基于布局与交互分析的安卓应用界面模式抽取与检索*

吴俊伟[1,2], 沈立炜[1,2+], 郭武楠[1,2], 王 超[1,2], 赵文耘[1,2]

[1](复旦大学计算机科学技术学院,上海 201203)

[2](上海市数据科学重点实验室（复旦大学）,上海 201203)

Layout and Interaction Analysis based UI Pattern Extraction and Retrieving of Android Applications

WU Jun-Wei[1,2], SHEN Li-Wei[1,2+], Guo Wu-Nan[1,2], WANG Chao[1,2], ZHAO Wen-Yun[1,2]

[1](School of Computer Science, Fudan University, Shanghai 200433, China)

[2](Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 200433, China)

+ Corresponding author: Phn +86-21-5135-5343, Fax +86-21-5135-5357, E-mail: shenliwei@fudan.edu.cn

Received 2000-00-00; Accepted 2000-00-00

Abstract: Android developers need to accumulate experience to enhance their ability to design Android interface and behavior. In order to reduce developers' effort of selecting, using and learning Android application, this paper proposed a method of layout and interaction analysis based UI pattern extraction and retrieving of Android applications. The UI pattern of an activity represents the interface layout and the interactive behavior of the activity. Taking the pattern as the target, this method extracts the UI pattern of each activity from a set of open source Android applications. On this basis, the method supports users to retrieve the related design details of activities by constructing queries. The method is implemented as a set of tool chains that provide automation support for extracting and retrieving. Furthermore, the accuracy and effectiveness of the method are verified by two working examples.

Key words: Android; User Interface Pattern; Pattern Extraction

摘 要: 安卓开发者需要通过不断地经验积累来提升其设计安卓界面及行为交互的能力。为了降低开发者在选择、试用、学习安卓应用方面所带来的人工成本，本文提出了一种基于布局与交互分析的安卓应用界面模式抽取与检索方法。以安卓活动（Activity）为单元的界面模式代表了活动的界面布局及交互行为。本方法以此为分析目标，从一组开源的安卓应用项目中抽取出每一个活动的界面模式。在此基础上，支持用户通过检索的方式获取与其需求相关的活动设计细节。该方法被实现为一组工具链，为抽取与检索提供自动化支持。最后，通过两个案例对本方法的准确性与有效性进行验证。

关键词: 安卓; 界面模式; 模式抽取

* Supported by the National Natural Foundation of China under Grant No. 61402113, 安卓应用开发中模式驱动的代码推荐与完成技术研究.

作者简介: **吴俊伟**(1979－),男,上海人,硕士研究生,主要研究领域为安卓代码推荐技术;**沈立炜**(1982－),男,博士,副教授,主要研究领域为软件分析技术，移动应用开发技术;**郭武楠**(1993－),男,本科,主要研究领域为软件代码推荐技术;王超(1992－),男,本科,主要研究领域为移动应用分析;赵文耘(1964－),男,硕士,主要研究领域为软件工程.

中图法分类号: **** 文献标识码: A

1 概述

据 Gartner 统计，基于安卓操作系统的移动设备出货量逐年剧增，且安卓操作系统更占据了 86.1%的市场份额[1]。安卓移动应用的普及使得越来越多的从业者进入安卓开发领域。为了提高用户使用移动应用的粘性，安卓开发者除了必须保证应用的功能正确性之外，还需要确保应用具有较高的可用性与易用性。

在安卓框架的四大组件中，活动（Activity）是包含用户界面的组件，是与用户进行交互的直接渠道。用户从提供可视化界面的 Activity 中获取信息。例如，资讯类 App（如今日头条）中的新闻页面即是一个 Activity 的实例，该页面包含一个列表由上自下列举新闻的概要内容。页面中也包含一个搜索框允许用户输入关键字从而检索相关新闻。另一方面，Activity 也负责对用户的动作进行响应，完成数据刷新或页面跳转等功能。例如在上述示例 App 中，用户可通过下拉列表的方式加载最新新闻。另外，用户也可通过点选列表中的某项新闻内容从而使得 App 跳转至该新闻的详细内容页面。依据安卓编码规范，Activity 的开发涉及界面布局和交互逻辑这两个部分。界面布局一般通过定义 layout 文件来指定该 Activity 包含的控件以及这些控件之间的位置关系。除此之外，也可通过编码方式在运行时动态添加控件。交互逻辑一般通过定义监听器的方式来响应用户在 Activity 上执行的动作，例如按钮点击、从下拉框中选择项目等。事件响应一般具有两种效果。一种是影响所在 Activity 中的其他控件（例如使列表重新刷新）；另一种是跳转至新的 Activity。

