Android Fuzz测试工具
——Monkey源代码分析

chenjie@xxx.com

2013.02

大纲

- **总体设计架构**
- **问题：**
  - 如何运行使用？
  - 如何创建和消费事件？
  - 如何向Activity注入事件？
  - 如何做监控？
- **扩展：IWindowManager事件注入背后！**
- **总结**
  - 设计亮点
  - 改进思路

问题1：如何运行使用？

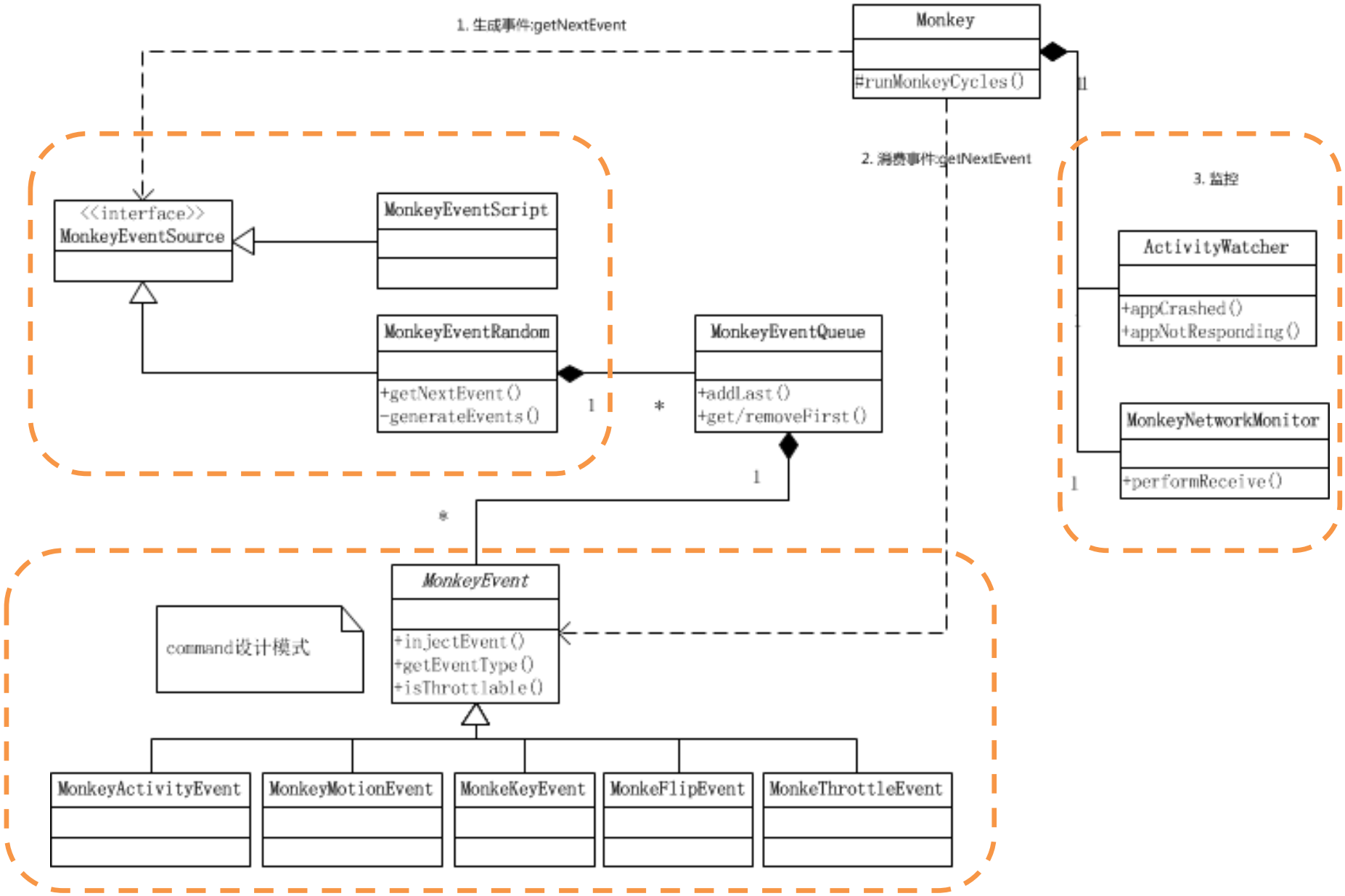
```
C:\Users\cat>adb shell monkey
usage: monkey [-p ALLOWED_PACKAGE [-p ALLOWED_PACKAGE] ...]
              [-c MAIN_CATEGORY [-c MAIN_CATEGORY] ...]
              [--ignore-crashes] [--ignore-timeouts]
              [--ignore-security-exceptions] [--monitor-native-crashes]
              [--kill-process-after-error] [--hprof]
              [--pct-touch PERCENT] [--pct-motion PERCENT]
              [--pct-trackball PERCENT] [--pct-syskeys PERCENT]
              [--pct-nav PERCENT] [--pct-majornav PERCENT]
              [--pct-appswitch PERCENT] [--pct-flip PERCENT]
              [--pct-anyevent PERCENT]
              [--wait-dbg] [--dbg-no-events]
              [--setup scriptfile] [-f scriptfile [-f scriptfile] ...]
              [--port port]
              [-s SEED] [-v [-v] ...] [--throttle MILLISEC]
              COUNT
```

```
C:\Users\cat>adb shell monkey -p cn.com.android123.cwj -v 100
:Monkey: seed=0 count=100
:AllowPackage: cn.com.android123.cwj
:IncludeCategory: android.intent.category.LAUNCHER
:IncludeCategory: android.intent.category.MONKEY
```

