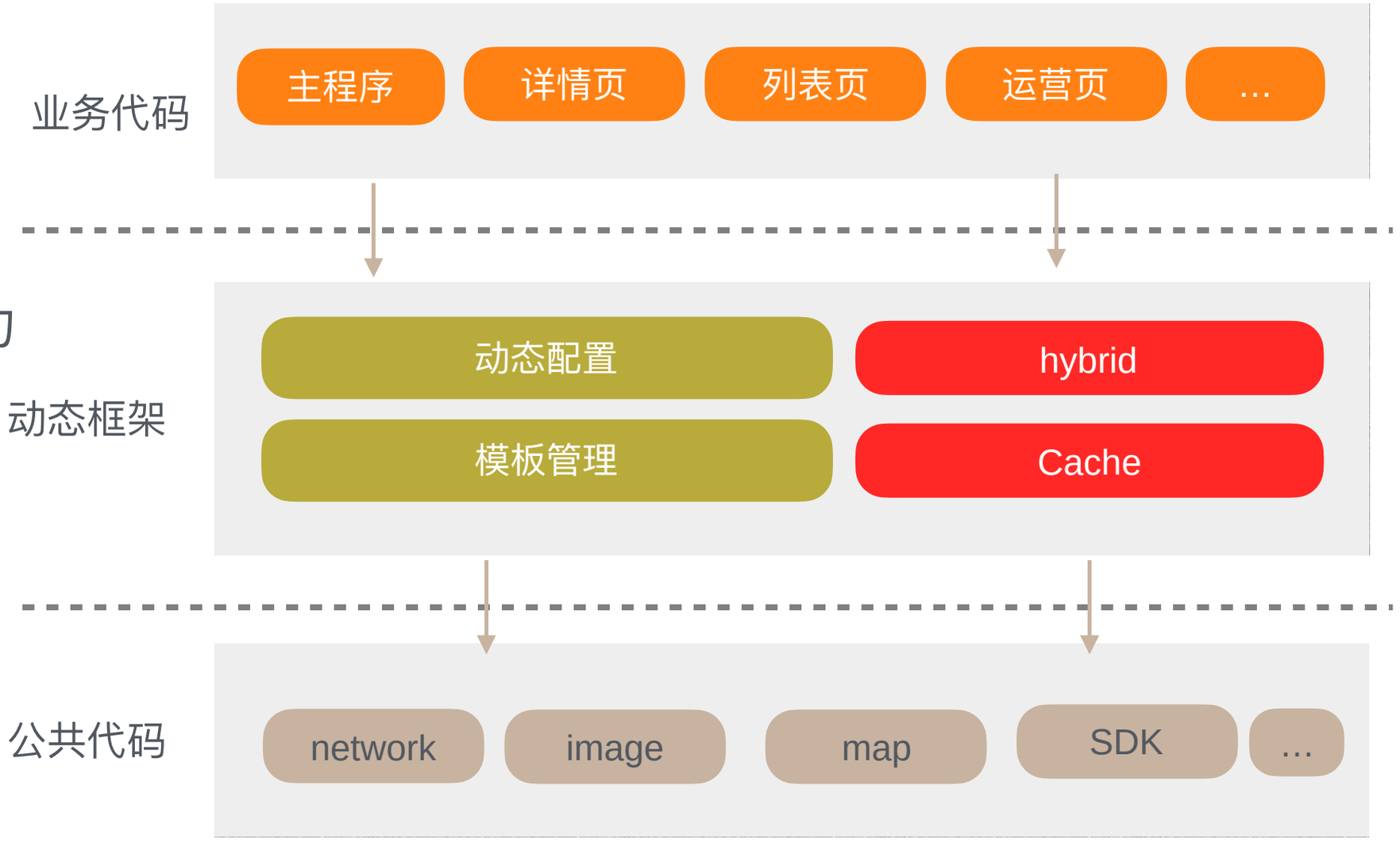
把内容转换成固定字节，保存到文件头中，如下图

读取时：



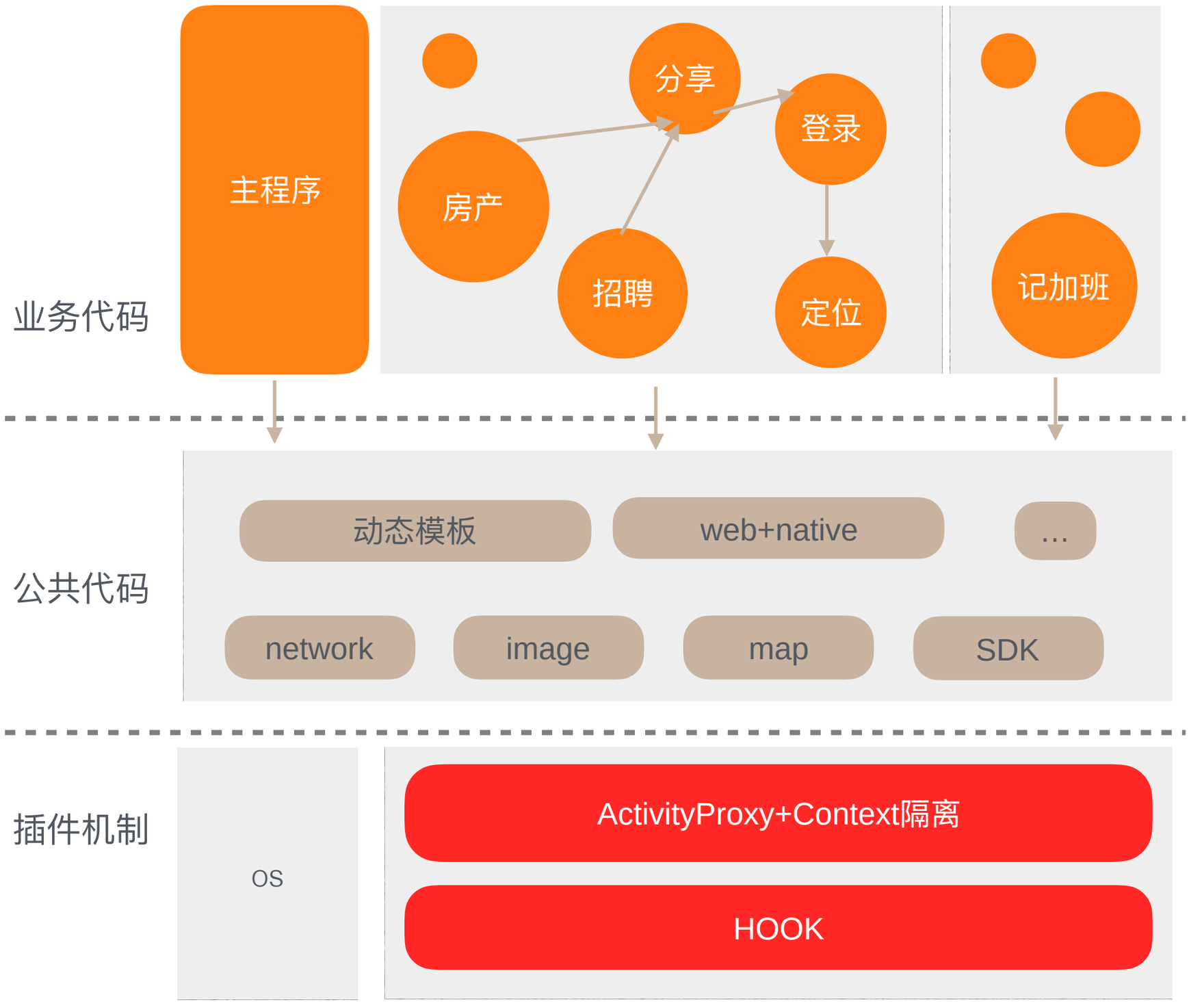
部分动态化

- 提升用户体验
- 保留了动态部署能力



插件化

- 并行开发能力
- 动态升级能力



