Tencent 腾讯

超级补丁技术

腾讯 俞尚(anlanyu)

颠覆Android移动端的开发模式

QQ空间 QZONE.COM

个人介绍

- 俞尚，2007年加入腾讯，T3.3高级工程师
- 早年QQ浏览器开发核心骨干，自研浏览器内核，在算法和性能方面有比较深入的研究。
- 2012年开始是QQ空间Android团队Leader，负责空间技术和团队管理工作
- 主要在QQ空间架构优化，性能调优，移动网络，Android内核等方面有些研究

目录

- 一、技术背景
  - 苦逼的现状、面临的问题、超级补丁包的优势
  - 与其它补丁技术对比
  - 实际应用效果
- 二、技术原理
- 三、技术难点
- 四、其它问题

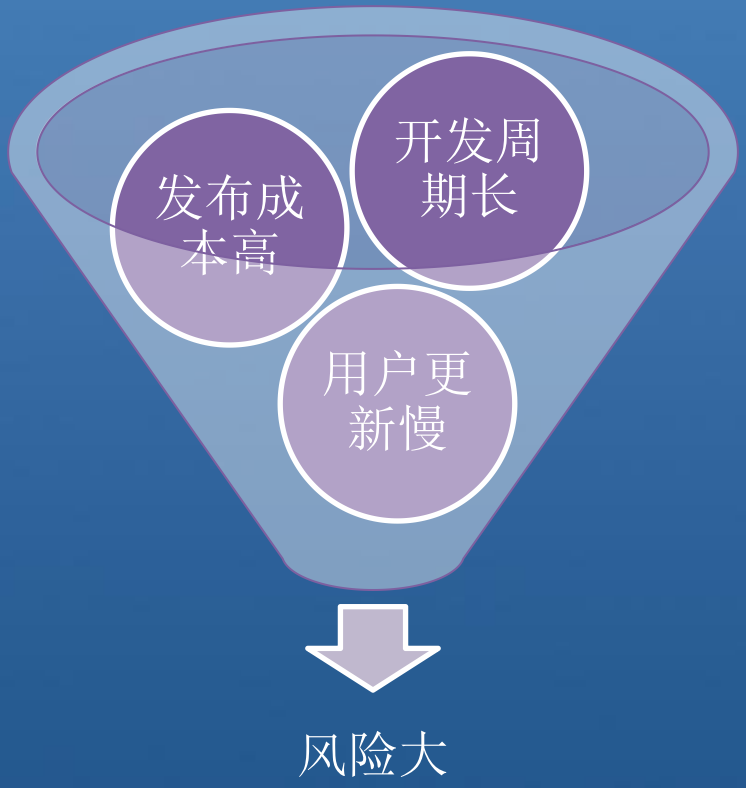
苦逼的现状

6.1版本

6.2版本

- 发布成本高，用户安装成本也高
- 版本周期长，并行开发
- 更多需求，更多加班，苦不堪言

面临的问题



“超级补丁包”来帮你

优势

- 随时发布
- 对开发透明
- 不足：需要重启App才能生效

功能

- 修复代码
- 替换资源
- 随时发布

实践

- 空间几亿用户验证
- 99%机型上验证
- 4.x-6.x版本上验证

常见的热补丁修复技术

- Dexposed & Andfix
  - 方法级的替换，不方便（举例说明）
  - 不支持资源替换
  - Dexposed在Art虚拟上很难实现
  - Andfix不支持Davik虚拟机

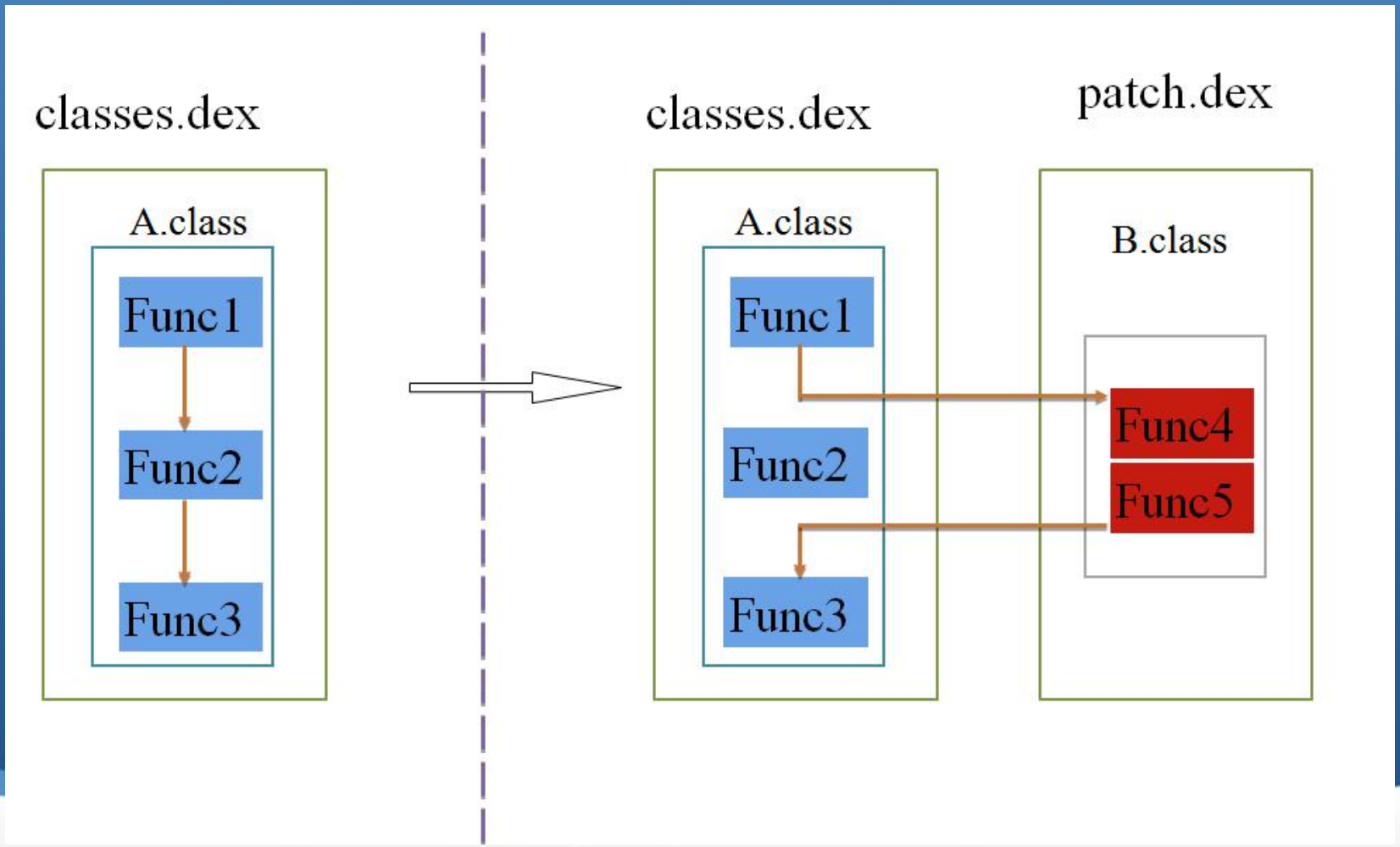
举例

```
package com.taobao.dexposed;

public class MainActivity extends Activity {

    private void showDialog() {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Dexposed sample")
                .setMessage(
                        "Please clone patchsample project to generate apk, and copy it to \"
                .setPositiveButton("ok", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {
                    }
                }).create().show();
    }
}
```

举例

```
public class DialogPatch implements IPatch {

    @Override
    public void handlePatch(final PatchParam arg0) throws Throwable {
        Class<?> cls = null;
        try {
            cls= arg0.context.getClassLoader()
                .loadClass("com.taobao.dexposed.MainActivity");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            return;
        }
         DexposedBridge.findAndHookMethod(cls, "showDialog",
            new XC_MethodReplacement() {
            @Override
            protected Object replaceHookedMethod(MethodHookParam param) throws Throwable {
                Activity mainActivity = (Activity) param.thisObject;
                AlertDialog.Builder builder = new AlertDialog.Builder(mainActivity);
                builder.setTitle("Dexposed sample")
                        .setMessage("The dialog is shown from patch apk!")
                        .setPositiveButton("ok", new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int whichButton) {
                            }
                        }).create().show();
                return null;
            }
        });
    }

}
```

Dexposed



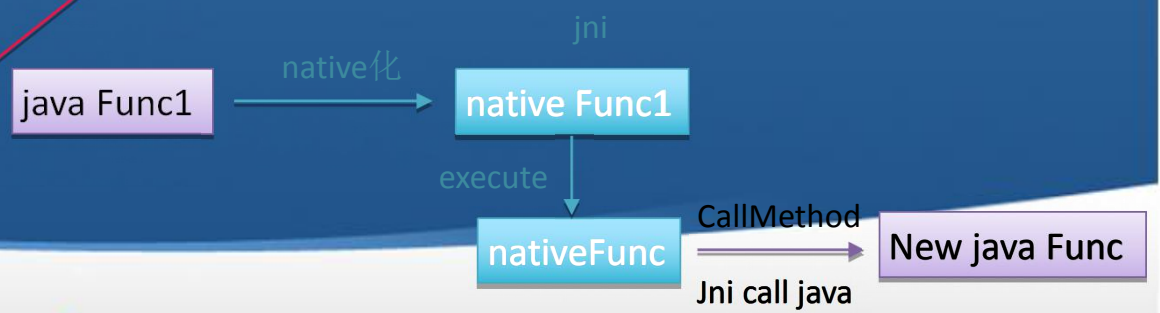
Dexposed

```
struct ClassObject : Object {
    …
    …
    int          directMethodCount;
    Method*      directMethods;
    int          virtualMethodCount;
    Method*      virtualMethods;
…
…
}
```