Activity 的开发技术虽然简单，但其布局与交互逻辑的设计则需要大量的经验积累。在安卓应用可随处获取的情况下，下载并使用各种类型的 App、学习当前流行的 Activity 设计方案能够为开发者带来启示。然而，这一方式无法在短期内实现大量应用界面的学习。另外，对开发者，尤其是新手而言，总结出 Activity 界面的交互逻辑也是较为困难的。

针对这一问题，本文提出了一种基于布局与交互分析的安卓应用界面模式抽取与检索方法。该方法聚焦于单个 Activity 的界面布局以及交互行为，从一组开源的安卓应用项目中抽取出每一个 Activity 的界面模式，支持用户通过检索的方式获取与其需求相关的 Activity 设计方案，从而为用户的开发工作提供帮助。具体而言，抽取处于预处理阶段，旨在通过自动化手段分析每一个安卓应用的每一个 Activity 的界面模式，厘清界面元素的层次结构及其事件行为。检索是预处理完成后面向用户的阶段，依据用户的设计需求从已抽取的界面模式中查找出最为接近的 Activity 界面模式，并反馈其实现代码和资源。在实现方面，本文结合并扩展 GATOR 与 FlowDroid 工具组成一条工具链，为安卓项目 Activity 界面模式的抽取提供自动化手段。另外，提供一套基于 JSON 的查询模板（PQT）支持用户对 Activity 设计需求的表述，并基于该需求实现相关界面模式的检索与结果排序。最后，本文通过两个实例对本方法的准确性与有效性进行验证。

本文的剩余结构如下：第二节介绍与本文方法相关的研究工作以及依赖的开源工具；第三节介绍方法框架以及每一阶段的方法策略；第四节介绍工具链的设计方案；第五节列举了应用案例；最后是本文的总结与未来工作的展望。

2 相关工作

开发者在开发过程中通过检索获得实例结果的场景属于安卓代码推荐的范畴。在之前工作中[2]，针对安卓开发中的代码推荐机制进行了经验性研究，将其分为代码搜索、基本推荐以及高级推荐这三种类型。代码推荐基于用户构造的查询语句从各类资源库中查找相关实现，其搜索层次可覆盖 API 的使用模式[3]、示例代码[4]等。与这些已有工作相比，本文提出方法的搜索目标聚焦在 Activity 的界面模式上。

UI 分析是与本文方法较为相关的研究领域。Kumar 等人提出了名为 Webzeitgeist 的平台支持大规模网页设计的挖掘[5]。该平台使用知识发现的技术理解网页的 DOM 设计决策并支持用户查找与其需求相关的网页实例。在此工作的基础上，Deka 等人开发了 ERICA 系统对安卓应用进行交互挖掘[6]，捕获应用设计中的静

态（UI 布局、可视化细节）和动态（用户流程）决策，并在此挖掘基础上将已有的应用交互分类为 23 种模式。本文方法受到以上两个工作的启发，在安卓界面模式搜索方面进行了初始的研究。本文同样采用知识发现技术来挖掘安卓应用的界面结构与交互行为，但将检索的目标局限于单个 Activity 之上。

安卓应用的界面模式是安卓设计中的重要环节。在该领域中，Nguyen 等人提出 REMAUI 方法能够分析位图中的界面元素并自动生成针对界面的原生代码[7]。Alharbi 等人从安卓项目的历史版本中通过数据挖掘的方法分析界面设计模式的演化[8]。Shirazi 等人则分析了大量安卓应用的界面结构，得到不同类型应用的界面差异性较大的结论[9]。另外也统计得到了最常出现的界面元素组合模式。与这些工作相比，本文方法所定义的界面模式不仅考虑 Activity 的元素结构，同时也将界面元素的事件行为作为检索的重要依据。

本文方法涉及对安卓 Activity 的界面及其事件行为的分析。当前已有诸多技术聚焦在安卓应用分析领域并发布了多种工具实现自动化分析。作为本文工具链中的重要组成部分，GATOR[10]和 FlowDroid[13]分别在界面分析和方法调用两方面展示其效果。

