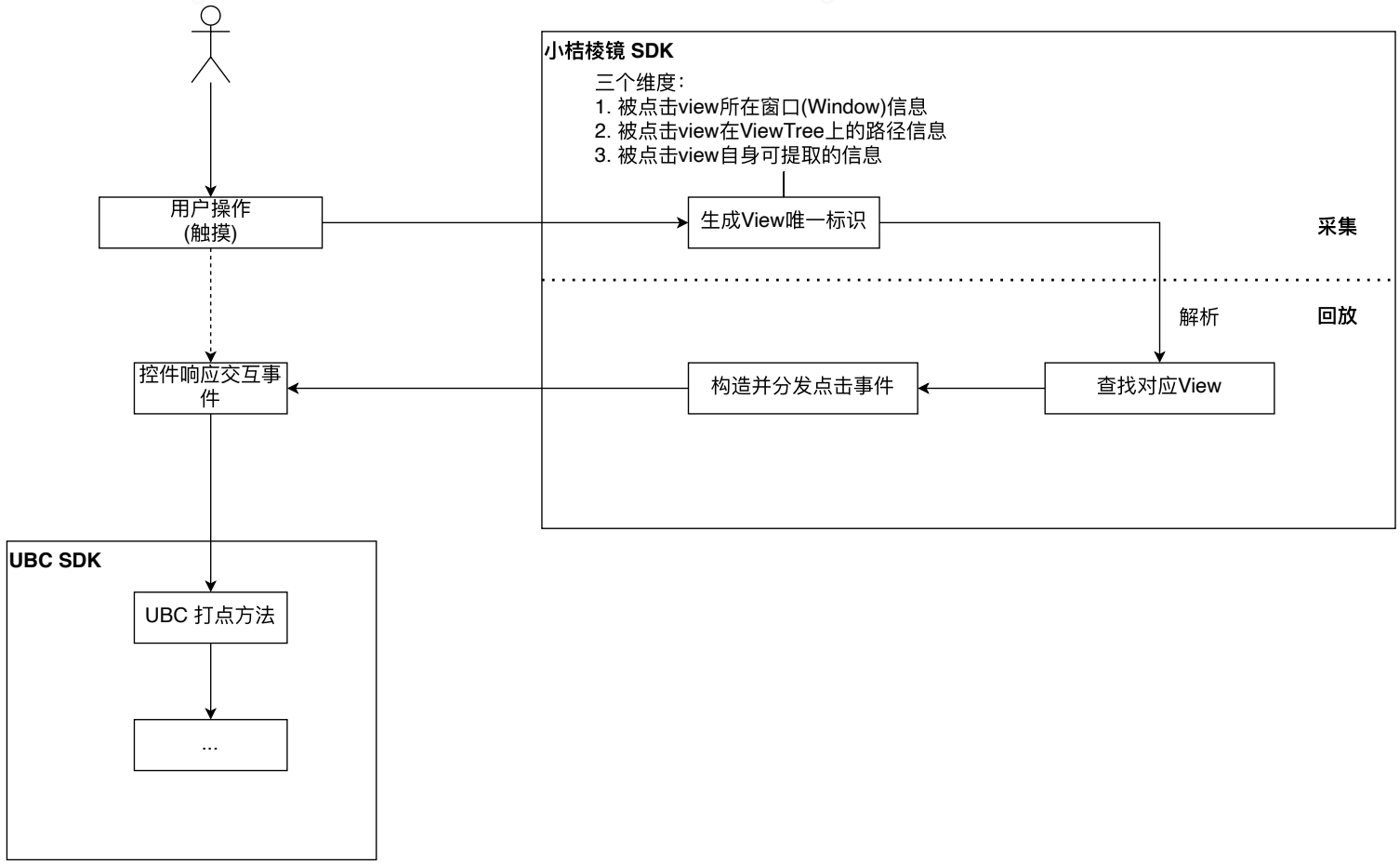
打点场景回归-小桔棱镜Android SDK调研

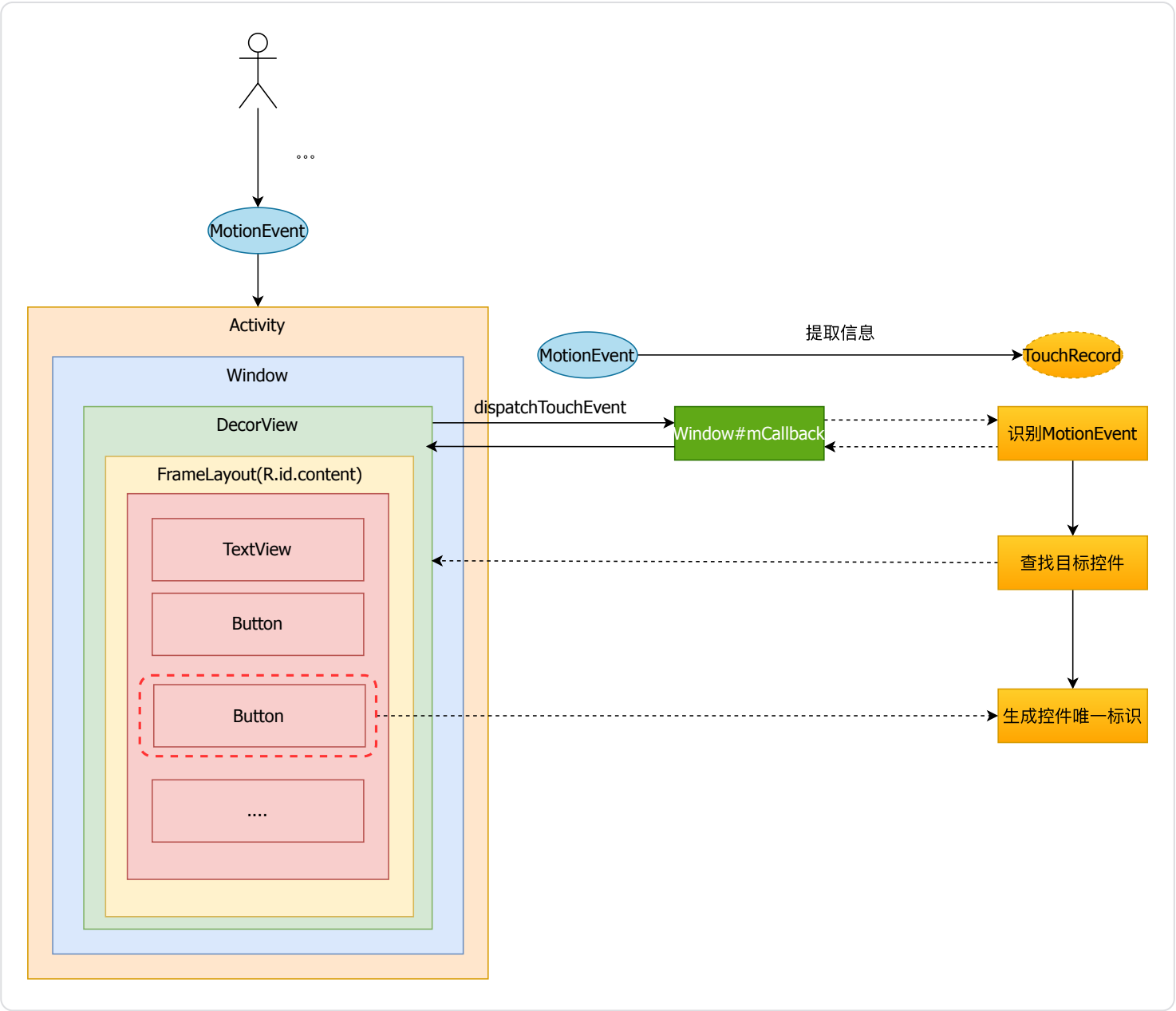
目录

> GitHub-DiDiPrism
>
> Android README
>
> Doc/系列文章/小桔棱镜-专注移动端操作行为的利器

一、基本流程


小桔棱镜 SDK
三个维度：
1. 被点击view所在窗口(Window)信息
2. 被点击view在ViewTree上的路径信息
3. 被点击view自身可提取的信息
用户操作
(触摸)
生成View唯一标识
采集
解析
回放
控件响应交互事件
构造并分发点击事件
查找对应View
UBC SDK
UBC 打点方法
...