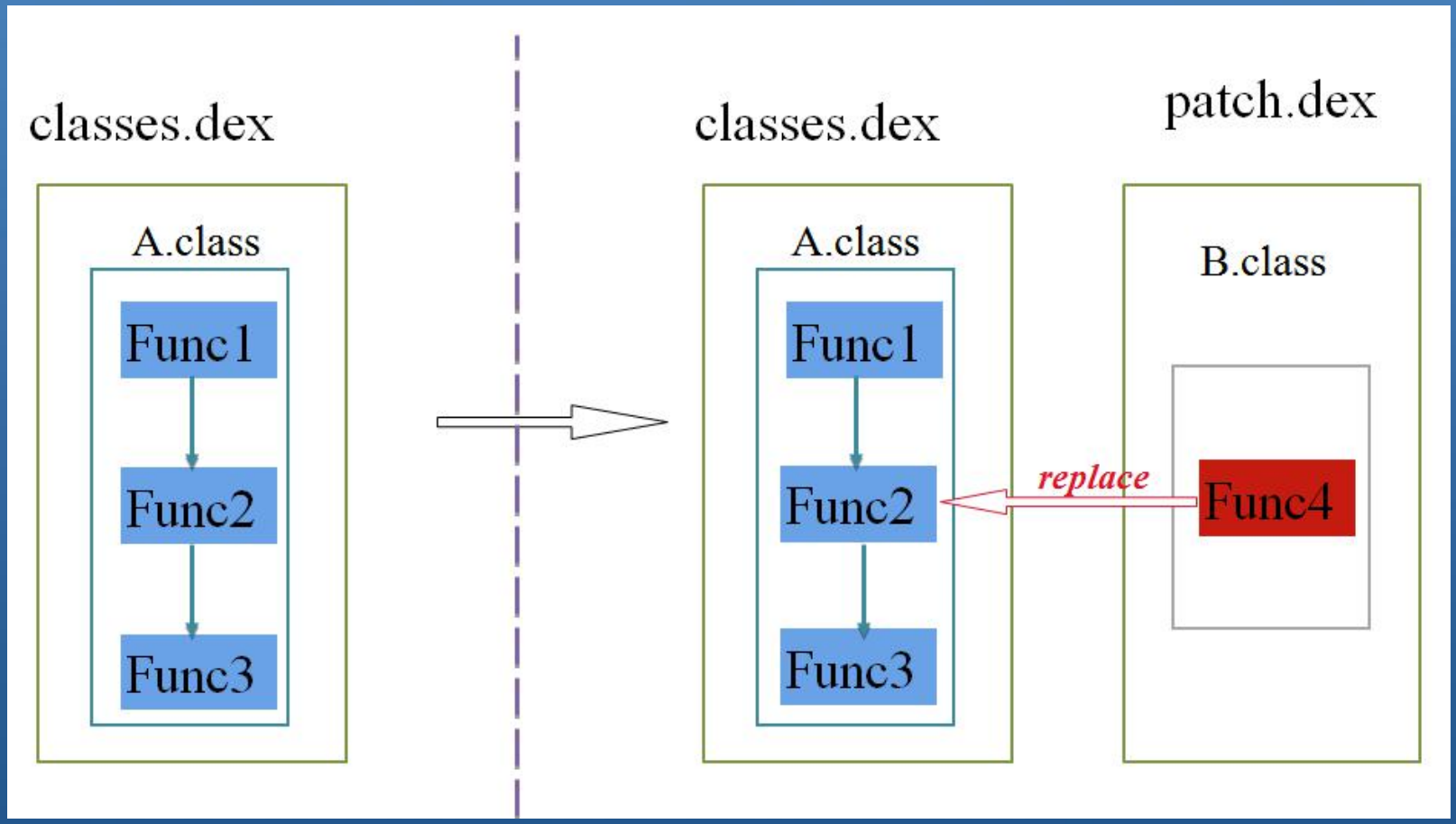
```
struct Method{
  ClassObject*    clazz;
    u2            accessFlags;
    u2            methodIndex;
    u2            registersSize;
    u2            outsSize;
    u2            insSize;
    DexProto      prototype;
    const char*   shorty;
    const u2*     insns;
    int           jniArgInfo;
    DalvikBridgeFunc nativeFunc;
    bool fastJni;
    bool noRef;
    bool shouldTrace;
    const RegisterMap* registerMap;
}
```

- java Method：
  - accessFlags 的值是1，2，4，8等
  - insSize >0
  - insns 指向dex cache中的指令入口

- Jni native Method：
  - accessFlags == ACC_NATIVE
  - nativeFunc 为真实的方法入口指针



AndFix



AndFix

核心代码：

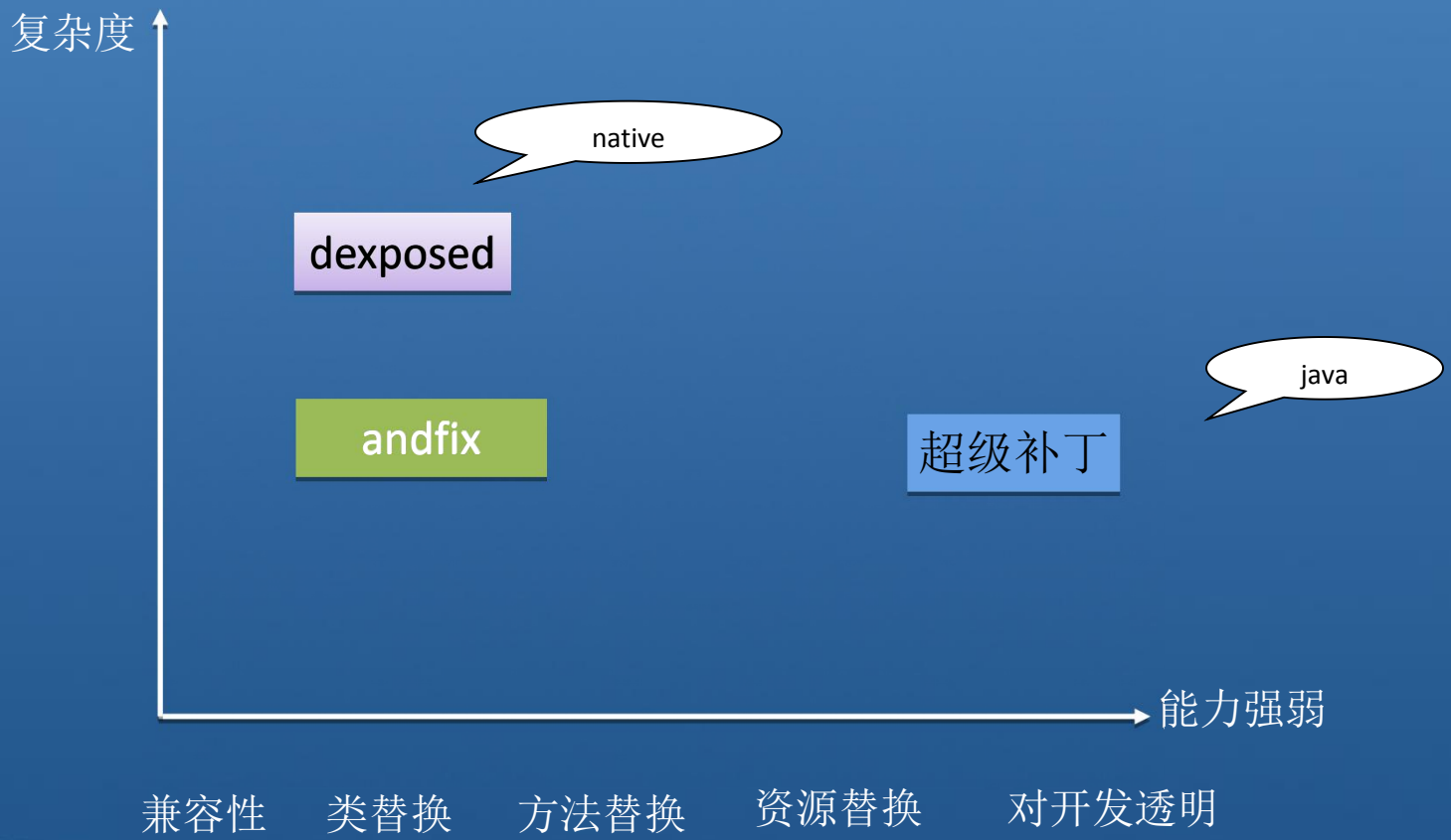
```
art::mirror::ArtMethod* smeth =
                    (art::mirror::ArtMethod*) env->FromReflectedMethod(src);

art::mirror::ArtMethod* dmeth =
                    (art::mirror::ArtMethod*) env->FromReflectedMethod(dest);

memcpy(smeth, dmeth, sizeof(art::mirror::ArtMethod));
```

补丁技术对比

	超级补丁包	Dexposed	Andfix
类替换	√	×	×
方法替换	√	√	√
资源替换	√	×	×
davik	√	√	×
art	√	×	√
即时生效	×	√	√
开发透明	√	×	×
复杂度	2	4	2
兼容性	5	3	3

- 优势：
  - 支持类替换，对开发透明
  - 支持资源替换
  - 同时支持Davik和Art虚拟机
  - 复杂度低，兼容性好（Java层实现）
- 缺点：
  - 需要重启才能生效
  - Android系统代码不能打补丁