启动App，用默认事件比例，并向其发送100个伪随机事件
（可以自行设置各种事件的比例）

事件	-s <seed>	伪随机数生成器的seed值。如果用相同的seed值再次运行Monkey，它将生成相同的事件序列。
	--throttle <milliseconds>	在事件之间插入固定延迟。通过这个选项可以减缓Monkey的执行速度。如果不指定该选项，Monkey将不会被延迟，事件将尽可能快地被产成。
	--pct-touch <percent>	调整触摸事件的百分比(触摸事件是一个down-up事件，它发生在屏幕上的某单一位置)。
	--pct-motion <percent>	调整动作事件的百分比(动作事件由屏幕上某处的一个down事件、一系列的伪随机事件和一个up事件组成)。
	--pct-trackball <percent>	调整轨迹事件的百分比(轨迹事件由一个或几个随机的移动组成，有时还伴随有点击)。
	--pct-nav <percent>	调整“基本”导航事件的百分比(导航事件由来自方向输入设备的up/down/left/right组成)。
	--pct-majornav <percent>	调整“主要”导航事件的百分比(这些导航事件通常引发图形界面中的动作，如：5-way键盘的中间按键、回退按键、菜单按键)
	--pct-syskeys <percent>	调整“系统”按键事件的百分比(这些按键通常被保留，由系统使用，如Home、Back、Start Call、End Call及音量控制键)。
	--pct-appswitch <percent>	调整启动Activity的百分比。在随机间隔里，Monkey将执行一个startActivity()调用，作为最大程度覆盖包中全部Activity的一种方法。
	--pct-anyevent <percent>	调整其它类型事件的百分比。它包罗了所有其它类型的事件，如：按键、其它不常用的设备按钮、等等。
约束限制	-p <allowed-package-name>	如果用此参数指定了一个或几个包，Monkey将只允许系统启动这些包里的Activity。如果你的应用程序还需要访问其它包里的Activity(如选择取一个联系人)，那些包也需要在此同时指定。如果不指定任何包，Monkey将允许系统启动全部包里的Activity。要指定多个包，需要使用多个-p选项，每个-p选项只能用于一个包。
	-c <main-category>	如果用此参数指定了一个或几个类别，Monkey将只允许系统启动被这些类别中的某个类别列出的Activity。如果不指定任何类别，Monkey将选择下列类别中列出的Activity：Intent.CATEGORY_LAUNCHER或Intent.CATEGORY_MONKEY。要指定多个类别，需要使用多个-c选项，每个-c选项只能用于一个类别。

总体设计


1. 生成事件:getNextEvent
Monkey
#runMonkeyCycles()
1
2. 消费事件:getNextEvent
3. 监控
<<interface>>
MonkeyEventSource
MonkeyEventScript
MonkeyEventRandom
+getNextEvent()
-generateEvents()
1
*
MonkeyEventQueue
+addLast()
+get/removeFirst()
1
*
ActivityWatcher
+appCrashed()
+appNotResponding()
MonkeyNetworkMonitor
+performReceive()
1
MonkeyEvent
+injectEvent()
+getEventType()
+isThrottlable()
command设计模式
MonkeyActivityEvent
MonkeyMotionEvent
MonkeKeyEvent
MonkeFlipEvent
MonkeThrottleEvent