二、采集原理


...
MotionEvent
Activity
Window
DecorView
FrameLayout(R.id.content)
TextView
Button
Button
....
MotionEvent
提取信息
TouchRecord
dispatchTouchEvent
Window#mCallback
识别MotionEvent
查找目标控件
生成控件唯一标识

2.1 获取被点击的控件

> 简单列一下目前一些主流方案：
>
> - 【无障碍功能】遍历 `Activity` 视图中所有的 `View`，给 `View` 设置 `AccessibilityDelegate`
> - 【ASM字节码插桩】通过 `ASM` 给 `click` 等事件插入代码
> - 【代码架构】通过继承 `baseActivity` 或 `baseDialog` 的方式，在 `dispatchTouchEvent ACTION_DOWN` 时，结合 `TouchTarget` 获取

主要流程

- 基于 `Andorid` 事件分发机制，通过 `Java` 反射代理 `WindowManagerGlobal#mViews` 对象，监听每个被添加的 `DecorView`，并获取对应的 `Window` 实例；
- **为 `Window` 设置 `Callback` 监听回调事件，以获取到事件分发回调**，如 `dispatchTouchEvent`、`dispatchKeyEvent` 等；
- 识别一次触摸事件，依据触摸位置在 `View` 树中自顶向下查找响应本次触摸事件的控件 *`TargetView`*

2.2 根据View生成唯一标识

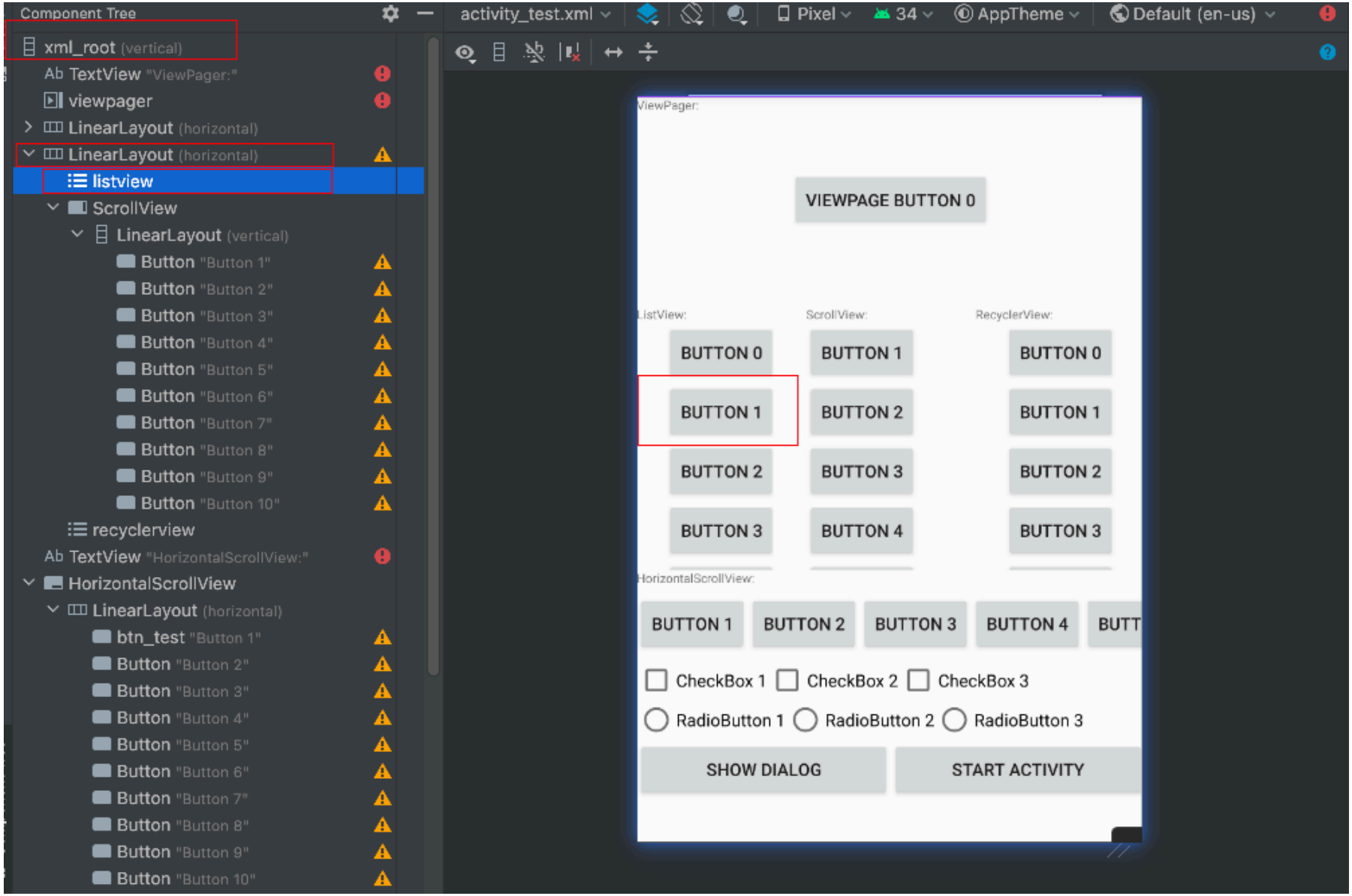
目前一些主流方案会将一些关键信息按约定的格式组合而成，作为唯一标识。这些关键信息有：Activity类名、View Id或Resource Id名称、View Class名称、View Path等。不同方案有不同的考量，优化的方式也不同。 下面就从三个维度，介绍下小桔棱镜的做法：