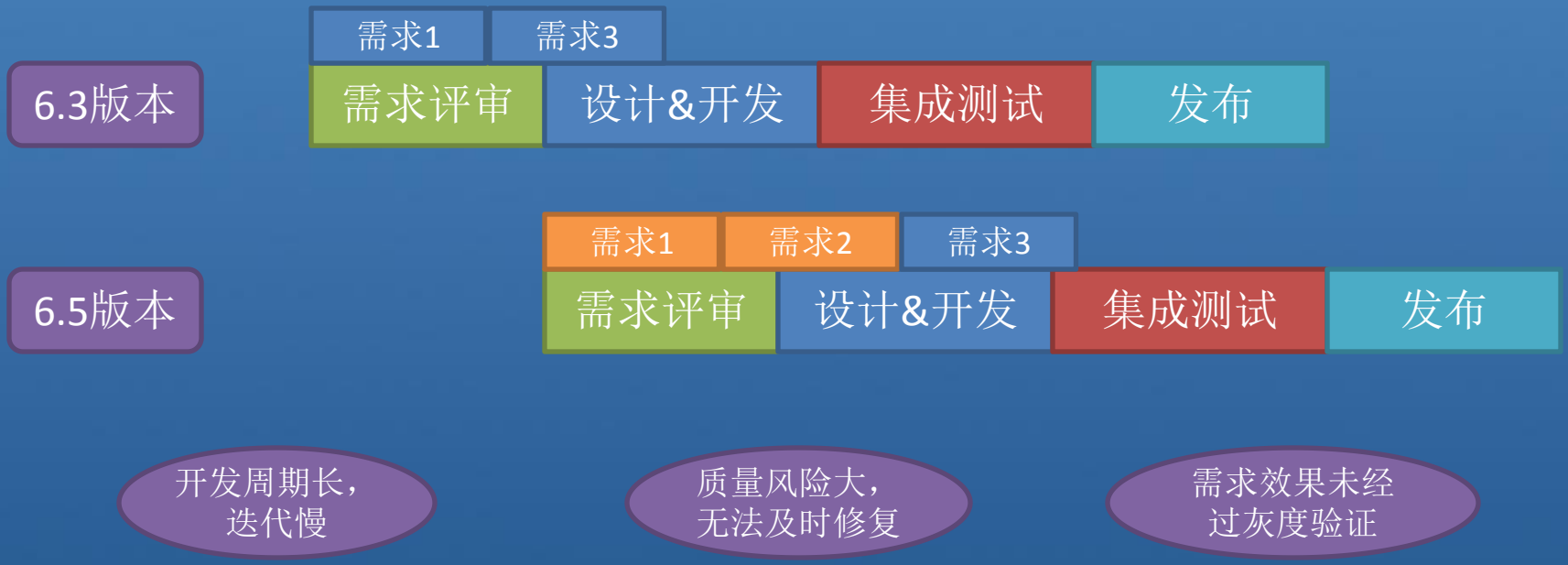
补丁技术对比



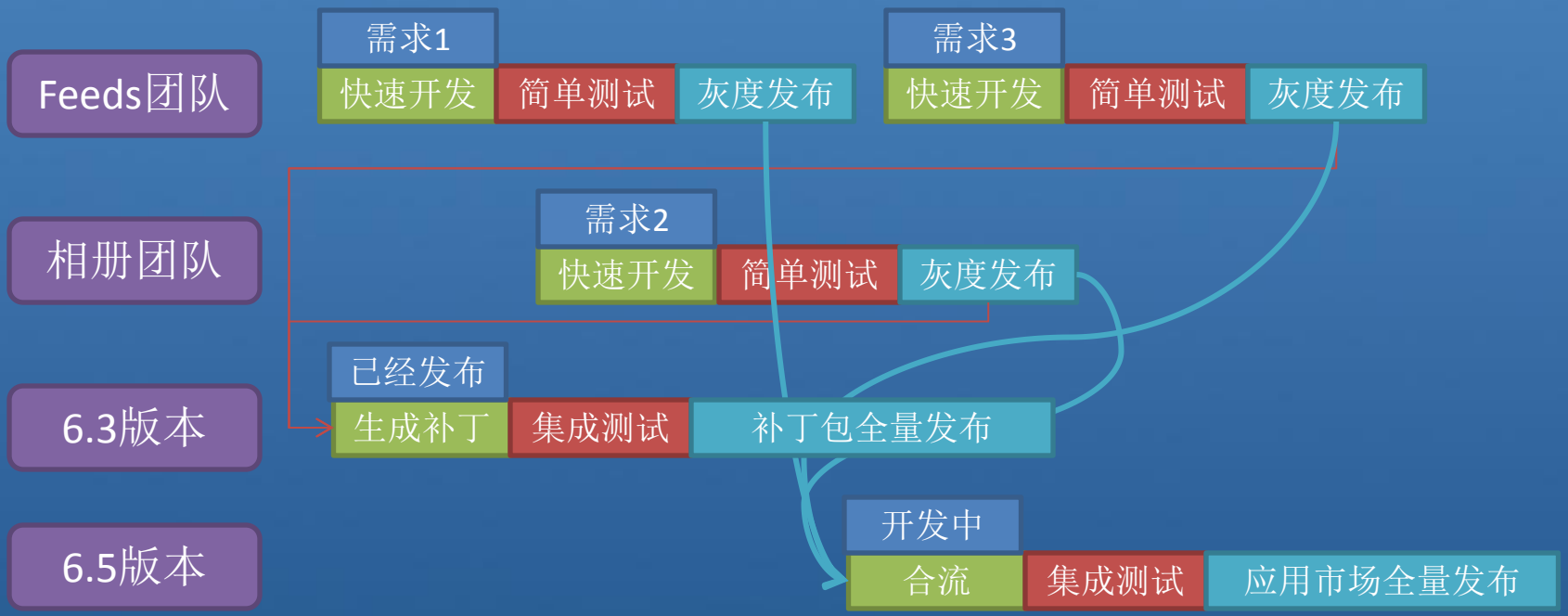
实际效果

- 更新速度：
  - 只要用户上线 ， 就会更新
- 更新成功率
  - 99%
- 安装包减少
  - 75%
- 无需发布
  - 不用上应用市场，节省人力
- 不用安装
  - 用户无感知

原来的开发模式



颠覆了原来的开发模式



开发周期短，需求独立发布

质量风险小，有问题可以随时回滚

需求效果经过灰度验证后

目录

- 一、技术背景
- 二、技术原理
  - 从Patch中加载Class
  - 从Patch中加载资源
  - 怎么生成补丁包
- 三、技术难点
- 四、其它问题

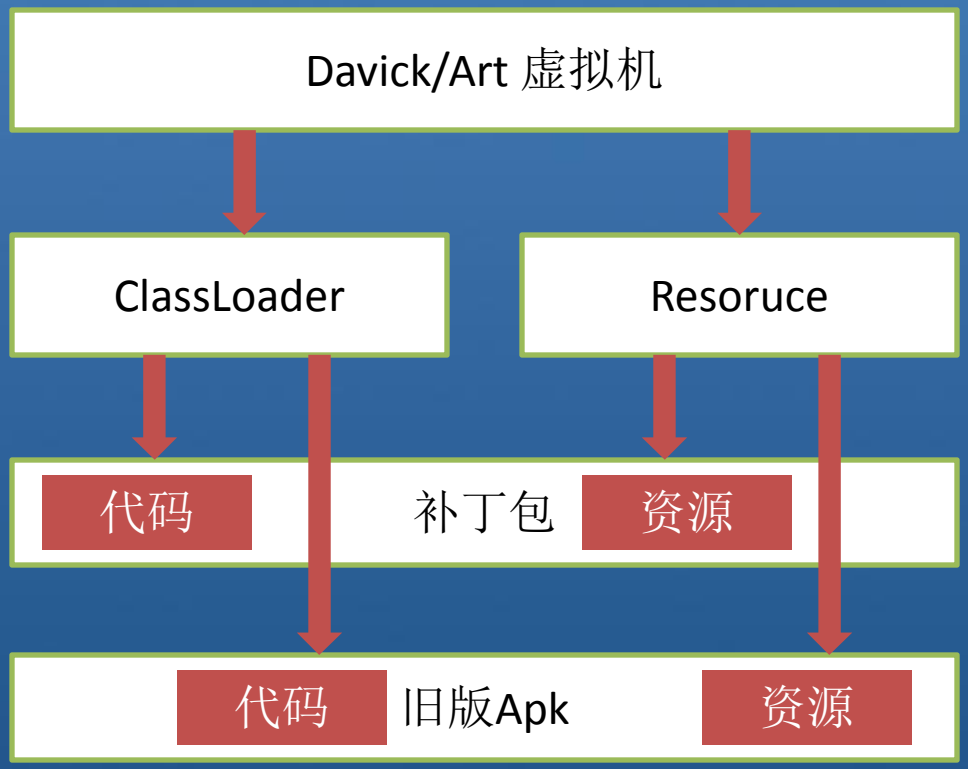
Davik/Art的能力

```
public class MyActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        setContentView(com.qzone.R.layout.qz_widget_tag_view);
        final Button button = (Button) findViewById(com.qzone.R.id.about_qzone_new);

        Drawable drawable = getResources().getDrawable(android.R.drawable.arrow_up_float);
        button.setBackgroundDrawable(drawable);
    }
}
```