GATOR 是面向安卓的程序分析工具集，主要包含两个组件，分别支持对 GUI 对象的静态分析，以及对事件和回调的控制流分析。前者模拟 GUI 的对象流、界面相关操作以及这些对象相关联的结构关系，得到 Activities 和 Listeners 相关联的视图 [10]。后者则采用上下文敏感分析方法[11]和窗口堆栈的显示建模[12]来实现 GUI 模型的生成并保证高精度的控制流分析。GATOR 工具的输入是编译完成的安卓项目，通过 GUIAnalysisClient 接口将分析得到的 GUI 信息输出。另外，它允许用户定制 GUIAnalysisClient 接口的实现，从而支持自定义的数据持久化等能力。

FlowDroid 是一个上下文相关、执行顺序相关且用于安卓应用的静态污点分析工具[13]。它可根据安卓中每个组件均具有的回调函数绘制出生命周期相关的控制流图。FlowDroid 基于 soot 和 Heros，使用一个精确的调用图以确保顺序相关和上下文相关；另一方面，IFDS 框架保证了字段相关和对象相关[14]。对于一个给定的安卓 apk 文件，FlowDroid 能对其进行分析并产生对应的函数调用关系图。

3 方法介绍

3.1 安卓应用界面模式

在本文中，将界面模式（Activity_UI_Pattern，缩写为 AUP）定义为以单个 Activity 为单元的、表示该 Activity 内布局结构及其交互行为的信息封装体。图 1 展示了安卓应用界面模式的元模型以及 AUP 的规范化定义。

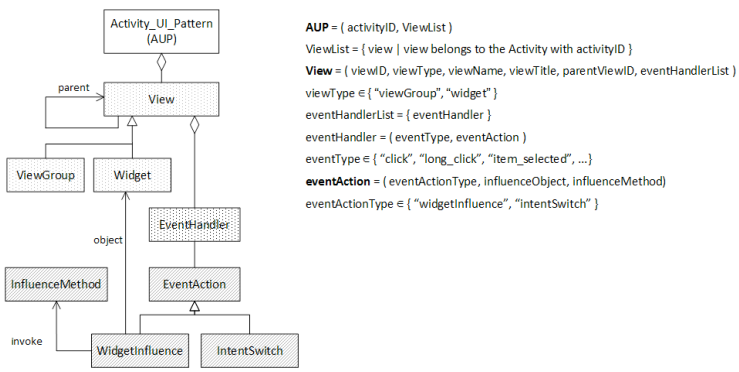

图 1. 安卓应用界面模式的元模型及规范化定义

AUP 由一组隶属于特定 Activity 的 View 构成，每一个 View 是组成安卓界面的视图控件。一组 View 构成树形结构，形成了 Activity 的界面布局。View 可分为两种类型，分别是 ViewGroup 和 Widget。ViewGroup 是容纳其他 View 的容器，即 ViewGroup 可以是其他 View 的父节点。他们用于指定所容纳的其他 ViewGroup

或 Widget 之间的相对位置关系。一般而言，ViewGroup 是界面上的不可见元素。例如，LinearLayout 仅负责将其内部控件按照横向或纵向方向进行排列；ScrollView 在界面上展现为一个允许用户上下拖拉的容器。相比之下，Widget 是可见的并且可与用户交互的控件元素。例如，EditText 允许用户输入数据，ImageView 能加载并展示一张图片。在由 View 所构成的界面树形结构中，ViewGroup 一般是内部节点，而 Widget 一般处在叶子结点的位置。在实现维度，View 的层次结构可在项目的 layout 资源文件中定义。layout 内容基于严格的 Schema 结构，描述了 ViewGroup 以及 Widget 之间的层次关系。安卓开发者也可通过编码灵活地控制界面上的 View 元素及布局。在 AUP 的规范化定义中，一个 View 是由六项内容构成的元组，分别是表示 View 唯一编号的 ID，表示其类型的 Type（ViewGroup 或 Widget），View 的全路径类名，展示界面字符的 Title，该 View 在层次树上的父节点（若该 View 为根节点，该 View 的 parentView 为 Activity_Root），以及一组事件句柄（eventHandlerList）。