1. **被点击view所在窗口信息** 我们并没有直接使用Activity类名，因为被点击的view除了发生在Activity中，还有发生在Dialog，所以我们加了一层窗口的逻辑，也就是窗口的类型，使用 `w` 表示窗口信息，格式如下： `w_&_{窗口名称}_&_{窗口类型}`，窗口名称其实也就是Activity类名，窗口类型有0或1，表示Activity或Dialog，由 `_&_` 连接，比如 `w_&_com.prism.MainActivity_&_0`。
2. **被点击view在ViewTree上的路径信息** 我们不记录每个view层级上的View Class名称或者index，只会记录关键层级，使用 `vp` 表示View Path信息。举几个例子：
   - 当层级上的view能获取到view id时，比如： `vp_&_titlebar_item_left_back/thanos_title_bar/content[01]/`，其中 `content[01]` 表示系统自带的那个 `android.R.id.content`，通过 `[01]` 区别。
   - 当层级上的view类型有 `AbsListView` 或 `RecyclerView` 时，比如： `vp_&_*/listView/navigation_drawer/drawer_layout/content[01]/_^_vl_&_l:4,10`，其中 `*` 表示被点击的那个ListView item层级，另外使用 `vl` 来描述可复用容器item的信息， `l:4,10` 表示AbsListView容器中第4位，在数据源中第10位。
3. **被点击view自身可提取的信息**

- 当view存在id时，使用 `vi` 表示id信息，比如： `vi_&_titlebar_item_left_back`
- 当view存在drawable等资源时，使用 `vr` 表示资源信息，比如： `vr_&_selector_btn_confirm`
- 当view存在可提取文本信息时，使用 `vc` 表示文本信息，比如： `vc_&_确定` (代码里用的也是vr)

最后，将以上三个维度提取的信息通过 `_^_` 连接起来，作为被点击view唯一标识。

示例




```
w_&_com.xiaojuchefu.prism.TestActivity_&_1_^_vi_&_btn_^_vcn_&_FeedOneImgView_^_vp_&_btn/*/listview/xml_root/content[01]/decor_content_parent/_^_vl_&_l:1,1_^_vr_&_BUTTON 1_^_vq_&_100
```