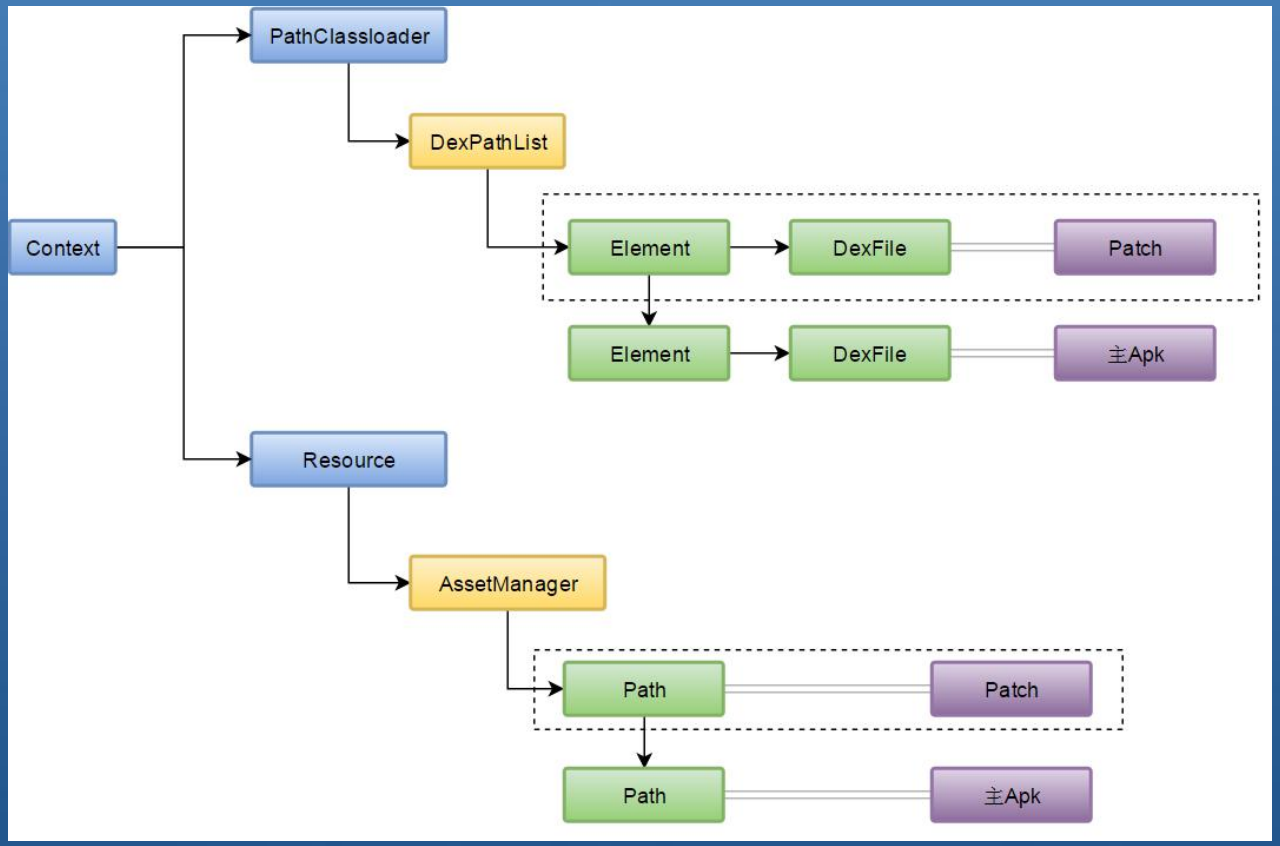
- 能从多个APK加载Class吗？
- 能从多个Apk加载Resource吗？
- 如果有重复的资源和Class虚拟机怎么处理？