插件化-实践

✦进程问题?

A 一个插件一个进程

会出现进程数膨胀问题，
不可接受

B 固定进程数

主业务间有依赖关系，
不能杀死进程

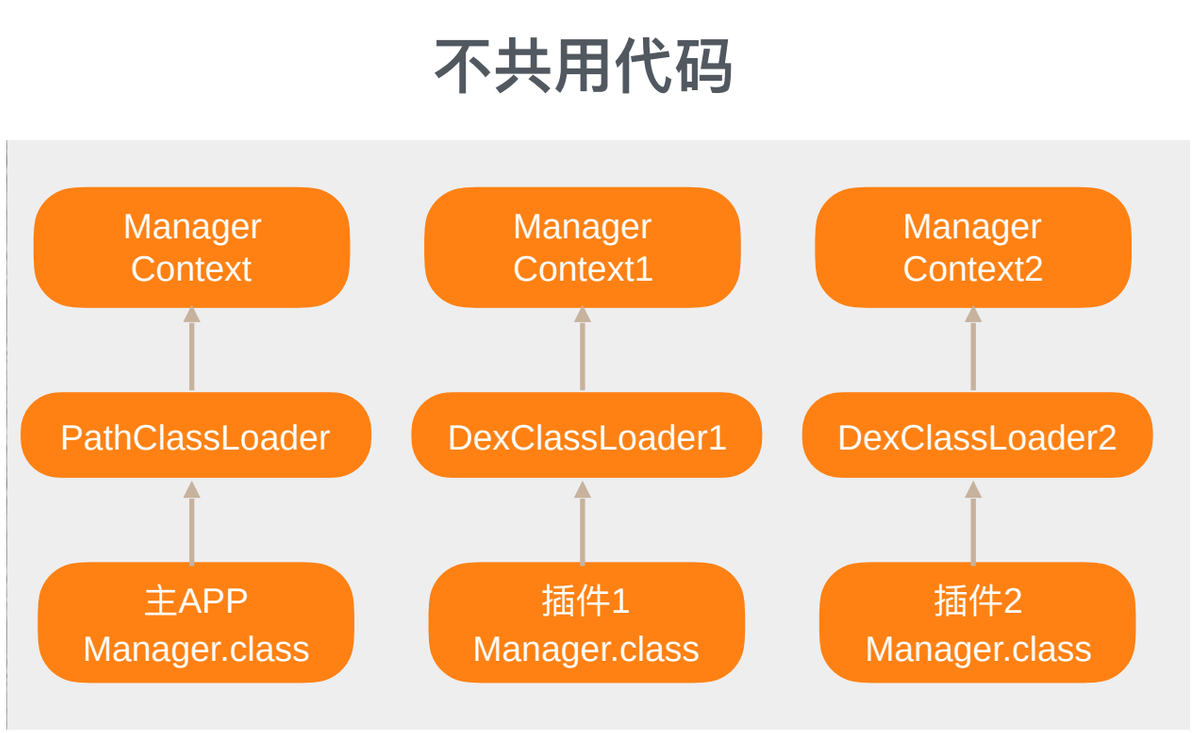
C 合并单一进程

会引起单例问题，比如
引入Fresco时资源混乱

插件化-实践

✦ 代码共用问题?