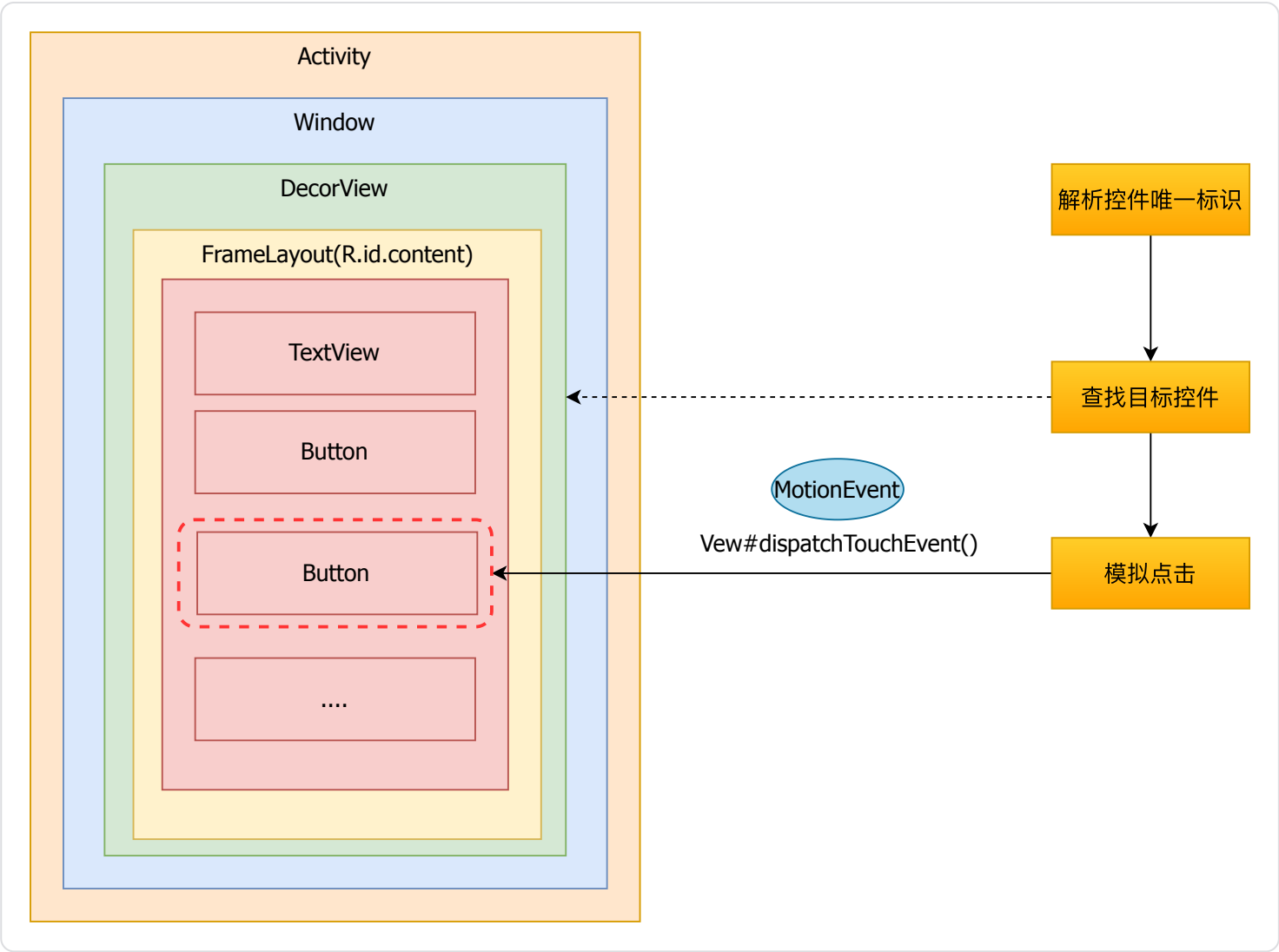
- 被点击view所在窗口信息

    - `w_&_com.xiaojuchefu.prism.TestActivity_&_1`
        - 窗口名称：`com.xiaojuchefu.prism.TestActivity`
        - 窗口类型：`1` (Activity)
- 被点击view在ViewTree上的路径信息
    - `vp_&_btn/*/listview/xml_root/content[01]/decor_content_parent/`
        - View路径
    - `vl_&_l:1,1`
        - View列表项描述：在列表中排第2位，数据源中的排第2位
- 被点击view自身可提取的信息
    - `vi_&_btn`
        - ViewID
    - `vr_&_BUTTON 1`
        - View资源描述：文本内容是 `BUTTON 1`
    - `vq_&_100`
        - View区位信息
    - `vcn_&_FeedOneImgView`
        - View类名，新增字段；适配需要匹配控件类型的场景，作为兜底策略，默认不开启

2.3 关键类&方法

- `PrismMonitorWindowCallbacks#touchEvent()`
- `TouchTracker#findTargetView()`

三、回放的原理



3.1 查找被点击的控件

主要流程

- 解析 `View` 唯一标识，在布局 `View` 树查找指定 `View` (对于列表项中的 `View` ，滚动到指定的下标位置展示出 `Item`)
  - 通过ViewID查找