问题2：如何创建和消费事件？

2.1：主循环Monkey.runMonkeyCycles

```
while (!systemCrashed && i < mCount) {
    . . .
    //生成事件
    MonkeyEvent ev = mEventSource.getNextEvent();
    if ( ev != null) {
        // We don't want to count throttling as an event.
        // MonkeyThrottleEvent事件是在MonkeyEventQueue.addLast中自动添加的
        if (!( ev instanceof MonkeyThrottleEvent)) {
            i++;
        }
        //消费事件
        int injectCode = ev.injectEvent(mWm, mAm, mVerbose);
        if (injectCode == MonkeyEvent. INJECT_FAIL) {
            if ( ev instanceof MonkeyKeyEvent) {
                mDroppedKeyEvents++;
            } else if ( ev instanceof MonkeyMotionEvent) {
                mDroppedPointerEvents++;
            } else if ( ev instanceof MonkeyFlipEvent) {
                mDroppedFlipEvents++;
            }
        } else if (injectCode == MonkeyEvent.INJECT_ERROR_REMOTE_EXCEPTION ) {
            systemCrashed = true;
        } else if (injectCode == MonkeyEvent.INJECT_ERROR_SECURITY_EXCEPTION ) {
            systemCrashed = ! mIgnoreSecurityExceptions;
        }
    }
}
```

2.2 MonkeySourceRandom如何按比例生成事件?

- **默认比例**

```
mFactors[FACTOR_TOUCH] = 15.0f;
mFactors[FACTOR_MOTION] = 10.0f;
mFactors[FACTOR_TRACKBALL] = 15.0f;
mFactors[FACTOR_NAV] = 25.0f;
mFactors[FACTOR_MAJORNAV] = 15.0f;
mFactors[FACTOR_SYSOPS] = 2.0f;
mFactors[FACTOR_APPSWITCH] = 2.0f;
mFactors[FACTOR_FLIP] = 1.0f;
mFactors[FACTOR_ANYTHING] = 15.0f;
```

- **generateEvents前据入参调整，规约累加**

```
private boolean adjustEventFactors () {
```

最后为了方便生成随机时间，对比例做了累加，并规范化到0~1之间

```
    // finally, normalize and convert to running sum
    float sum = 0.0f;
    for (int i = 0; i < FACTORZ_COUNT; ++i) {
        sum += mFactors[i] / 100.0f;
        mFactors[i] = sum;
```

2.3 事件序列

- **第一个入队的Event必然是ActivityEvent**
- **Event序列：Key，Touch，Motion**
  - ACTION_DOWN
  - ACTION_MOVE * N
  - ACTION_UP
- **延时事件**

2.4 几种事件的生成方式

- **Monkey事件体系**

2.4.1 MonkeyMotionEvent

- 含FACTOR_TOUCH, FACTOR_MOTION
- 是一个基于屏幕绝对位置（x, y）的事件序列：ACTION_DOWN/MOVE/UP，MOVE事件序列带有随机性

```
if (motionEvent) {
    int count = random.nextInt(10);
    for ( int i = 0; i < count; i++) {
        // generate some slop in the up event
        x = (x + (random.nextInt() % 10)) % display.getWidth();
        y = (y + (random.nextInt() % 10)) % display.getHeight() ;

        e = new MonkeyMotionEvent(MonkeyEvent.EVENT_TYPE_POINTER ,
                downAt, MotionEvent. ACTION_MOVE, x, y, 0);
```

2.4.2 MonkeyTrackballEvent

- **原理：**
  - 1、先产生10个随机MOVE事件
  - 2、再有一定几率产生一个click事件

```
private void generateTrackballEvent(Random random) {
    Display display = WindowManagerImpl.getDefault().getDefaultDisplay();
    boolean drop = false;
    int count = random.nextInt(10);
    MonkeyMotionEvent e;
  ??小bug，count本应作为循环条件的
    for (int i = 0; i < 10; ++i) {

  ??问题：click事件不需要指定坐标吗？
    e = new MonkeyMotionEvent(MonkeyEvent.EVENT_TYPE_TRACKBALL , downAt,
            MotionEvent.ACTION_DOWN, 0, 0, 0);
    e.setIntermediateNote( true);
    mQ.addLast(e);


    e = new MonkeyMotionEvent(MonkeyEvent.EVENT_TYPE_TRACKBALL , downAt,
            MotionEvent.ACTION_UP, 0, 0, 0);
    e.setIntermediateNote( false);
    mQ.addLast(e);
```

2.4.3 MonkeyKeyEvent

- **原理：**
  - 种类：FACTOR_NAV,FACTOR_MAJORNAV, FACTOR_SYSOPS,NORMAL_KEYS
  - 包含ACTION_DOWN和ACTION_UP事件对

```
MonkeyKeyEvent e =new MonkeyKeyEvent(KeyEvent.ACTION_DOWN , lastKey);
mQ.addLast(e);

e =new MonkeyKeyEvent(KeyEvent. ACTION_UP, lastKey);
mQ.addLast(e);
```