EventHandler（事件句柄）是 View 响应用户操作的逻辑入口，一个 View 可能具有多种事件句柄。由于安卓框架具有事件驱动（event-driven）的特色，因此赋予了开发者定义事件监听器并在回调函数中实现交互操作的能力。例如，Button 控件上可附加 Click 监听器，并定义响应事件，从而使应用程序能够获知用户点击按钮的场景并实现相应的功能。一般可通过四种方式定义事件句柄：匿名内部类、自定义事件监听类、Activity 实现事件监听器接口并将 Activity 作为控件监听器，以及在 layout 文件中定义事件并通过反射方式调用方法。每一个事件句柄具有相应的事件行为。

EventAction（事件行为）是表示 EventHandler 行为的抽象元素，表示用户在 View 上执行动作之后所触发的功能类型。事件行为分为控件影响（WidgetInfluence）与页面跳转（IntentSwitch）两种类型。前者是在事件句柄的回调函数中对其他控件进行了操控，即调用了本 Activity 内其他控件的方法。这类行为的信息包括影响的控件对象（一般为可见的 Widget 对象）以及所施加的影响行为（即调用的方法）。例如，重刷新 ListView 即是在回调函数中针对 ListView 的 Widget 对象调用其 onRefresh 方法。后者则表示在事件回调函数中离开本 Activity 并跳转至新的 Activity 的逻辑过程。将事件行为规范化定义为一个涵盖事件行为类型、影响对象以及影响方法的元组。当该事件行为是页面跳转类型时，元组中的后两项内容无须填充。

3.2 方法过程

图 2 是本文提出的安卓应用界面模式抽取与检索方法的框架示意图。方法主要分为两个阶段。首先，界面模式抽取是从安卓项目代码中构建界面模式的阶段。该阶段包含 view 层次识别、event 行为识别以及综合前两项识别结果构建面向 Activity 界面模式这三个子活动。构建出的界面模式被持久化存储于界面模式库中，用于后续检索与复用。其次，界面模式搜索是安卓开发者在应用设计过程中向界面模式库检索并获得反馈的阶段。开发者将设想中的待开发 Activity 的界面构造为查询请求，随后基于该请求表述从安卓界面模式库中检索出具有相关性的 Activity 实例。同时，针对返回的结果根据其与实际需求的相似性进行排序。

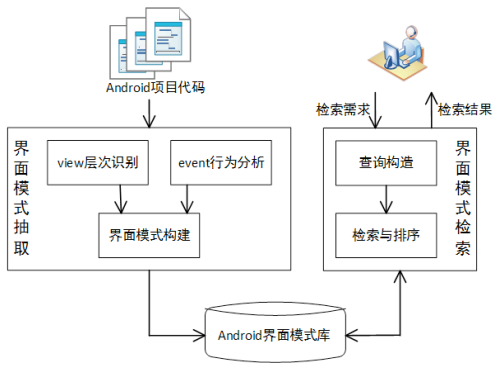

图 2 安卓应用界面模式抽取与检索方法框架

3.3 界面模式抽取

界面模式抽取步骤旨在从安卓项目中分析出每一个 Activity 的界面模式，包括其 View 层次结构以及交互行为。得益于 GATOR [10]以及 FlowDroid [13]，该步骤能够快速地在预处理阶段获得构成 Activity 界面模式的主体内容。

（1）通过 GATOR 进行 view 层次识别

针对一组安卓项目，使用 GATOR 工具能够获得每一个安卓 App 中每一个 Activity 的 View 层次结构以及 View 具有的事件句柄，即覆盖图 1 所示元模型中的虚点图案的元素。GATOR 工具针对单个 Activity 运行后得到层次结构 ViewHierarchy，定义为：

ViewHierarchy = ViewList = {View}，即一个 Activity 的 UI 层次结构是一组 View 的集合；

View = (viewID, viewType, viewName, viewTitle, parentView, eventHandlerList)，与元模型中 view 的定义一致；

EventHandlerList = {eventHandler}；

eventHandler = (eventID, eventType, eventHandlerMethod)，一项事件句柄包括其唯一编号 ID，表示其事件类型的 Type，以及在代码中实现该事件的方法名。