- 通过 `ViewPath` 查找
- 通过 `View` 资源引用查找

3.2 发送模拟事件

- 对指定 `View` 分发自定义的事件模拟点击操作

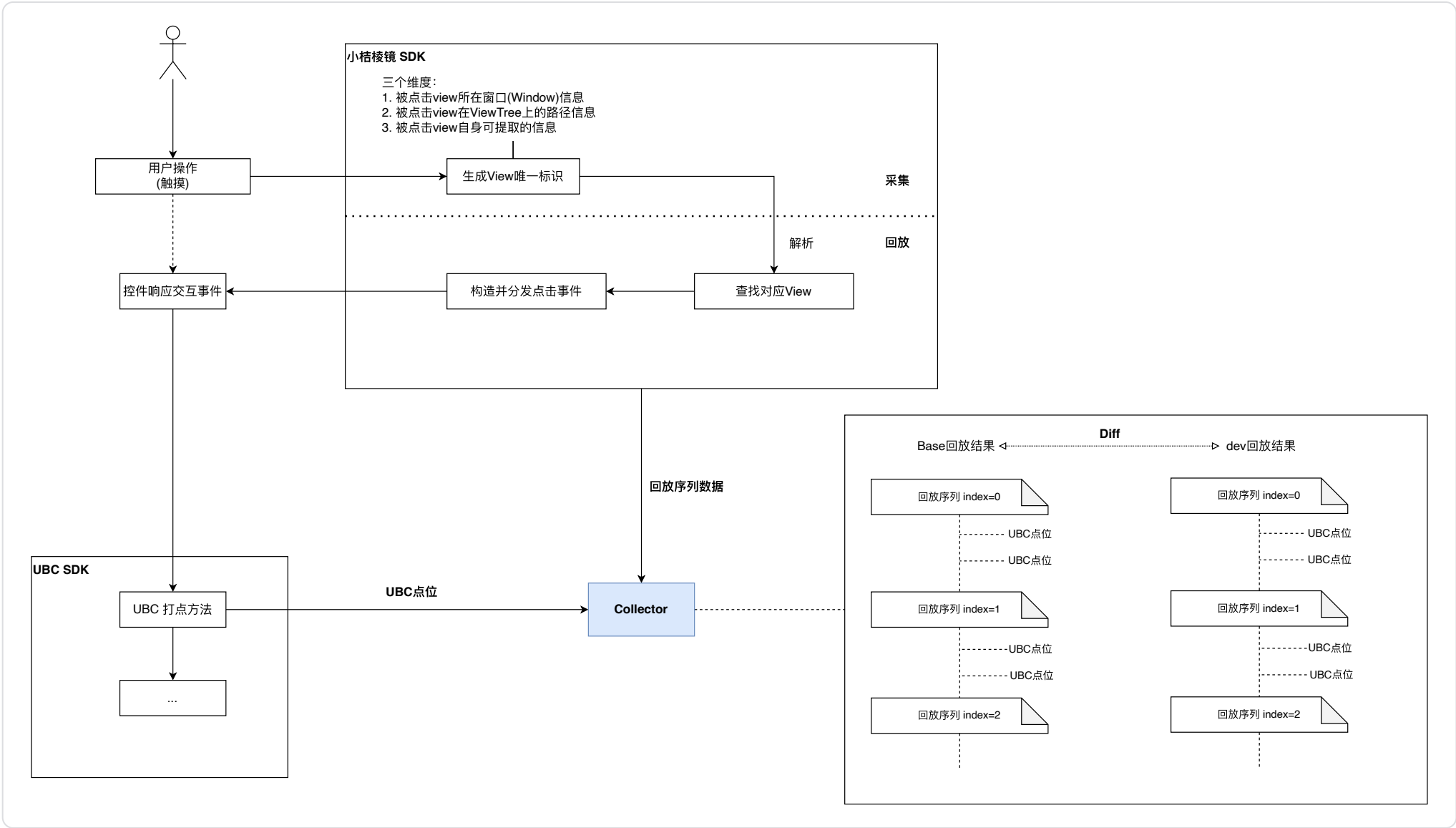
3.3 关键类&方法

`PlaybackHelper#findTargetView()`

`MotionHelper#simulateClick()`

四、回放点位Diff

回放期间通过 `Collector` 收集UBC打点信息，产出Base回放结果作为Diff的基准数据，比对dev回放结果；


小桔棱镜 SDK
三个维度：
1. 被点击view所在窗口(Window)信息
2. 被点击view在ViewTree上的路径信息
3. 被点击view自身可提取的信息
用户操作
(触摸)
生成View唯一标识
采集
解析
回放
控件响应交互事件
构造并分发点击事件
查找对应View
回放序列数据
UBC SDK
UBC 打点方法
...
UBC点位
Collector
Diff
Base回放结果
dev回放结果
回放序列 index=0
UBC点位
UBC点位
回放序列 index=1
UBC点位
UBC点位
回放序列 index=2
回放序列 index=0
UBC点位
UBC点位
回放序列 index=1
UBC点位
UBC点位
回放序列 index=2

TODO：

1. 手百线下集成，验证效果