2.4.4 MonkeyActivityEvent

- 1. Monkey.getMainApps()中初始化了可以切换的App列表：基于Component

```
for ( int a = 0; a < NA; a++) {
     ResolveInfo r = mainApps.get(a);
    if ( mValidPackages.size() == 0 ||
            mValidPackages.contains(r.activityInfo .applicationInfo .packageName )) {
         }
        mMainApps.add(new ComponentName(
                r.activityInfo.applicationInfo .packageName ,
                r.activityInfo. name));
```

- 2. 只会触发一个Application的入口Activity

```
private Intent getEvent() {
  Intent intent = new Intent(Intent. ACTION_MAIN);
  intent.addCategory(Intent. CATEGORY_LAUNCHER);
  intent.setComponent( mApp);
  intent.addFlags(Intent. FLAG_ACTIVITY_NEW_TASK);
  return intent;
 }
```

2.4.5 MonkeyFlipEvent

- 1. 模拟手机设备翻转，每次取反

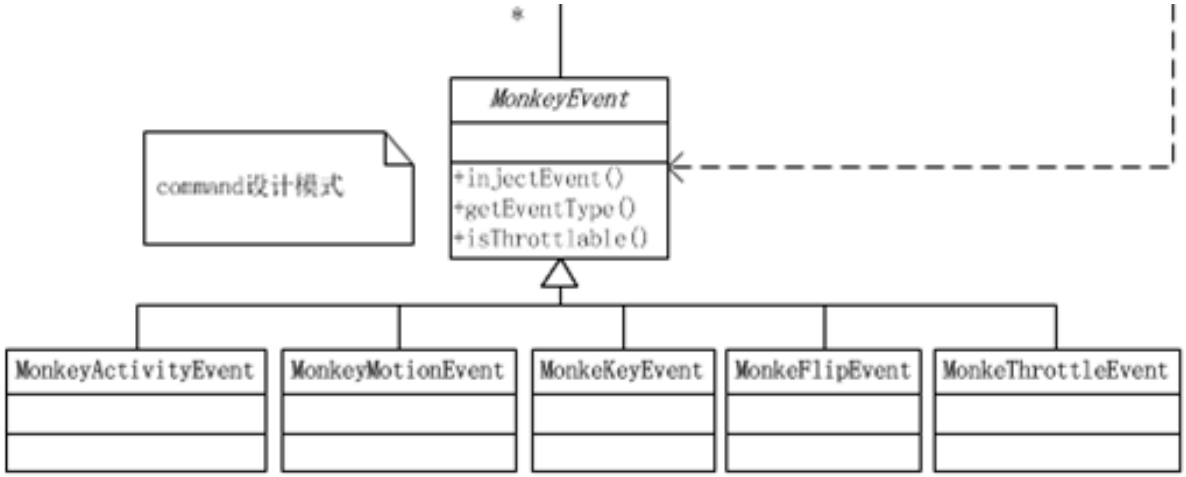
```
else if (cls < mFactors[FACTOR_FLIP]) {
 MonkeyFlipEvent e = new MonkeyFlipEvent(mKeyboardOpen);
mKeyboardOpen = !mKeyboardOpen;
mQ.addLast(e);
return;
```

- 2. 直接写设备文件来模拟

```
// inject flip event
try {
    FileOutputStream f = new FileOutputStream("/dev/input/event0");
    f.write(mKeyboardOpen ? FLIP_0 : FLIP_1);
    f.close();
   return MonkeyEvent.INJECT_SUCCESS;
```

问题3. 如何向Activity注入事件？

- **Command设计模式**




```
/**
 * a method for injecting event
 * @param iwm wires to current window manager
 * @param iam wires to current activity manager
 * @param verbose a log switch
 * @return INJECT_SUCCESS if it goes through, and INJECT_FAIL if it fails
 *         in the case of exceptions, return its corresponding error code
 */
public abstract int injectEvent (IWindowManager iwm, IActivityManager iam, int verbose);
```

3.1 依赖WindowManager注入

- **MotionEvent**
- **KeyEvent**

http://www.oschina.net/code/explore/android-4.0.1/core/java/android/view/IWindowManager.aidl

```
068     // These can only be called when injecting events to your own window,
069     // or by holding the INJECT_EVENTS permission.  These methods may block
070     // until pending input events are finished being dispatched even when 'sync' is false.
071     // Avoid calling these methods on your UI thread or use the 'NoWait' version instead.
072     boolean injectKeyEvent(in KeyEvent ev, boolean sync);
073     boolean injectPointerEvent(in MotionEvent ev, boolean sync);
074     boolean injectTrackballEvent(in MotionEvent ev, boolean sync);
075     boolean injectInputEventNoWait(in InputEvent ev);
```

```
int type = this.getEventType();
MotionEvent me = getEvent();

if ((type == MonkeyEvent.EVENT_TYPE_POINTER &&
        !iwm.injectPointerEvent(me, false))
        || (type == MonkeyEvent.EVENT_TYPE_TRACKBALL &&
                !iwm.injectTrackballEvent(me, false))) {
    return MonkeyEvent.INJECT_FAIL;
}
```

