**UI自动化测试侵入式框架**

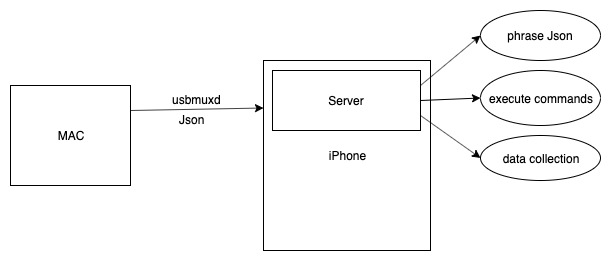
**背景**

安卓目前已经有一个成熟的UI自动化测试框架AndroidSpy ，而iOS在市面上已经使用的框架里面没有类似实现的框架，于是自研一个iOS端的自动化测试框架就提上了日程。

**要求**

按照安卓的实现思路来说，我们需要做成一个framework的形式，在带测试app接入sdk后，我们可以在app里面建立一个server，这个server监听mac传来的数据，对于数据进行解析从而执行命令

**总体设计**



如图，在mac和iPhone之间利用进行通信，传输Json数据，在宿主app中，sdk可以建立一个server监听端口发送来的Json数据，解析数据执行具体的命令，也可以捕获性能数据，对数据处理后同样通过端口转发给mac

**具体设计**

**框架的启动**

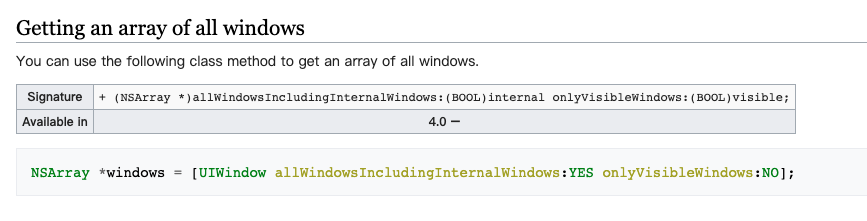
Mach-o的动态注入方法，可以避免修改业务方代码，以及在没有源码的情况下可以直接注入，只需要保持签名一致即可

**通信**

通信方面最开始参考意见，准备采用GCDWebDriver进行通信，GCDWebDriver的介绍参考预研文章[UI自动化测试框架前期调研](https://bytedance.feishu.cn/space/doc/doccnAQiQyahUvCYw8FZcfZkcwe) ，目前设计是通过该框架在应用本身建立一个服务器，可以收发HTTP请求，通过请求携带的json数据完成指令的下发和数据的回传

**控件树拉取**

页面控件树的拉取参考FLEX的实现，具体方法为调用UIWindow的私有方法来获取当前页面的所有控件：



在FLEX里面具体实现为：

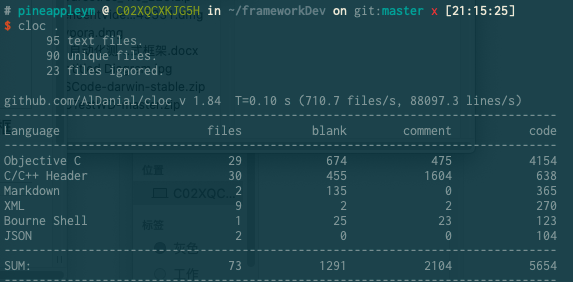
1. + (NSArray<UIWindow \*> \*)allWindows
2. {
3. **BOOL** includeInternalWindows = **YES**;
4. **BOOL** onlyVisibleWindows = **NO**;
5. // Obfuscating selector allWindowsIncludingInternalWindows:onlyVisibleWindows:
6. NSArray<NSString \*> \*allWindowsComponents = @[@"al", @"lWindo", @"wsIncl", @"udingInt", @"ernalWin", @"dows:o", @"nlyVisi", @"bleWin", @"dows:"];
7. **SEL** allWindowsSelector = NSSelectorFromString([allWindowsComponents componentsJoinedByString:@""]);
8. NSMethodSignature \*methodSignature = [[UIWindow class] methodSignatureForSelector:allWindowsSelector];
9. NSInvocation \*invocation = [NSInvocation invocationWithMethodSignature:methodSignature];
10. invocation.target = [UIWindow class];
11. invocation.selector = allWindowsSelector;
12. [invocation setArgument:&includeInternalWindows atIndex:2];
13. [invocation setArgument:&onlyVisibleWindows atIndex:3];
14. [invocation invoke];
15. **\_\_unsafe\_unretained** NSArray<UIWindow \*> \*windows = **nil**;
16. [invocation getReturnValue:&windows];
17. **return** windows;
18. }

**点击实现**

点击的实现较为复杂，由于iOS本身的封闭性，需要对于系统本身的多个控件进行封装，并需要hook IOKit进行点击事件的发起，具体实现在此不列出。

**预计工作量**

完全工作量大概在10k代码以上，目前已经写了一小部分，代码量见图

****

**任务分解**

1. **iOS端服务器架构**
2. **iOS和Mac通信设计**
3. **mac端软件适配**
4. **后端服务器搭建**
5. **iOS端控件树拉取**
6. **iOS端控制点击完成**
7. **iOS端数据监控设计**
8. **iOS端crash信息dump采集**
9. **iOS更多功能的完善**
10. **Mach-o动态注入完成**
11. **整套流程自动化完善**
12. **对于市面上较为知名的app进行mach-o注入测试框架的效果**