1. 单例问题



2. apk包增大

3. 插件之间交互变复杂之后联调成本增加

PART 02

组件化实践

开发周期

开发期

并行开发

提升开发效率

运行期

提高运行效率

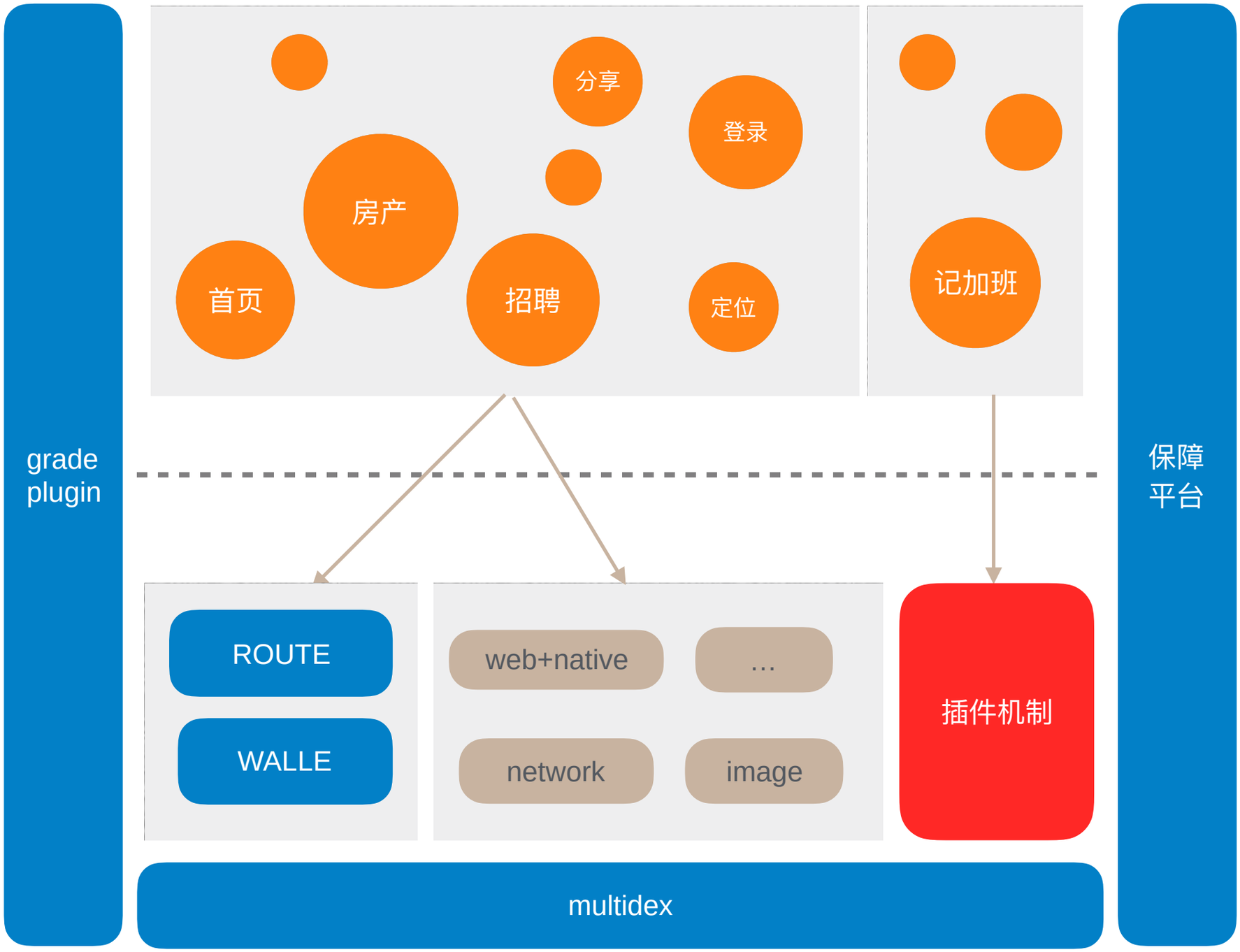
按需加载

运维期

动态升级

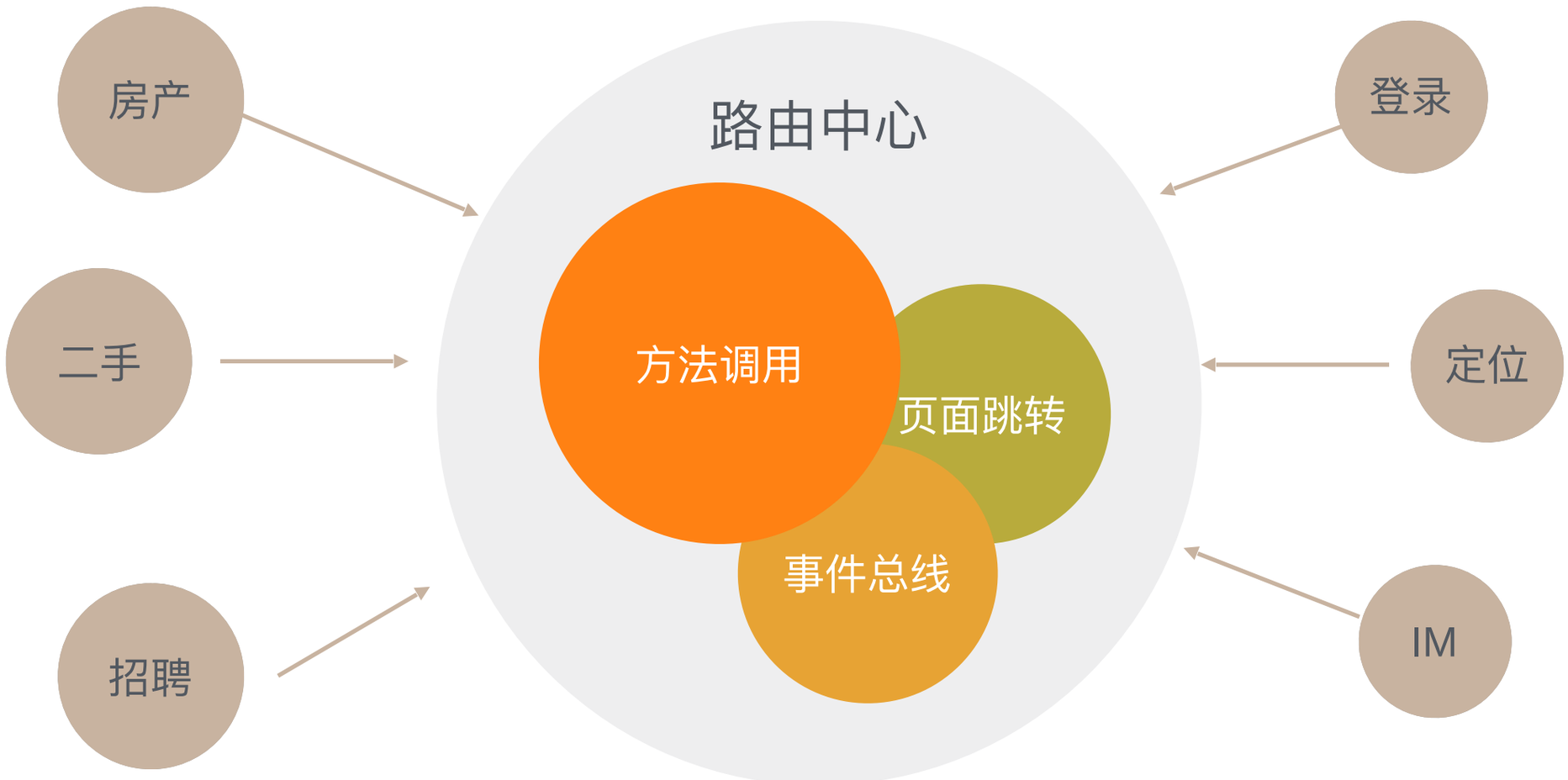
Walle框架结构

- 并行开发能力
- 降低依赖层次
- 提升编译速度



路由中心

- 为什么要有路由中心?