3.2 其他

- FlipEvent：写设备文件
- ActivityEvent：Intent
- ThrottleEvent：只是sleep

```
public int injectEvent(IWindowManager iwm, IActivityManager iam, int verbose) {
  try {
       Thread.sleep(mThrottle);
   } catch (InterruptedException e1) {
       System.out.println("** Monkey interrupted in sleep.");
      return MonkeyEvent.INJECT_FAIL;
   }

  return MonkeyEvent.INJECT_SUCCESS;
}
```

问题4、如何做监控： ActivityWatcher

- **ActivityWatcher**
  - extends IActivityWatcher
  - 主要监控Activity的crash和ANR事件

```
public interface IActivityWatcher
extends IInterface
```

Testing interface to monitor what is happening in the activity manager while tests are running.
Not for normal application development.

Method Summary	
boolean	activityResuming(String pkg) The system is trying to return to an activity.
boolean	activityStarting(Intent intent, String pkg) The system is trying to start an activity.
boolean	appCrashed(String processName, int pid, String shortMsg, String longMsg, An application process has crashed (in Java).
int	appNotResponding(String processName, int pid, String processStats) An application process is not responding.

问题4、如何做监控： MonkeyNetworkMonitor

- **MonkeyNetworkMonitor**
  - extends IIntentReceiver .Stub
  - 统计运行期mobile和wifi连接时长

```
mNetworkMonitor .register(mAm);
```

```
mNetworkMonitor.start();
int crashedAtCycle = runMonkeyCycles();
mNetworkMonitor.stop();
```

```
mNetworkMonitor .unregister(mAm);
```

```
mNetworkMonitor.dump();
```

问题4、如何做监控： MonkeyNetworkMonitor

- **为什么监控网络会和Intent扯上关系？**
  - 求证：每次网络连接变化都会由系统发出广播事件！而Monkey捕获这类广播！

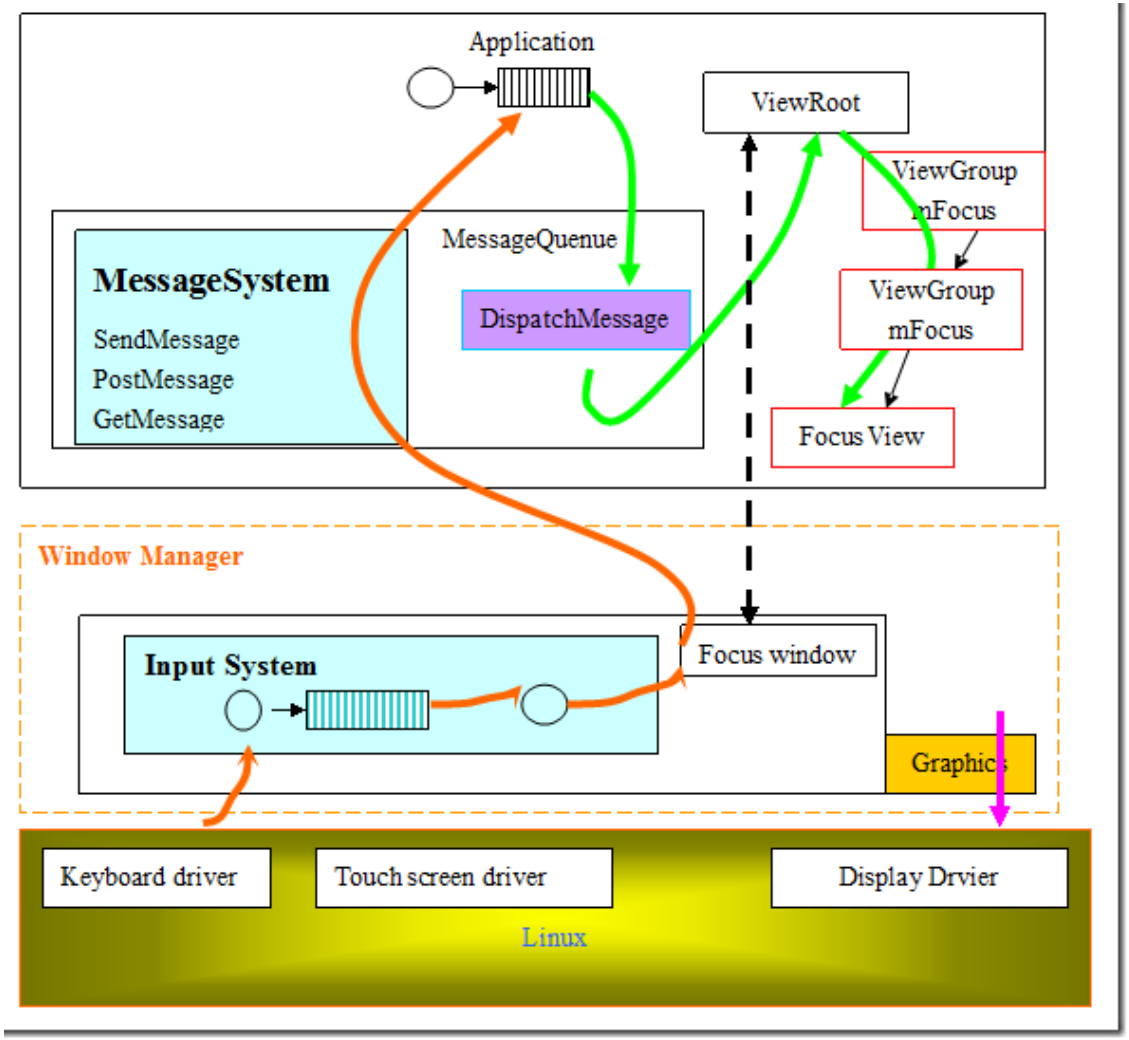
```
public void register(IActivityManager am) throws RemoteException {
    if (LDEBUG) System.out.println( "registering Receiver");
     am.registerReceiver(null, this, filter, null);
}
private final IntentFilter filter = new IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);
```

```
/**
 * Class that answers queries about the state of network connectivity. It also
 * notifies applications when network connectivity changes. Get an instance
 * of this class by calling
 * {@link android.content.Context#getSystemService(String) Context.getSystemService(Context.
 * <p>
 * The primary responsibilities of this class
 * <ol>
 * <li>Monitor network connections (Wi-Fi, GP
 * <li>Send broadcast intents when network co
 * <li>Attempt to "fail over" to another netw
 * is lost </li>
 * <li>Provide an API that allows application
 * state of the available networks </li>
 * </ol>
 */
public class ConnectivityManager
```

```
private void updateNetworkStats() {
    ......
    long delta = timeNow - mEventTime;
    switch (mLastNetworkType) {
        case ConnectivityManager.TYPE_MOBILE:
            mMobileElapsedTime += delta;
            break;
        case ConnectivityManager.TYPE_WIFI:
            mWifiElapsedTime += delta;
            break;
        default:
            ......
```