（2）通过 FlowDroid 进行 event 行为分析

FlowDroid 可用于分析安卓 App 内的方法调用关系，即包括 View 事件句柄的响应函数中的方法调用。其分析结果表示为如下结构：

MethodInvocations = {MethodInvocationPair}；

MethodInvocationPair = (sourceMethod, targetMethod)，即由源头方法（调用方）与目标方法（被调用方）所构成的元素对；

sourceMethod = (sourceMethodClass, sourceMethodName)；

targetMethod = (targetMethodClass, targetMethodName),即源头方法与目标方法均由方法所在的类及方法名构成。

（3）结合 GATOR 与 FlowDroid 结果的界面模式构建

为了构成图 1 所示元模型中的斜线图案元素，完成单个 Activity 的界面模式的构造，需要分析并识别事件句柄中涉及的行为动作。

图 3 的算法描述了结合 GATOR 分析结果与 FlowDroid 分析结果得到完整的界面模式（AUP）的过程。该算法的核心点是将 GATOR 分析得到的事件句柄方法与 FlowDroid 分析得到的源头方法进行匹配，从匹配得到的目标方法中识别行为动作，按规则将其确定为控件影响动作或者页面跳转动作。

```
Function GenerateAUP
Input: ViewHierarchy,              //通过 GATOR 分析得到的 AppX 中某个 Activity 的 View 层次结构与事件句柄
       MethodInvocations           //通过 FlowDroid 分析得到的 AppX 的方法调用对
Output: AUP                        //AppX 中该 Activity 的界面模式
begin
  AUP = ( activityID, ViewHierarchy )      //将 AUP 初始化为未包含事件动作的 ViewHierarchy
  for each View in ViewHierarchy           //遍历层次树中的所有 View
  begin
    for each eventHandler in View.eventHandlerList   //遍历该 View 可能具有的所有事件句柄
    begin
      targetMethodList = {}
      for each MethodInvocationPair in MethodInvocations
      begin
        if MethodInvocationPair.sourceMethod == eventHandlerMethod then   //当源头方法为事件句柄方式时
        begin
          if MethodInvocationPair.targetMethod.targetMethodClass = any viewType of View then
          //存在目标方法的类名为该 Activity 中 View 的类型名
          begin
```

```
                //构造并添加 eventHandler 的类型为 widgetInfluence 的 eventAction 元素
                eventAction = (‘widgetInfluence’, targetMethodClass, targetMethodName)
                eventAction -> eventHandler
            end
            if ((targetMethodName==’startActivity’) or (targetMethodName == ‘startActivityForResult’)) then
            begin
                //构造并添加 eventHandler 的类型为 intentSwitch 的 eventAction 元素
                eventAction = (‘intentSwitch’)
                eventAction -> eventHandler
            end
          end
        end
      end
    end
    return AUP
end
```

图 3. 构造 Activity 界面模式（AUP）的算法过程

3.4 界面模式检索过程

为了使用户能够检索已被抽取的界面模式，本文采用基于 JSON 的查询模板（PQT）来结构化表述待开发 Activity 界面的设计需求。后台应用程序能够读取并分析该需求，实现界面模式的检索与结果反馈。

（1）PQT 与查询构造

基于 PQT 的查询实例（query）是一个 JSON 对象，用于检索的键包括 viewType、viewName、events 和 subViews。前两项表示 view 的类型与类名。events 是代表事件句柄与动作的数组，其中每一项 event 是由 eventType 与 eventAction 构成的对象，eventAction 则是由 actionType、targetWidget 和 influenceMethod 构成的对象。这几项模板内容与图 1 所示元模型对应。subViews 表示本 view 所容纳的子视图数组，每一个数组元素是嵌套定义的表示 view 的 JSON 对象。

PQT 可用于描述具一组 View，基于 PQT 的查询实例通过 JSON 的方式定义了这组 View 的属性与层次关系，查询实例的 JSON 对象一般是容纳子视图的 ViewGroup。另外，PQT 模板在使用时也可根据实际需求进行裁剪，即当需求未涉及某项内容时可移除表示该内容的键。图 4 展示了一个查询实例，该实例表示用户所期望的 Activity 界面是由一个文本框（TextView）以及一个按钮（Button）所构成，并且按钮的点击事件会引起文本框内值的改变。

```
query = {
  "viewType": "ViewGroup",
  "viewName": "android.widget.LinearLayout",
  "subViews": [
    {
      "viewType": "Widget",
      "viewName": "android.widget.TextView"
      "events": [ ]
    }
    {
      "viewType": "Widget",
      "viewName": "android.widget.Button",
      "events": [
        {
          "eventType": "click",
          "eventAction": {
            "actionType": "WidgetInfluence",
            "targetWidget": "android.widget.TextView",
            "influenceMethod": "setText"
          }
        }
      ]
    }
  ]
}
```