超级补丁包的原理

- 加载补丁包

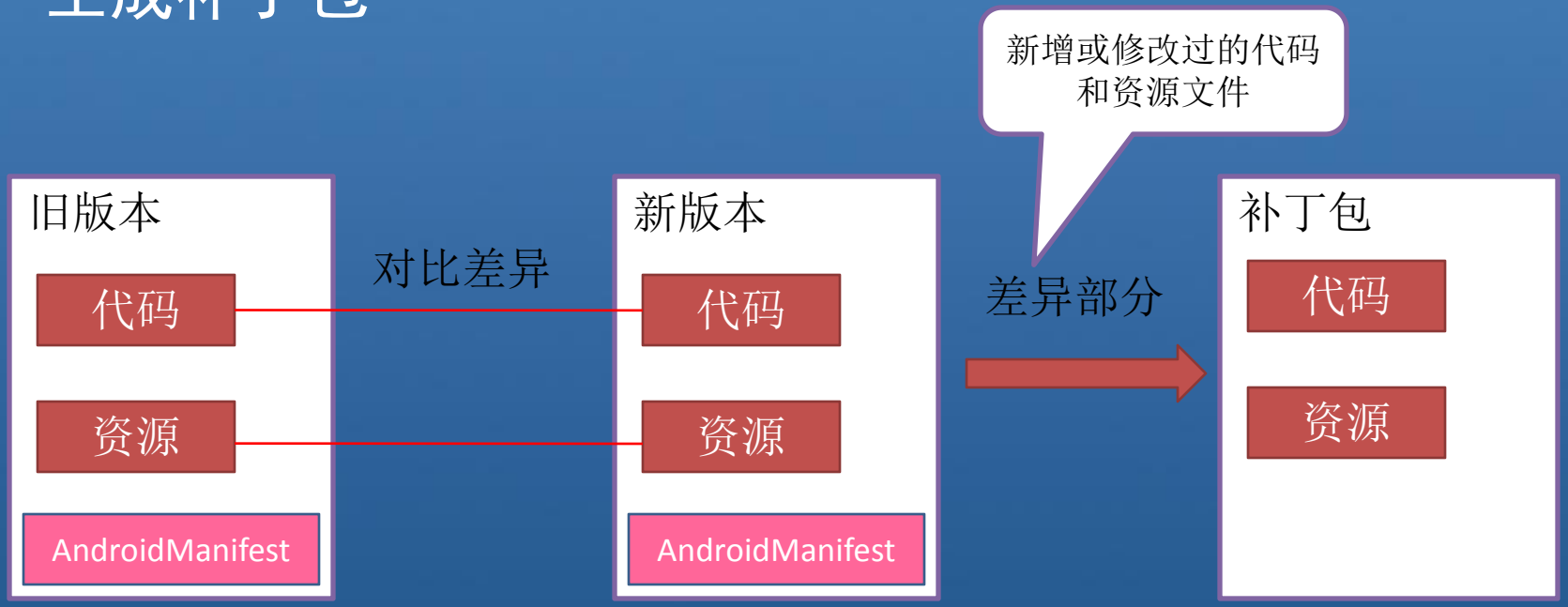


Context加载资源和代码

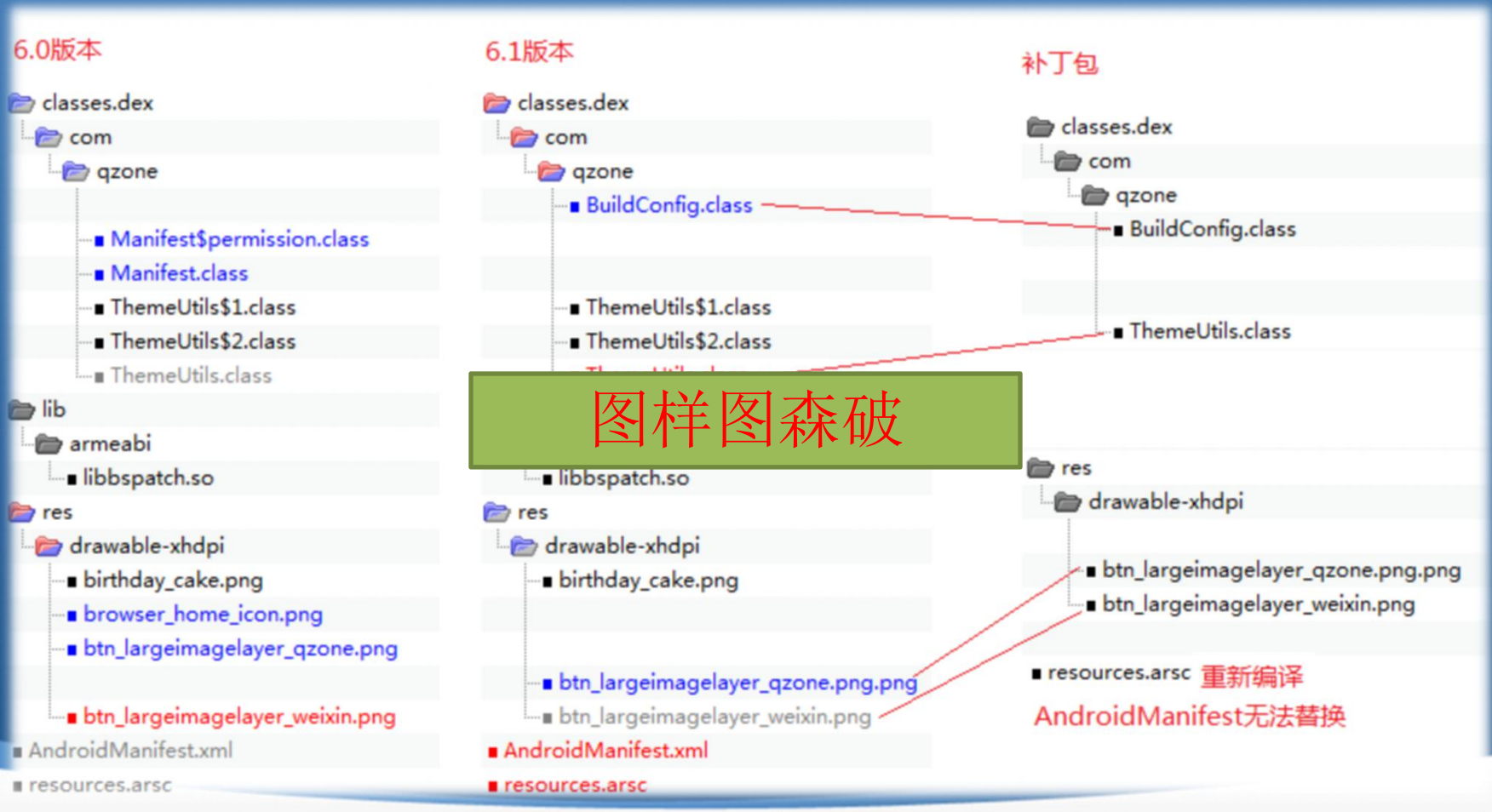


超级补丁包原理

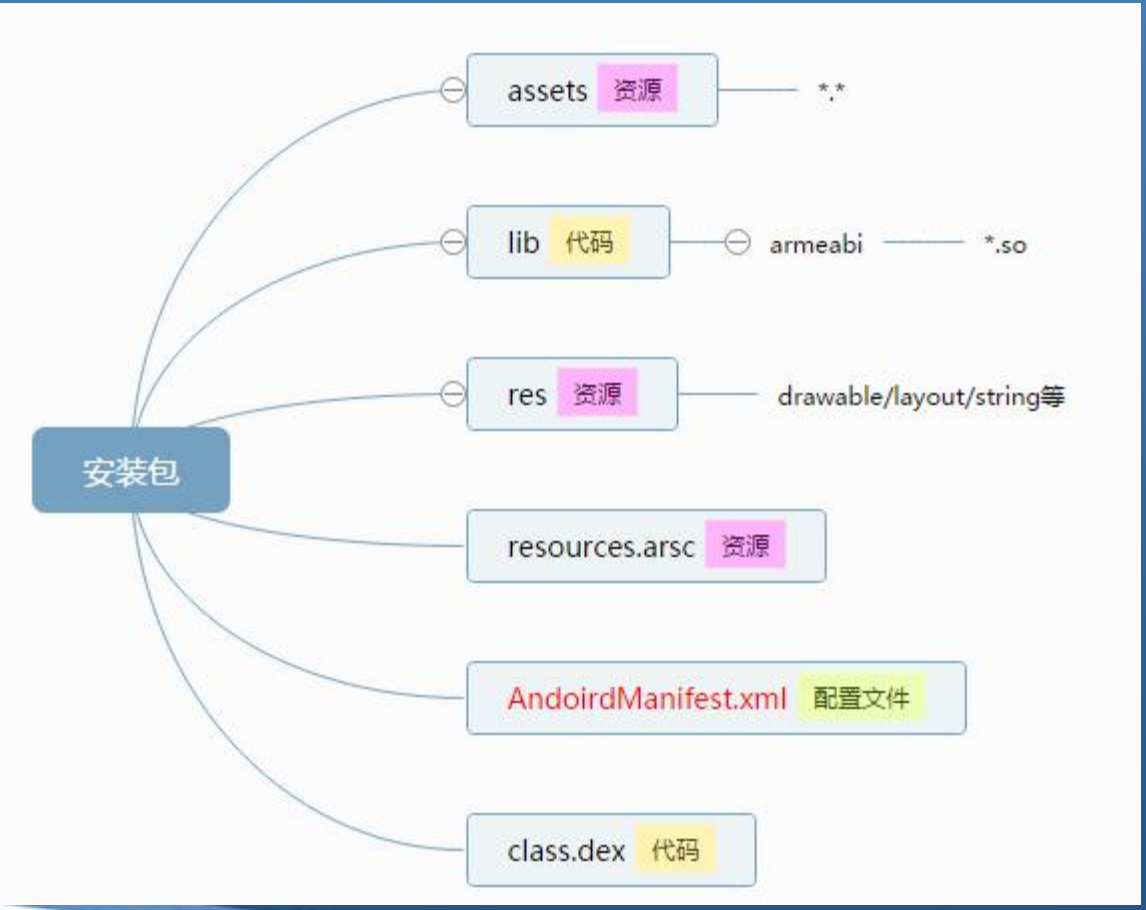
- 生成补丁包



举例


6.0版本
classes.dex
com
qzone
Manifest$permission.class
Manifest.class
ThemeUtils$1.class
ThemeUtils$2.class
ThemeUtils.class
lib
armeabi
libbspatch.so
res
drawable-xhdpi
birthday_cake.png
browser_home_icon.png
btn_largeimagelayer_qzone.png
btn_largeimagelayer_weixin.png
AndroidManifest.xml
resources.arsc
6.1版本
classes.dex
com
qzone
BuildConfig.class
ThemeUtils$1.class
ThemeUtils$2.class
libbspatch.so
res
drawable-xhdpi
birthday_cake.png
btn_largeimagelayer_qzone.png.png
btn_largeimagelayer_weixin.png
AndroidManifest.xml
resources.arsc
图样图森破
补丁包
classes.dex
com
qzone
BuildConfig.class
ThemeUtils.class
res
drawable-xhdpi
btn_largeimagelayer_qzone.png.png
btn_largeimagelayer_weixin.png
resources.arsc 重新编译
AndroidManifest无法替换

哪些能打补丁

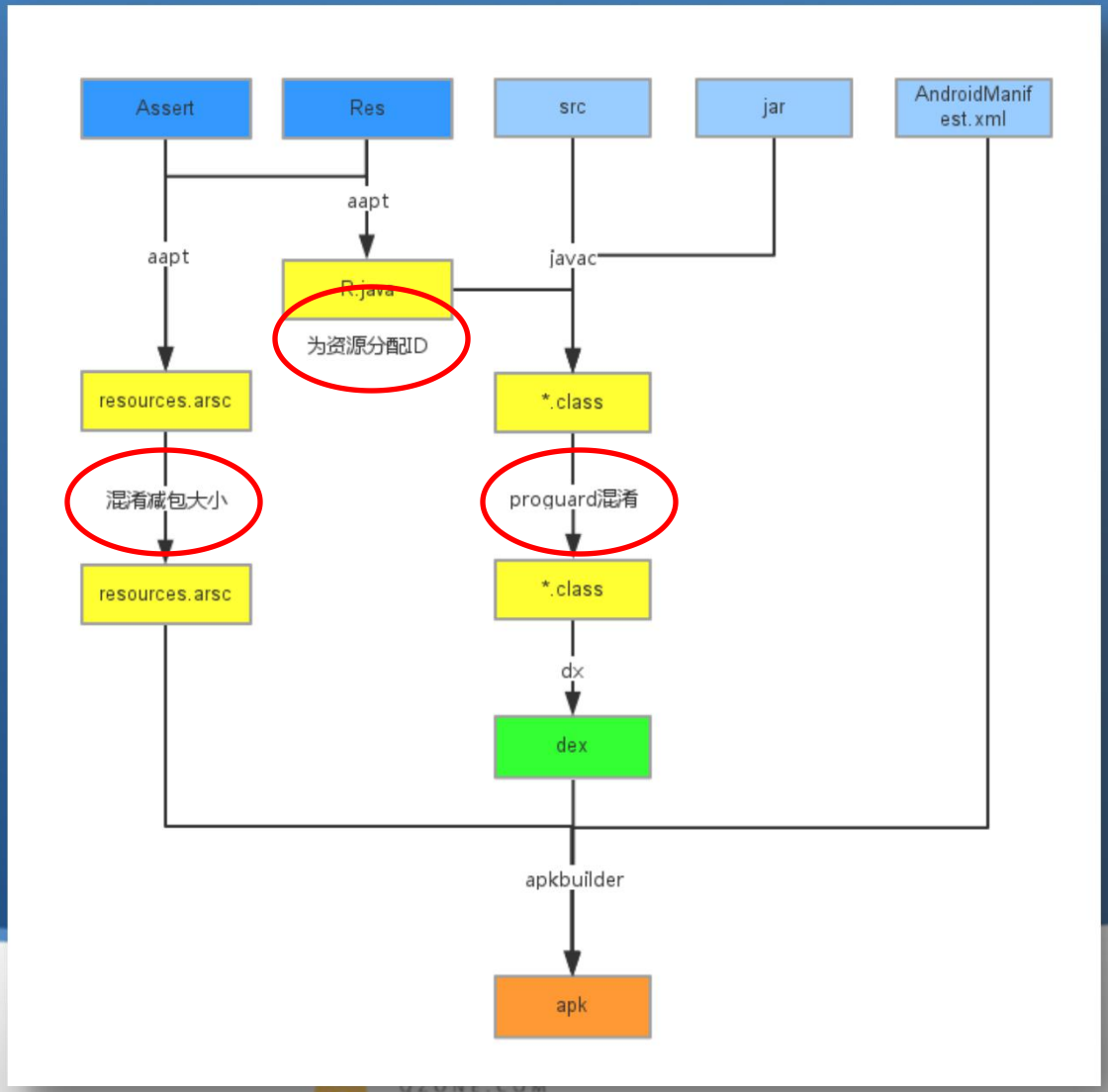


- 资源
  - Assets
  - Res
  - Resource.arsc
- 代码
  - Dex
  - So
- 描述文件
  - AndroidManifest

目录

- 一、技术背景
- 二、技术原理
- 三、技术难点
  - 生成补丁包难点（混淆，ID变化，无编译）
  - 加载补丁包难点
- 四、其它问题

从Apk打包过程说起


Assert
Res
src
jar
AndroidManifest.xml
aapt
aapt
javac
R.java
为资源分配ID
resources.arsc
*.class
混淆减包大小
proguard混淆
resources.arsc
*.class
dx
dex
apkbuilder
apk

R.Java & Resource.arsc

- R.java

```
public final class R {
    public static final class anim {...}
    public static final class array {...}
    public static final class attr {...}
    public static final class color {...}
    public static final class dimen {...}
    public static final class drawable {
        public static final int activity_widget_to_left_arrow=0x7f030001;
        public static final int activity_widget_to_right_arrow=0x7f030002;
        public static final int aio_file_progress_layerlist=0x7f030003;
        public static final int alertdiag_inputbox=0x7f030004;
```