IWindowManager事件注入背后！

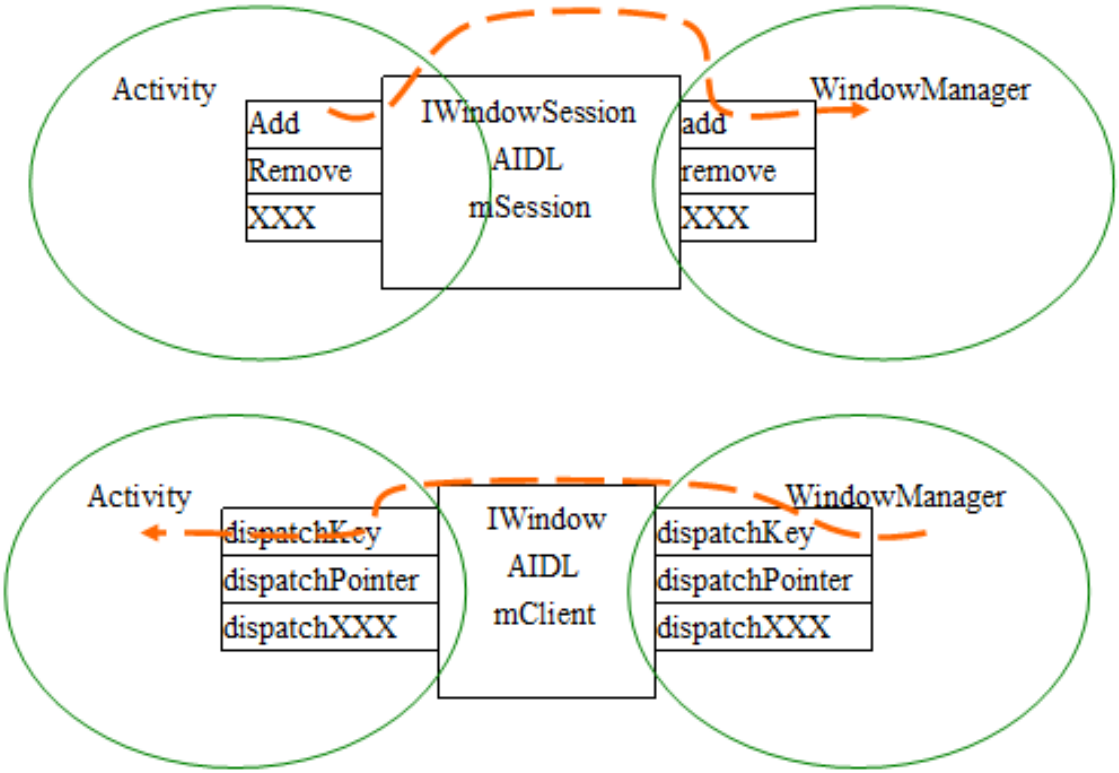
- IWindowManager和WindowManagerService的binder通信与连接关系？