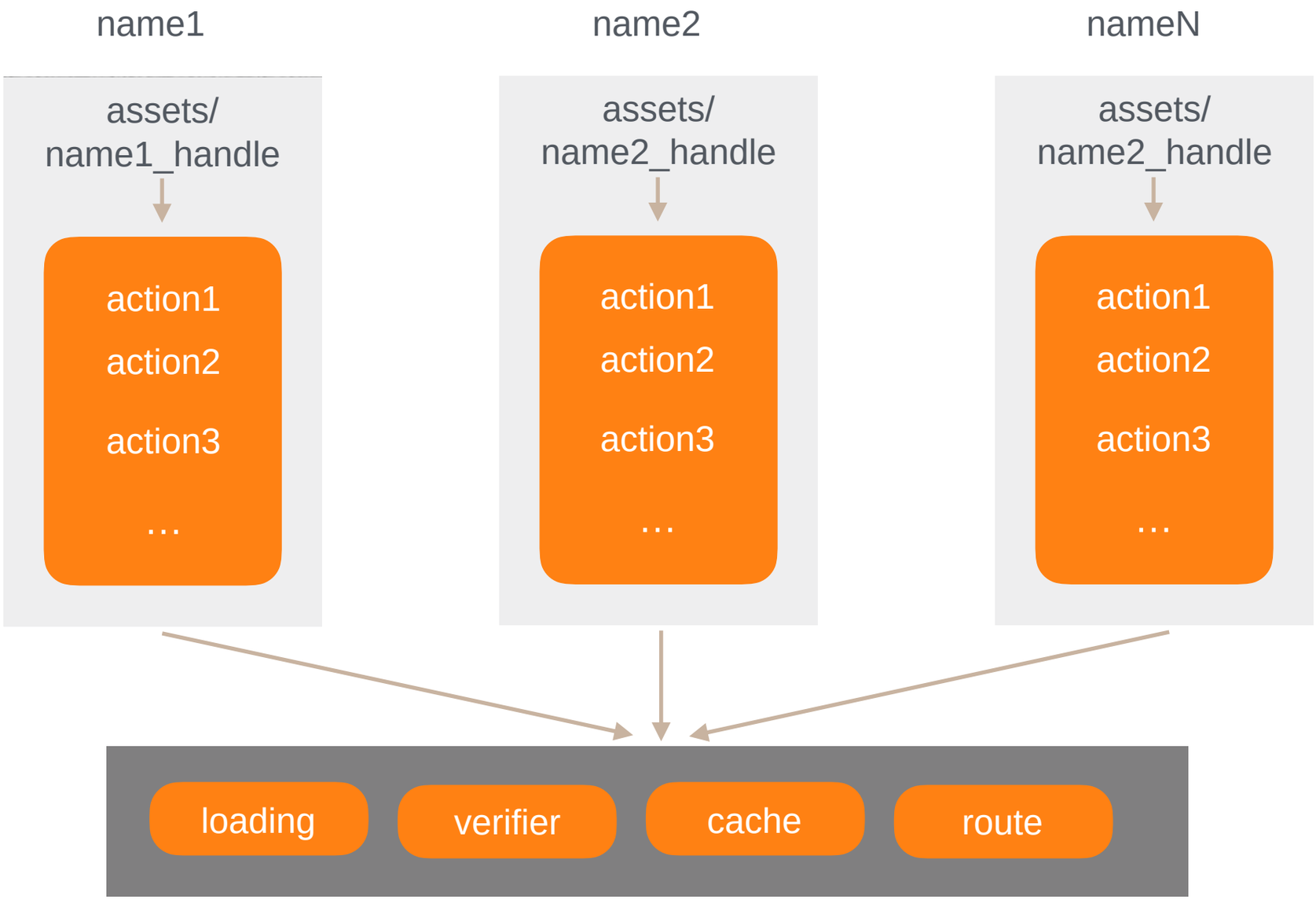


1. 代码和资源解耦，方便组件维护

2. 编译时可以随时去掉不需要的组件，提升编译速度

路由中心实现

- wbmain://authority/path?query
- wbmian://component/{name}/{action}?{query}



路由优势

client

```
public int getTaskScore(int taskId){
    Response result = Walle.route(“"wbmain://
    component/share/getTaskScore"”);
    return result.getInt(“result”)