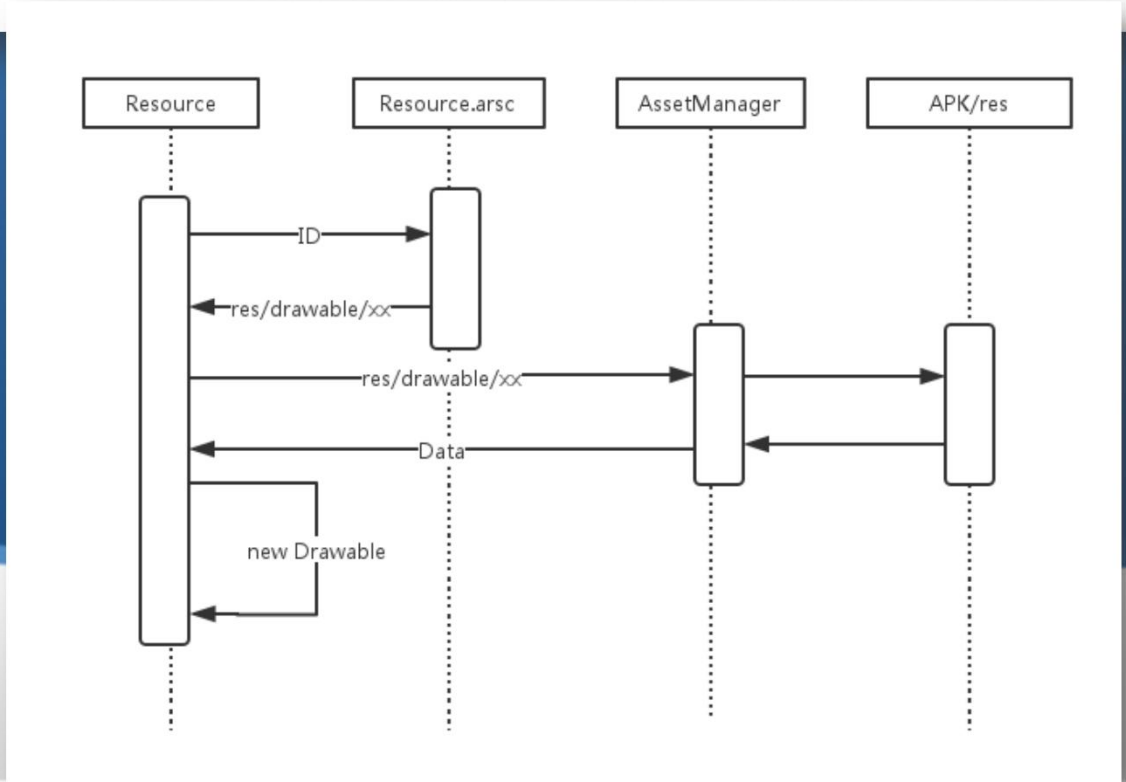
0x 7F	03	007c
PackageID	Type	ID

- Resource.arsc

```
0x7F 03 0001 ------ "/res/drawable/activity_widget_to_left_arrow"
0x7F 03 0002 ------ "/res/drawable/activity_widget_to_right_arrow"
0x7F 03 0003 ------ "/res/drawable/aio_file_progress_layerlist"
0x7F 03 0004 ------ "/res/drawable/alertdiag_inputbox"
```

加载资源的原理

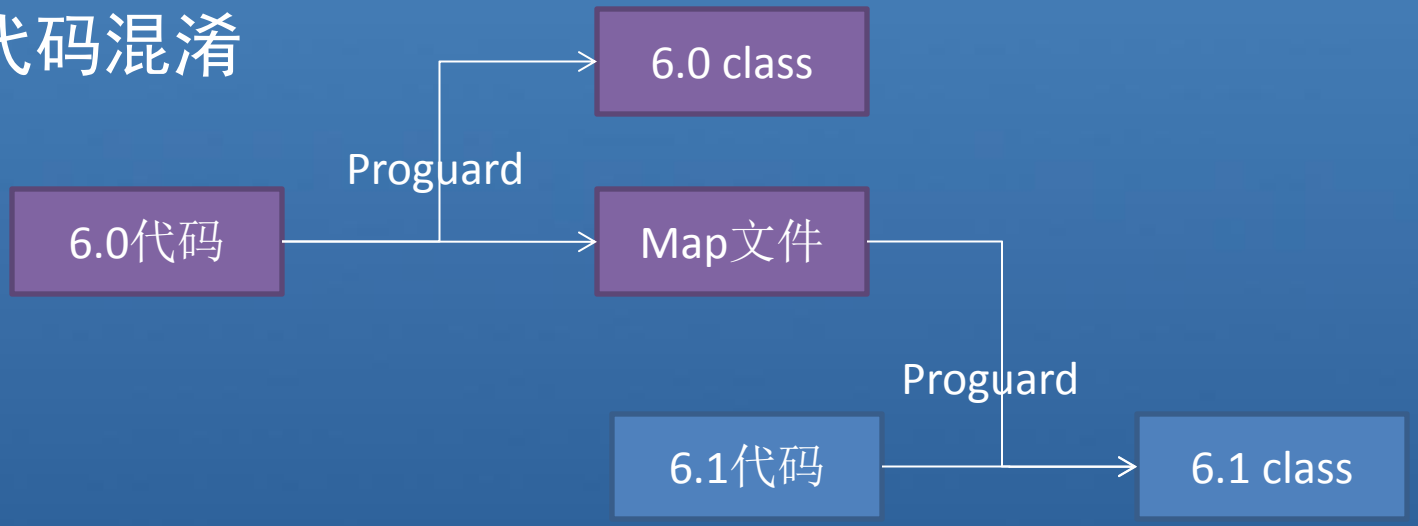
```
public class MyActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        setContentView(com.qzone.R.layout.qz_widget_tag_view);
        final Button button = (Button) findViewById(R.id.content);

        Drawable drawable = getResources().getDrawable(R.drawable.qz_bubble_popup_arrow_up_pressed);
        button.setBackgroundDrawable(drawable);
    }
}
```



难题1：代码和资源会被混淆

解决方案

- 代码混淆



  - java -jar proguard.jar -obfuscationdictionary map
  - 具体参考（http://proguard.sourceforge.net/manual/usage.html）
- 资源混淆
  - 非官方操作，自己解决吧

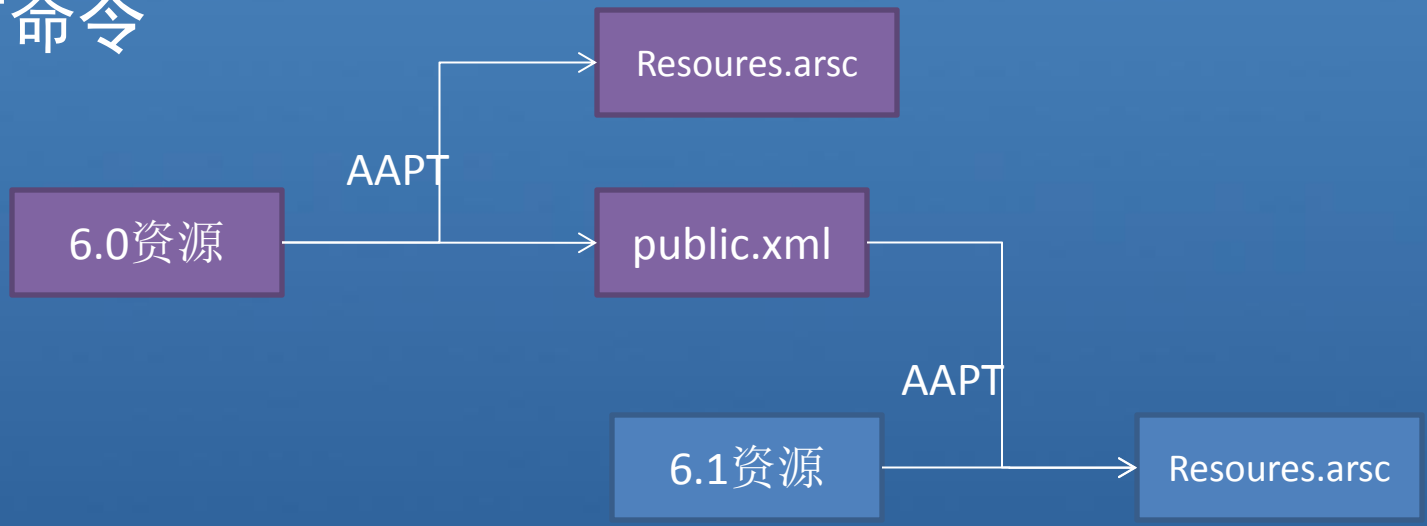
难题2：资源ID分配会有变化

```
public class MyActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        setContentView(com.qzone.R.layout.qz_widget_tag_view);
        final Button button = (Button) findViewById(R.id.content);

        Drawable drawable = getResources().getDrawable(R.drawable.qz_bubble_popup_arrow_up_pressed);
        button.setBackgroundDrawable(drawable);
    }
}
```