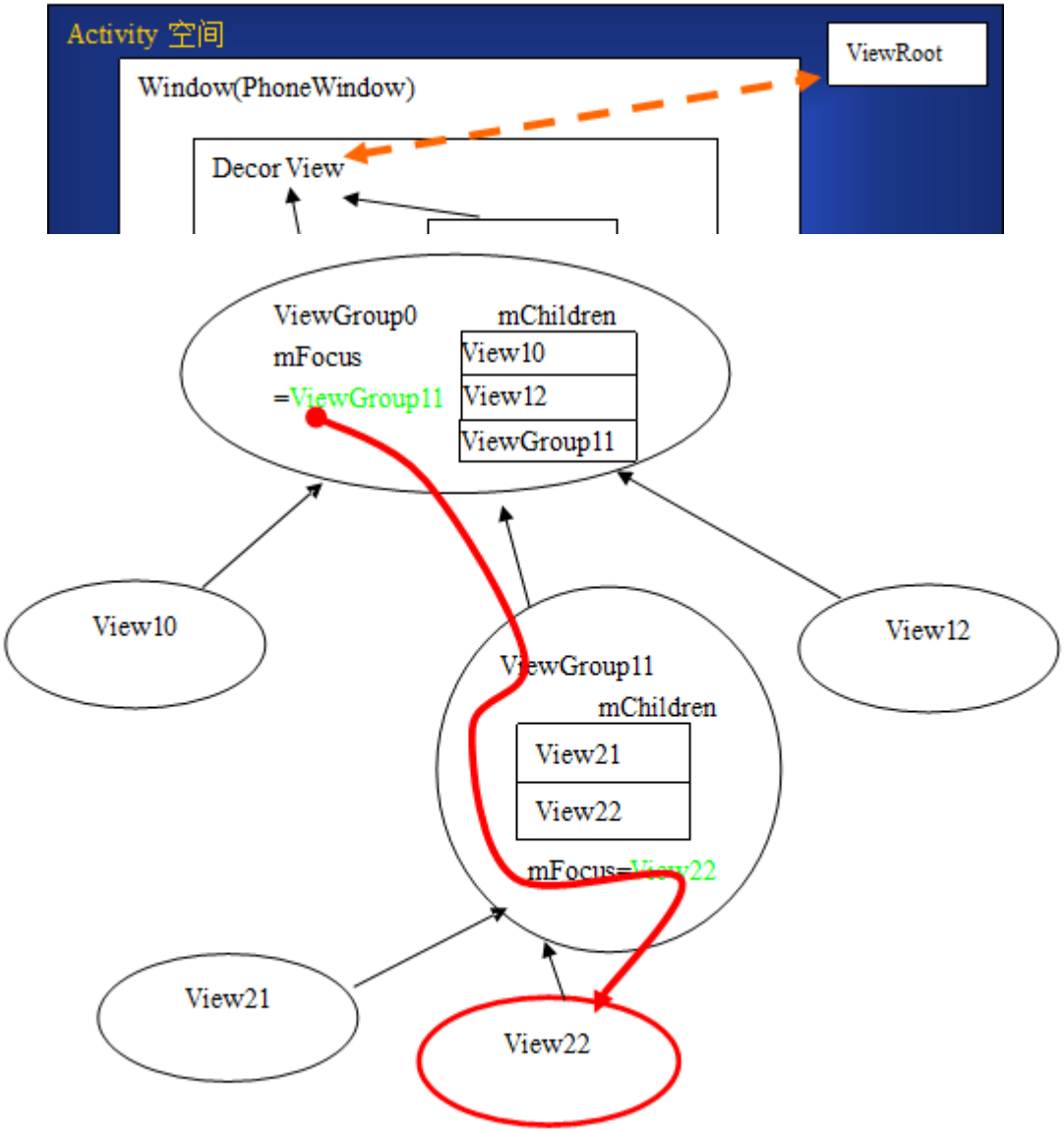
IWindowManager事件注入背后！

- injectEvent的整个过程？
  - WM.injectKeyEvent()
    - WM.dispatchKey();
      - focus.mClient.dispatchKey(event);