}
```

service

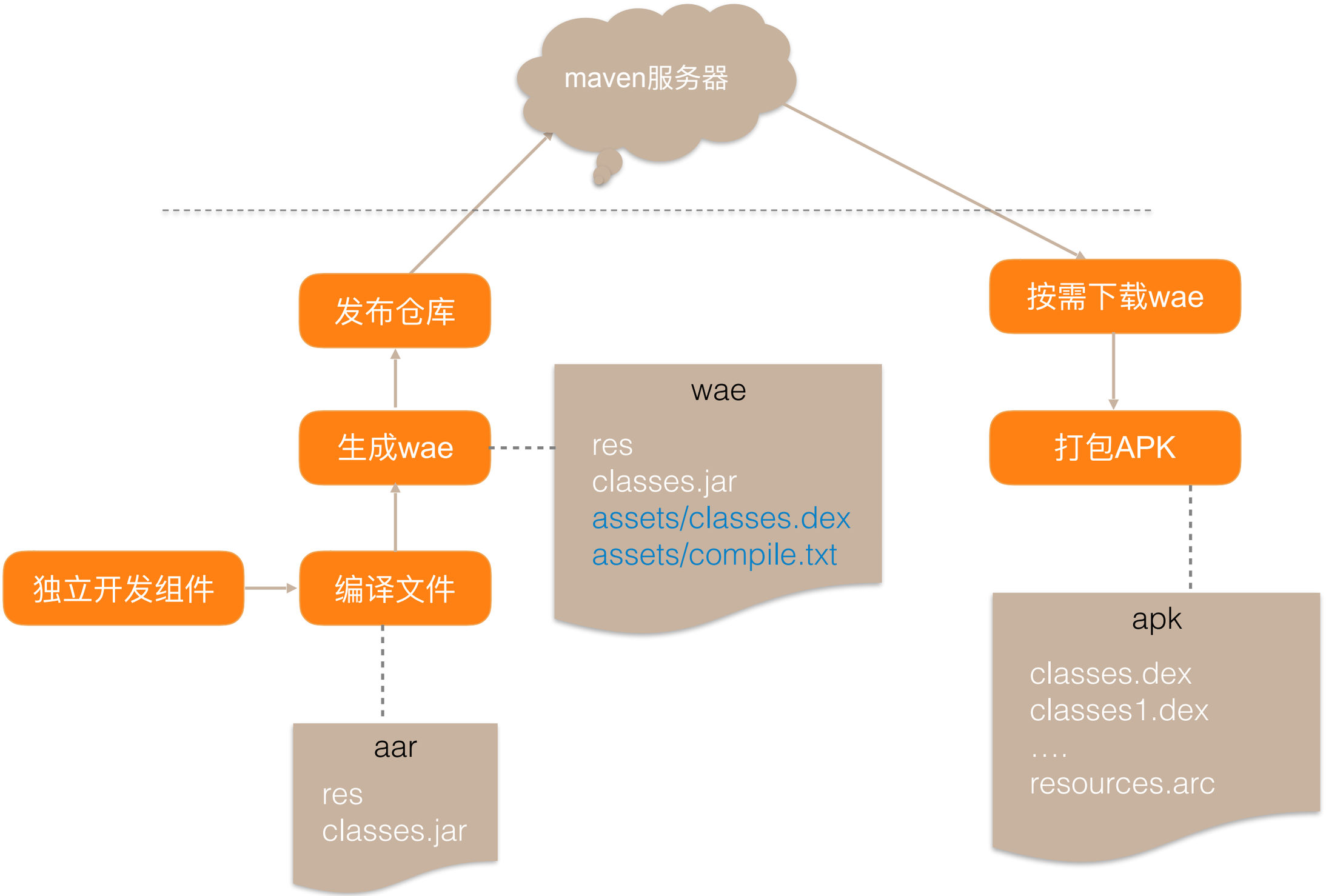
```
public class ShareHandle extends ComHandle {
    @Action(uri="wbmain://component/share/getTaskScore")
    public void getTaskScore(Context context, Request req, Response res){
        int taskid = req.getInt("taskId");
        …
        res.putInt(“result”,0);
    }
}
```