0x7F 03 0001 ----→0x7F 03 0002

解决方案

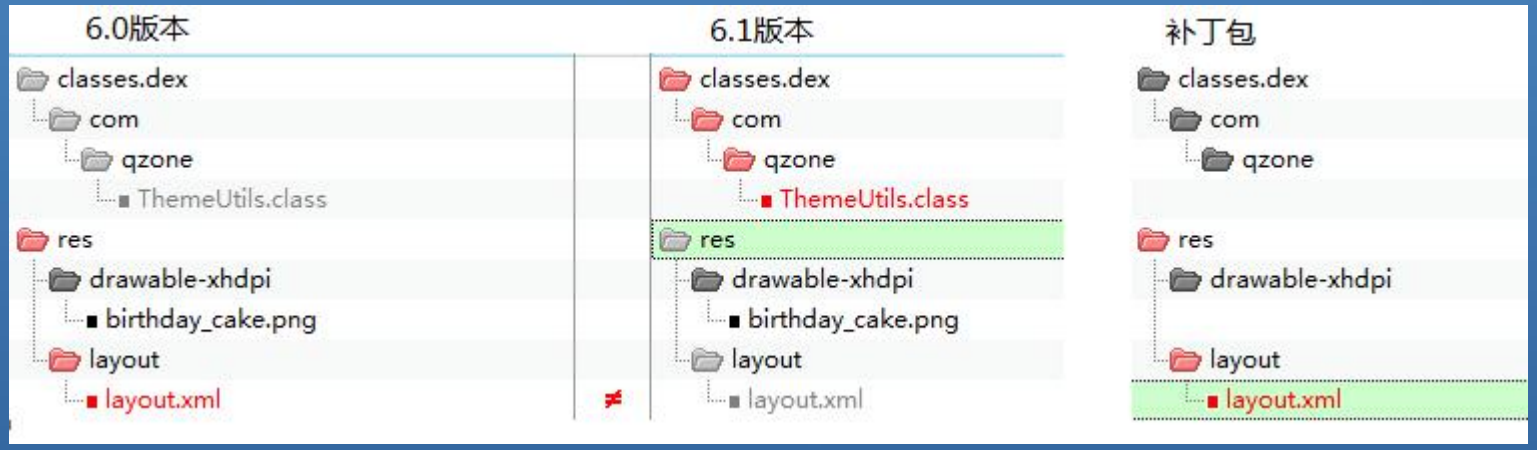
- AAPT命令

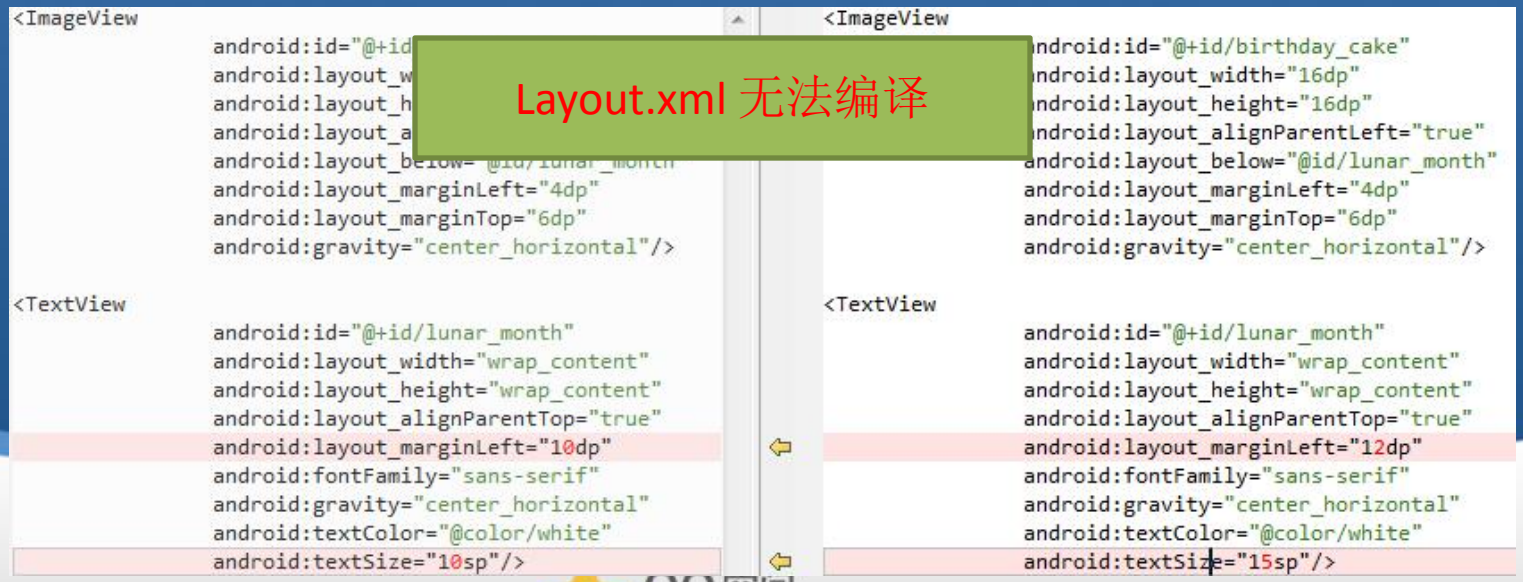


- res/ values/目录下添加public.xml文件

```
<public type="id" name="title_bar" id="0x7f020000" />
<public type="id" name="bar_title" id="0x7f020001" />
<public type="id" name="bar_back_button" id="0x7f020002" />
<public type="id" name="bar_left_button" id="0x7f020003" />
<public type="id" name="bar_right_button" id="0x7f020004" />
<public type="id" name="bar_center_left_button" id="0x7f020005" />
```

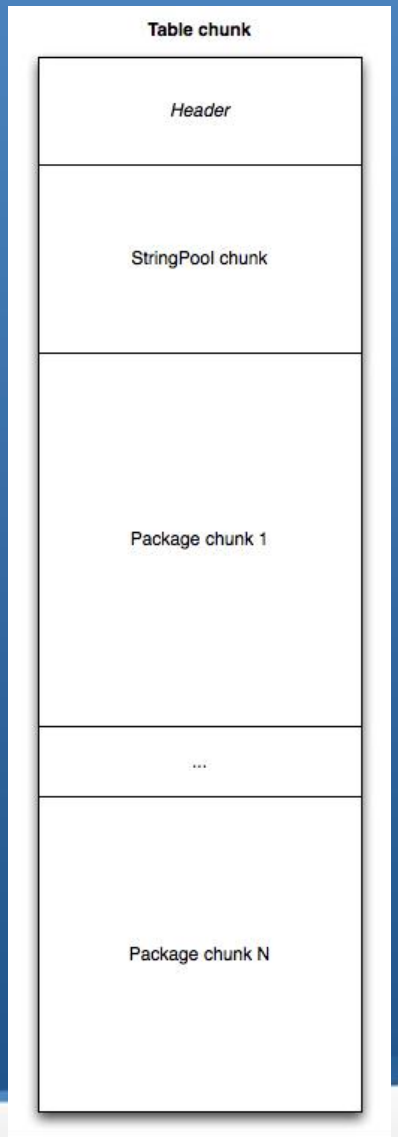
难题3：Resources.arsc无法编译

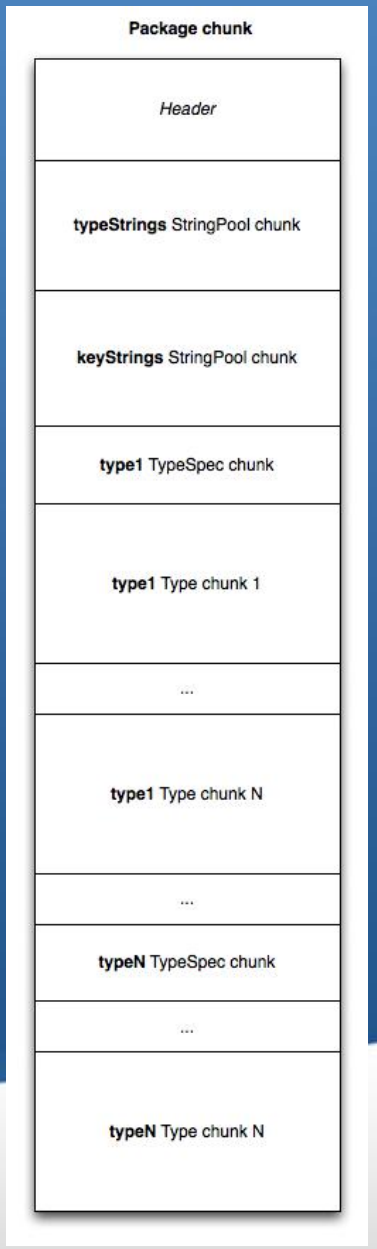




解决方案

- 不用AAPT 自己生成Resources.arsc
- 参考：https://justanapplication.wordpress.com/category/android/android-resources/