      //（成为焦点的）WindowState对象，有一个mClient成员，这就是IWindow！



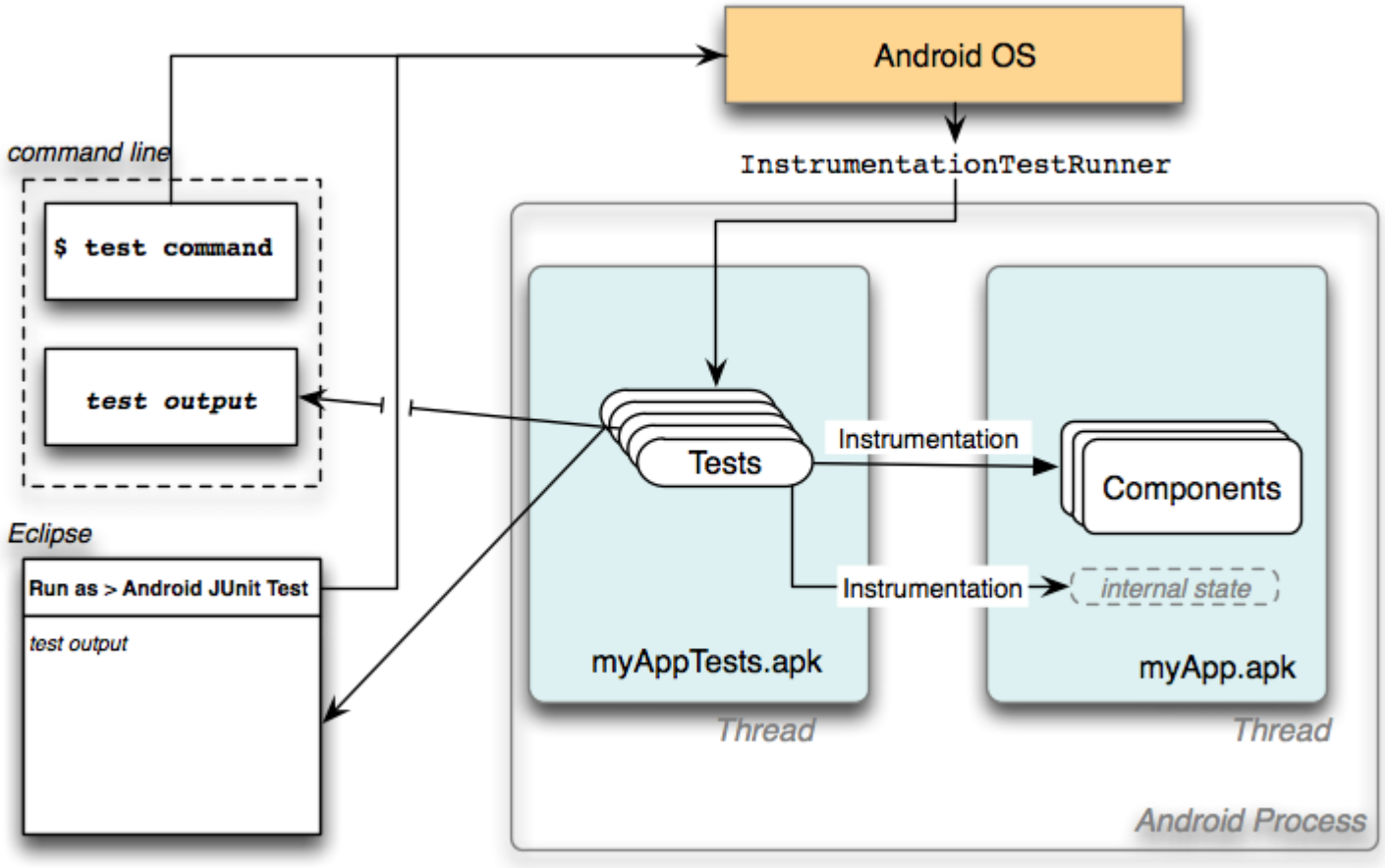
IWindowManager事件注入背后！



总结

- **设计亮点**
  - 事件的Command设计模式
    - 接口统一，可扩展，可自我运行
  - 观察者模式：Monitor体系
- **改进思路**
  - 缺点：时间根据事件比例全随机
  - 1、可能覆盖不到App中的所有Activity
  - 2、不会根据Activity特点做针对性测试
    - ListView：上下滑动，图片点击等
    - MonkeySourceScript如何使用？
  - 3、没有输入提交能力

让Instrmentation拥有Monkey的能力？



附录

- **最新的源码：**
  - https://code.google.com/p/android-source-browsing/source/browse/cmds/monkey/src/com/android/commands/monkey/?repo=platform--development&name=android-4.2_r1