图 4. 基于 PQT 的查询实例

（2）检索与排序

一般而言，用户在其期望的界面模式中更着重于可见控件（Widget）的设计，而对 Layout 布局则并不是特别关注。为了向用户提供更广泛的界面模式反馈，方法的检索过程将更聚焦于对界面可见 Widget 元素的匹配，而忽略 Layout 等用于控制 Widget 布局的容器元素。在此基础上，将针对界面模式的检索进一步分为三个层次。

第一个层次是最严格的层次，即搜索符合 Widget 类型、数量，以及事件动作全部条件的 Activity 界面模式。

第二个层次相对第一个层次稍显弱化，即需要符合 Widget 类型与数量，但只需要满足事件句柄条件而不一定满足事件内动作的界面模式。第二层次的查询结果显然涵盖前一个层次，在给出具体结果时，为了消除重复，移除所包含的第一个层次的查询结果。

第三个层次是最弱化的层次，即只需要符合 Widget 类型与数量，而不需要匹配事件动作的界面模式。同样，在给出具体结果时移除前两个层次的重复部分。

界面模式的检索与排序过程如图 5 所示。该过程分为查询条件组织、分条件查询、多层次查询结果合并以及查询结果排序这几项活动。

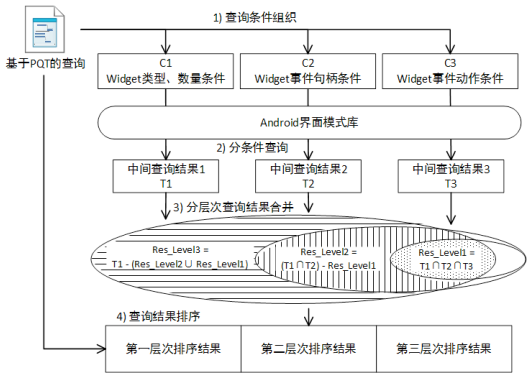

图 5. 界面模式的检索与排序过程

1) 从三个检索层次中可以得知，满足查询需求的界面模式是从已抽取的界面模式中通过匹配各项条件过滤得到的。这些条件包括 C1、C2 与 C3。C1 指明了特定 Widget 的类型与数量（例如，期望在 Activity 中包含 2 个 Button 控件）；C2 指明了 Widget 的事件句柄（例如，期望 Activity 中的 Button 控件响应 Click 事件）；C3 则指明 Widget 的事件动作（例如，Button 的 Click 事件能够更改 TextView 的值）。基于此定义，从基于 PQT 的查询文本中分类组织这些条件。

2) 使用三类查询条件，分别从界面模式库中匹配得到符合特定条件的中间查询结果（AUP 的集合），分别命名为 T1、T2 与 T3。根据查询条件的语义，T2 包含 T3，但 T2、T3 与 T1 不一定具有包含关系。

3) 根据不同的检索层次，通过将不同条件的 AUP 集合进行取交集与取差集操作，得到满足特定检索条件的结果，命名为 Res_Level1、Res_Level2 和 Res_Level3。在第一层次查询中，Res_Level1 为 T1、T2 与 T3 的交集，得到符合所有条件的最严格的查询结果；在第二层次查询中，Res_Level2 为 T1 与 T2 的交集并且去除属于 Res_Level1 的重复结果；在第三层次查询中，Res_Level3 为最弱化的 T1 移除 Res_Level1 和 Res_Level2 后得到的结果。

4) 对同一类检索层次中的 AUP 进行排序。排序的依据是查询需求与各界面模式在 Layout 层次结构方面的相似性。该相似性数值计算方法借鉴文献[17]所提出的基于树相似度的计算方法：首先将各 AUP 与查询需求组织为层次树形结构，并将树中结点按照从高层（离根结点近的层次）到低层（离根结点远的层次）的顺序赋予由大到小的权重；随后识别出两棵树中最大公共子树并计算子树中节点的权重总和；最后计算子树

权重总和在两棵树所有结点的权重总和中的比例得到 AUP 与查询需求的相似度数值。相似度越高，说明该 AUP 与查询需求的结构更为接近，在层次中的排名也更靠前。

4 工具实现

本文针对 F-Droid 中的一组开源安卓项目，综合已有的安卓分析工具形成了一条工具链，实现应用界面模式的抽取与检索这两个阶段的任务。工具链的架构如图 6 所示。其中，灰色底色框表示采用已有工具，白色底色框为自行开发工具或对已有工具的扩展。在预处理阶段，使用该组工具进行抽取的过程包括：