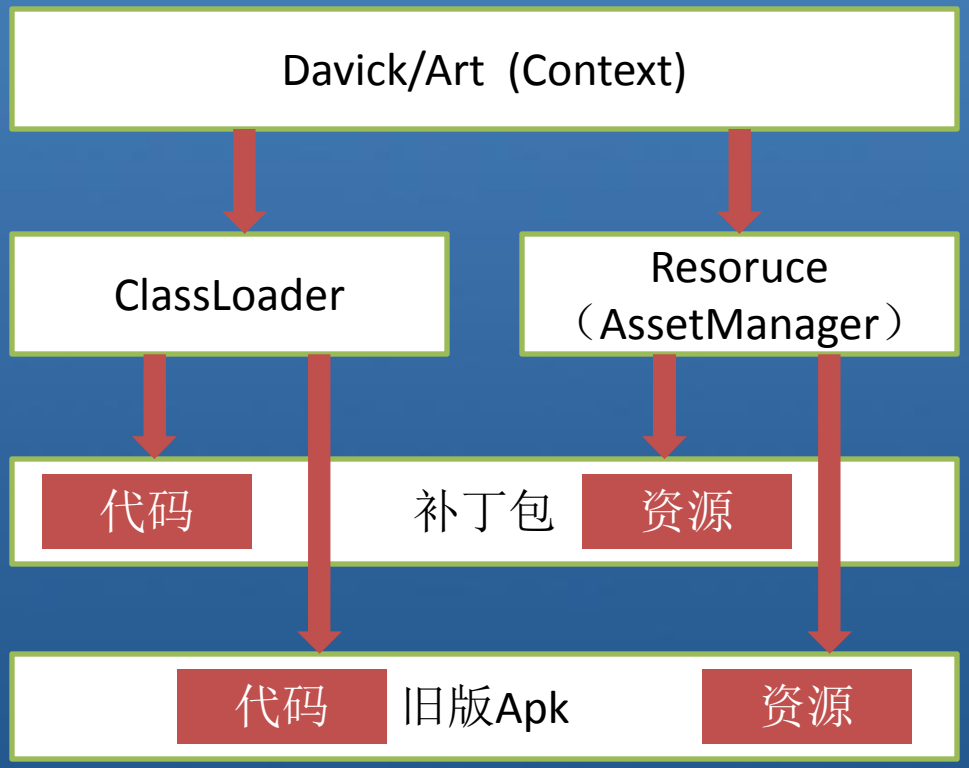




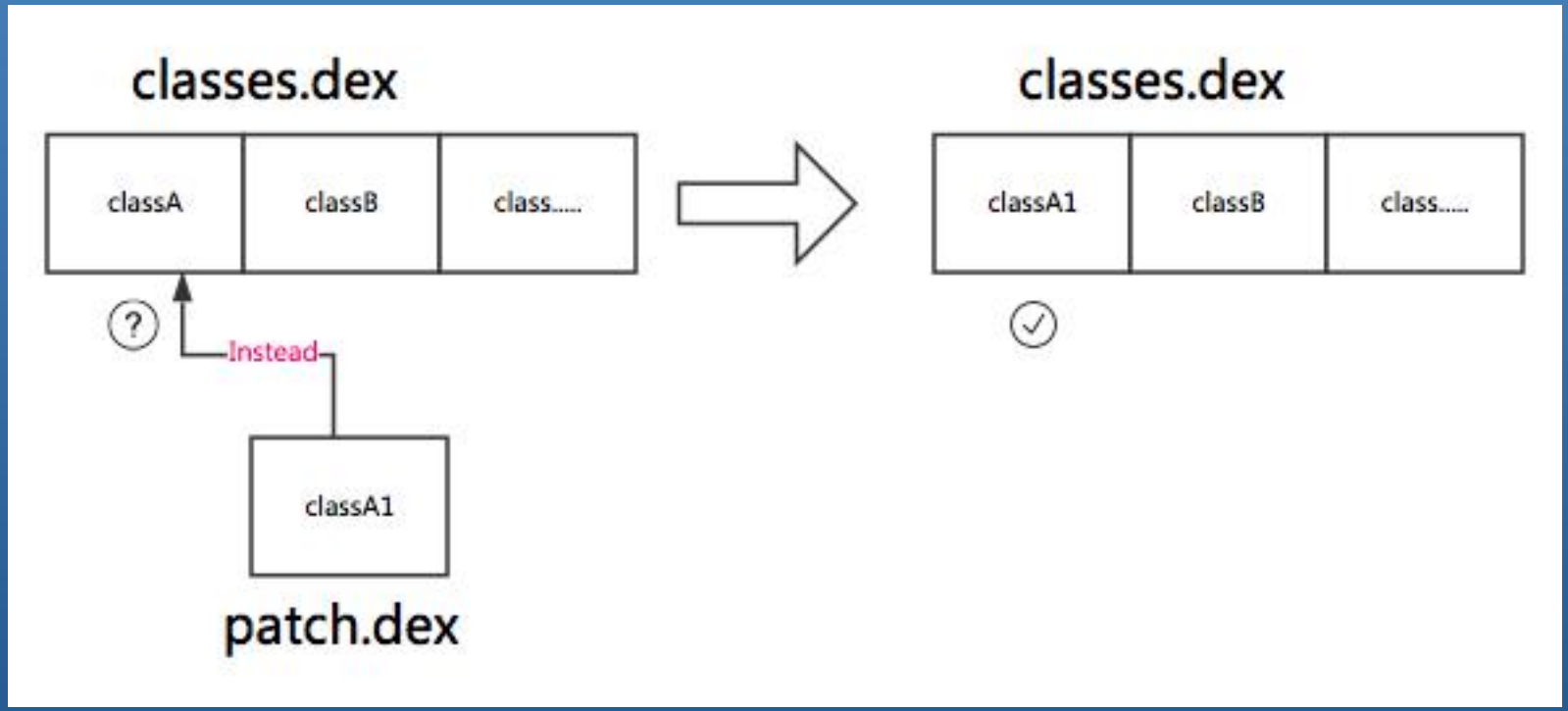
目录

- 一、技术背景
- 二、技术原理
- 三、技术难点
  - 生成补丁包难点
  - 加载补丁包难点（ISPREVARIFIED错误，无法增加Activity）
- 四、其它问题

加载补丁



加载补丁包



就这么简单？

困难1：CLASS_ISPREVERIFIED错误

IllegalAccessError:
Class ref in pre-verified class resolved to unexpected implementation

```
I  04-27 11:15:14.574  23277  23277  STARTUP         attachBaseContext end >>>>> 0ms ,pid:23277
W  04-27 11:15:14.574  23277  23277  dalvikvm        Class resolved by unexpected DEX: Lcom/qzone/module/ModuleManager;(0x41ae3bd0 □
                                                    ):0x75fec000 ref [Lcom/qzonex/app/QZoneActivityManager;] Lcom/qzonex/app/QZon □
                                                    eActivityManager;(0x41ae3bd0):0x715ed000
W  04-27 11:15:14.574  23277  23277  dalvikvm        (Lcom/qzone/module/ModuleManager; had used a different Lcom/qzonex/app/QZoneA □
                                                    ctivityManager; during pre-verification)
D  04-27 11:15:14.574  23277  23277  AndroidRuntime  Shutting down VM
W  04-27 11:15:14.574  23277  23277  dalvikvm        threadid=1: thread exiting with uncaught exception (group=0x41638e40)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  FATAL EXCEPTION: main
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  Process: com.qzone, PID: 23277
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  java.lang.IllegalAccessError: Class ref in pre-verified class resolved to une □
                                                    xpected implementation
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at com.qzone.module.ModuleManager.init(ModuleManager.java:61)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at com.qzonex.app.QZoneRealApplication.attachBaseContext(QZoneRealApplicatio □
                                                    n.java:125)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at android.app.Application.attach(Application.java:181)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at android.app.Instrumentation.newApplication(Instrumentation.java:991)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at android.app.Instrumentation.newApplication(Instrumentation.java:975)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at android.app.LoadedApk.makeApplication(LoadedApk.java:509)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at android.app.ActivityThread.handleBindApplication(ActivityThread.java:4446 □
                                                    )
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at android.app.ActivityThread.access$1500(ActivityThread.java:144)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1265)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at android.os.Handler.dispatchMessage(Handler.java:102)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at android.os.Looper.loop(Looper.java:136)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at android.app.ActivityThread.main(ActivityThread.java:5146)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at java.lang.reflect.Method.invokeNative(Native Method)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at java.lang.reflect.Method.invoke(Method.java:515)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.jav □
                                                    a:732)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:566)
E  04-27 11:15:14.574  23277  23277  AndroidRuntime  at dalvik.system.NativeStart.main(Native Method)
```

源代码

```
    */
    if (!fromUnverifiedConstant &&
        IS_CLASS_FLAG_SET(referrer, CLASS_ISPREVERIFIED))
    {
        ClassObject* resClassCheck = resClass;
        if (dvmIsArrayClass(resClassCheck))
            resClassCheck = resClassCheck->elementClass;

        if (referrer->pDvmDex != resClassCheck->pDvmDex &&
            resClassCheck->classLoader != NULL)
        {
            ALOGW("Class resolved by unexpected DEX:"
                  " %s(%p):%p ref [%s] %s(%p):%p",
                referrer->descriptor, referrer->classLoader,
                referrer->pDvmDex,
                resClass->descriptor, resClassCheck->descriptor,
                resClassCheck->classLoader, resClassCheck->pDvmDex);
            ALOGW("(%s had used a different %s during pre-verification)",
                referrer->descriptor, resClass->descriptor);
            dvmThrowIllegalAccessError(
                "Class ref in pre-verified class resolved to unexpected "
                "implementation");
            return NULL;
        }
    } ?  end if ! fromUnverifiedConsta... ?
```

解决方案：

- 阻止类被打ISPREVERIFIED标记


```
public class AllClass {
    public AllClass() {
        if (NotDoVerifyClasses.DO_VERIFY_CLASSES)
            System.out.print(this.getClass());
    }
}
```

```
public class NotDoVerifyClasses {
    public static boolean DO_VERIFY_CLASSES = false;
}
```

- DexOpt时，如果调用的另一个类不在同一个Dex中，就不会打上ISPREVERIFIED标记

困难2:不能新增Activity

- 因为AndroidManifest.xml是安装时向系统注册Activity等信息，运行时无法替换。没有办法增加Activity

解决方案

- 预埋多个Activity
- 编译补丁包时，把新增Activity名字修改为ChunkActivity。
- 算是部分解决问题（不能Export）

```
<activity
        android:name="chunk.ChunkActivity1"
        android:configChanges="keyboardHidden|orientation|screenSize|keyboard"/>
<activity
        android:name="chunk.ChunkActivity2"
        android:configChanges="keyboardHidden|orientation|screenSize|keyboard"/>
<activity
        android:name="chunk.ChunkActivity3"
        android:configChanges="keyboardHidden|orientation|screenSize|keyboard"/>
<activity
        android:name="chunk.ChunkActivity4"
        android:configChanges="keyboardHidden|orientation|screenSize|keyboard"/>
```

目录

- 一、技术背景
- 二、技术原理
- 三、技术难点
- 四、其它问题
  - 安全性
  - 补丁包工具

四、其它问题：安全性

- 补丁包无法修改AndroidManifest，不会增加Android权限
- 签名校验
- 支持回滚

四、其它问题：专用补丁包工具

- 保证代码和资源混淆不变化
- 自动给代码插桩阻止系统进行Dexopt等
- 针对新增Activity改名，检测Export的Activity报错等

Tencent 腾讯

谢谢大家

QQ空间
QZONE.COM