- 方法级别的同步调用
- 使用简单、开发成本小

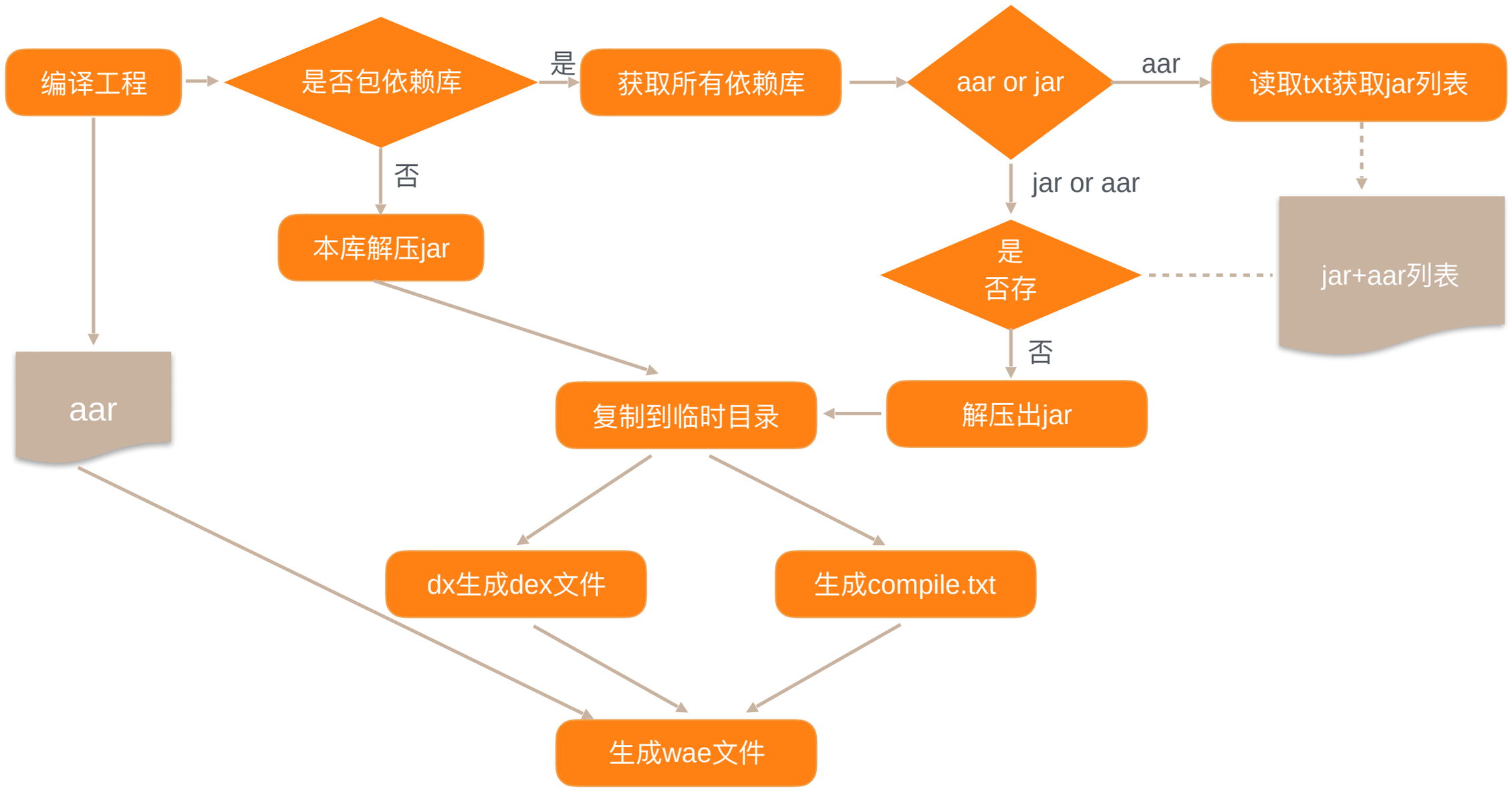
路由中心实践

路由虽好，但不能滥用

开发期流程

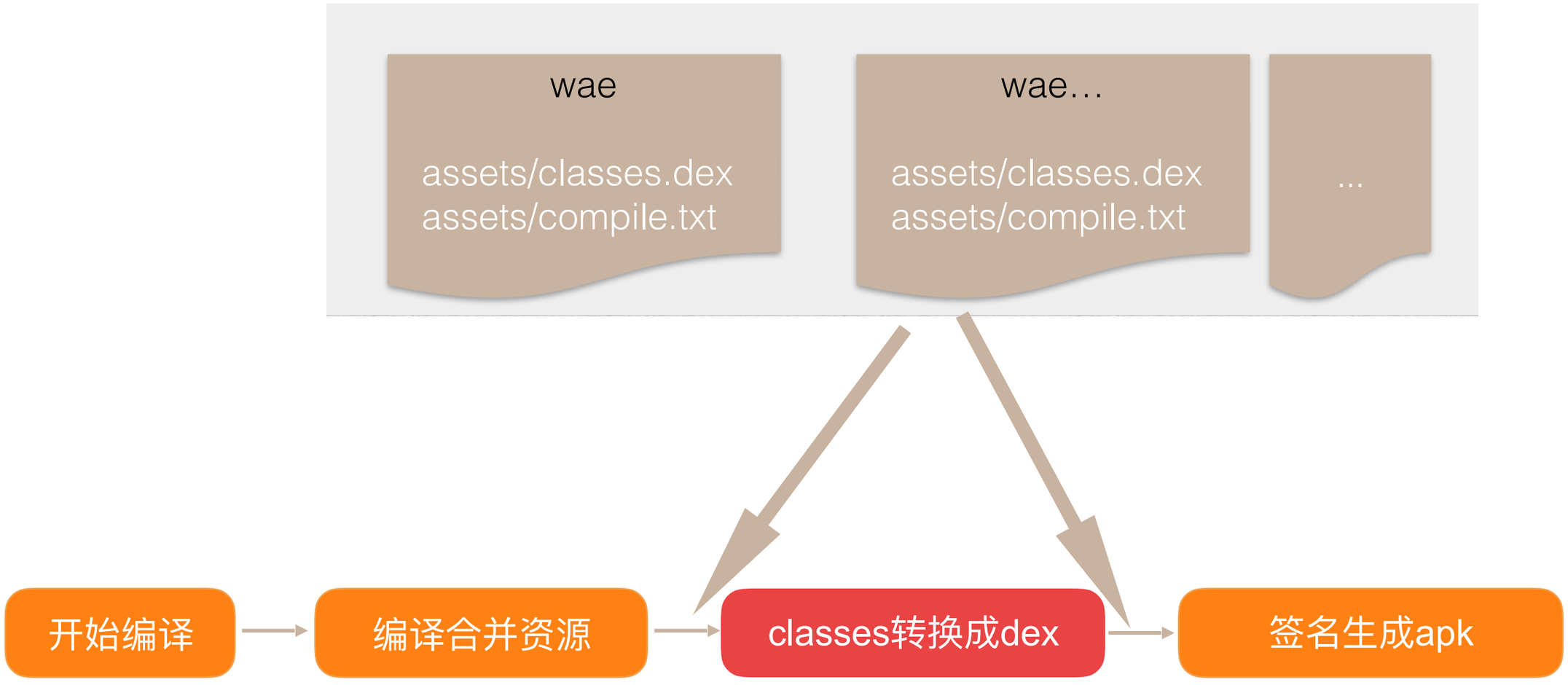


开发期-发布wae流程



- dex太大怎么解决?

开发期-打包



开发期-打包优势

实例：编译APK总用时147s，下面是各Task用时

Task	时间(s)	占比（%）
mergeDebugResources	10	11%
processDebugResourcces	7	
transformClassesWithJarMergingForDebug	7	73%
transformClassesWithMultidexlistForDebug	16	
transformClassesWithDexForDebug	85	

- 跳过合成dex后，总时长可缩短70%

Walle框架分析

	并行开发	组件化程度	解耦化	编译速度	运行效率	兼容性	动态性	开发成本
深度插件化	支持	高	高	低	中	中	高	高
容器化	支持	高	高	低	中	中	中	高
Walle	支持	高	高	高	高	高	低+高	低


PART 03
保障平台与规划

开发期-保障支撑平台

工具集

- MergeRequest
- wae发布工具
- 代码仓库管理
- ...

构建平台

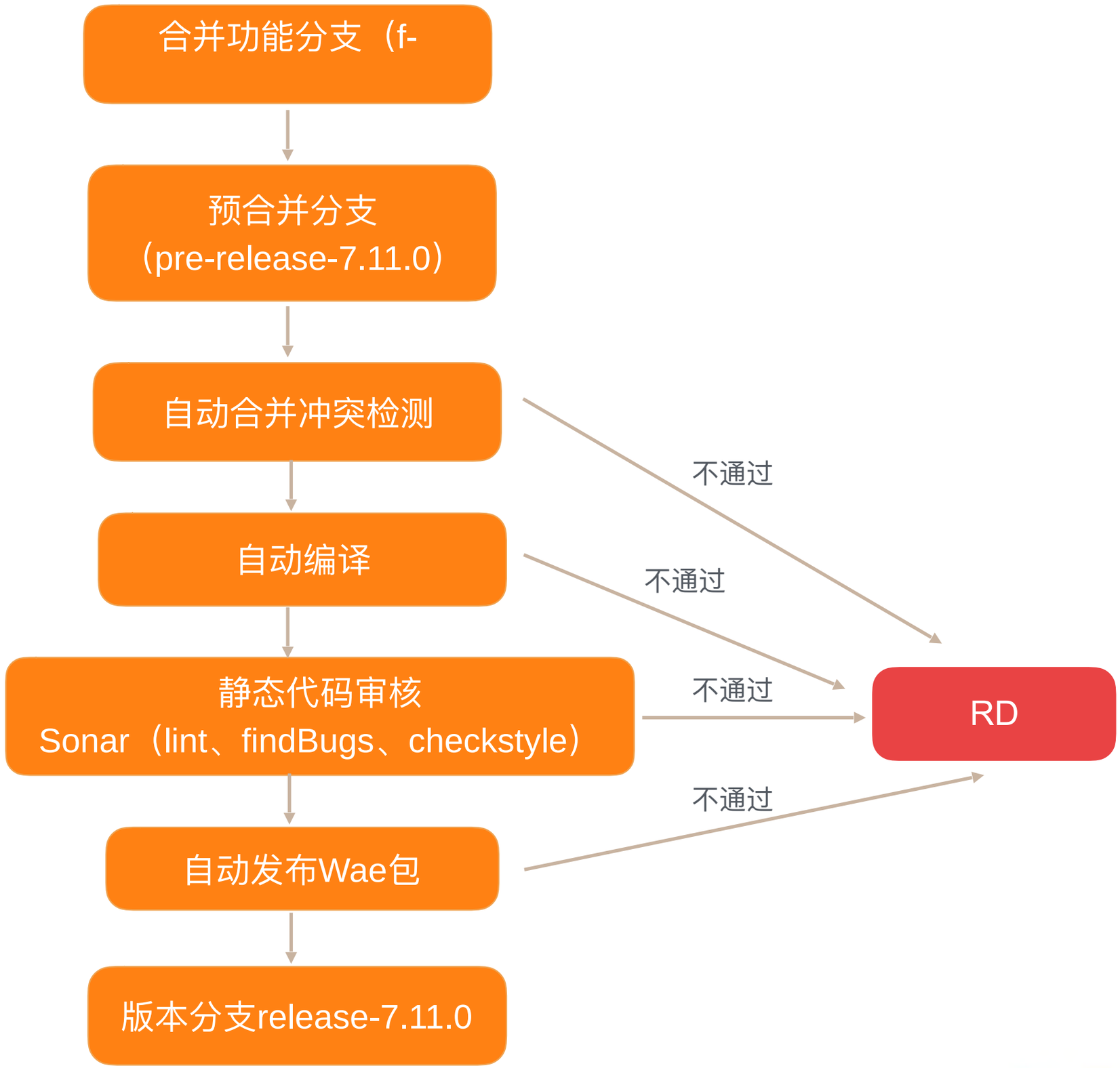
- 内置资源打包
- 代码自动检测合并
- Jenkins

质量保障

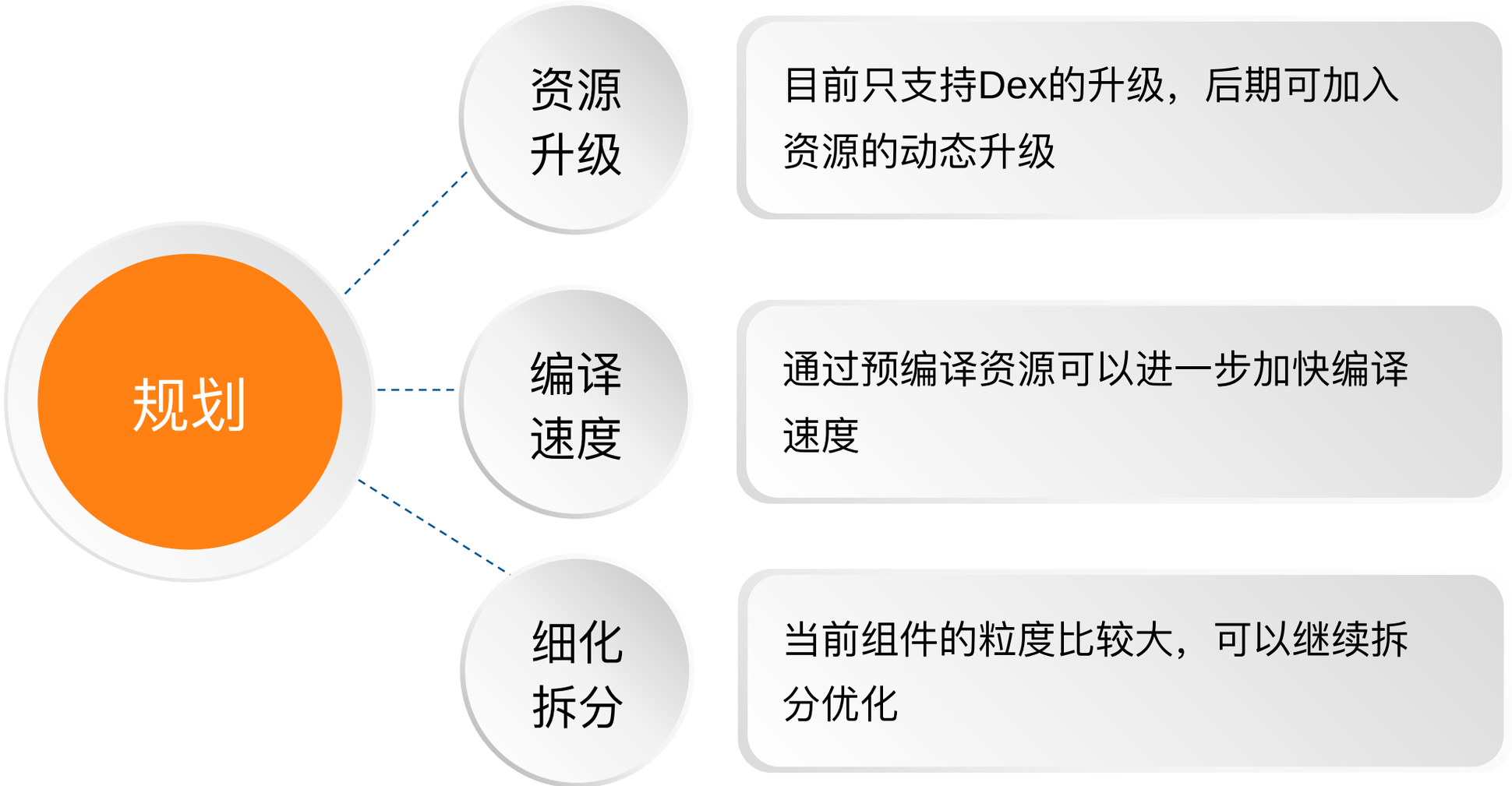
- 自动检测上线单
- 自动化测试
- Testin
- 灰度发布
- 日志系统
- Bugly

开发期-支撑平台

- 合并流程自动化
- 上线检查自动化
- 发版流程自动化



后期规划



THANKS!

GMTC
全球移动技术大会2017