（1）从 F-Droid 中获得安卓项目的代码地址，并使用爬虫工具下载最新版本的安卓项目至本地服务器。

（2）自动编译安卓源代码，生成项目的 bin 目录。该目录是进行 Soot 分析（GATOR 与 FlowDroid 均建立在 Soot 之上）的前提。

（3）使用 GATOR 工具对每一个安卓应用进行 UI 分析，获得 Activity 的 View 层次结构。与 Activity 相关的代码文件以及资源文件存储至 NoSQL 数据库（采用 MongoDB）。在 GATOR 分析基础上，通过实现 GUIAnalysisClient 接口将 View 结构信息保存至 SQL 数据库（采用 MySQL）。

（4）使用 FlowDroid 工具对每一个安卓应用进行方法调用分析，获得每一个应用所对应的 gexf 文件（可使用 Gephi 工具打开并展示方法调用图）。开发读取 gexf 文件的应用将方法的调用关系保存至 SQL 数据库内。

（5）界面模式构造应用程序读取 SQL 数据库内已存储的 GATOR 分析结果与 FlowDroid 分析结果，依据图 3 所示算法关联事件行为，并将关联信息扩展至数据库。

在检索阶段，工具的使用场景包括以下步骤：

（1）针对用户提供的基于 PQT 的检索需求，通过检索请求构造程序将需求转换为一组分类型的 SQL 查询语句。

（2）检索请求执行程序将前一阶段构造的 SQL 查询语句在数据库中进行执行，得到一组反馈结果。另外，按照检索层次条件将分类型的 SQL 返回结果进行取交集与取差集处理。

（3）检索结果排序与展示程序将对同一层次中的 Activity 模式根据其与检索需求的接近程度进行排序。最后，从 NoSQL 数据库中读取各 Activity 源文件及相关资源文件，向用户展示。

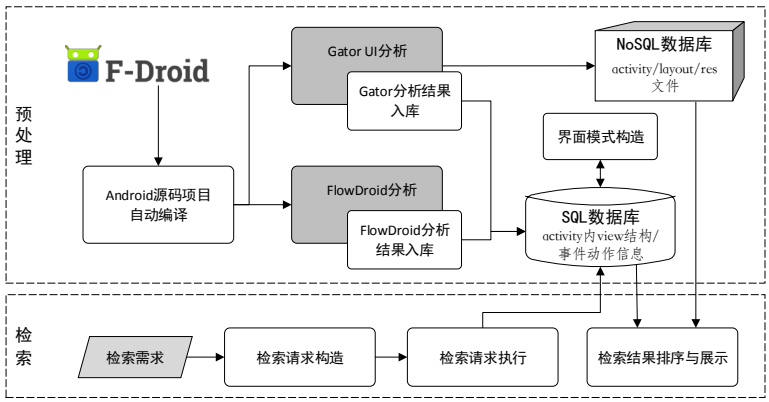

图 6. 工具链架构

在预处理阶段，通过工具链的执行，共分析来源于 F-Droid 的 252 个 App 项目，抽取得到 994 项 Activity 界面模式。其中包含的各项模式元素的数量信息见表 1。

表 1. 预处理分析得到的 Activity 界面模式元素数量

#View		# EventHandler	# EventAction	
# ViewGroup	# Widget		# WidgetInfluence	# IntentSwitch
43728	68288	47360	32969	14391

5 案例

本部分通过两个检索案例来验证本文所提出的方法及工具链。第二个案例通过自定义的基于 PQT 的查询，验证检索结果的有效性。

5.1 准确性验证案例

为验证本方法的准确性，计划一项控制性实验，针对已被抽取的某一个安卓 App 的 Activity 界面构造对应的基于 PQT 的查询，验证检索结果是否符合预期，即希望该 Activity 能够作为返回的界面模式中的第一排位。选择名为 GeoQuiz 的应用的 QuizActivity 活动构造其对应的查询 JSON 文本（由于篇幅原因在此不列出）。该需求表示用户需要查找由 5 个 widget 组成的界面，其中包括 2 个事件类型为 click 的按钮，1 个文本框，1 个事件类型为 click、动作类型为 widgetinfluence、影响方法为 setText 的按钮，以及 1 个事件类型为 click、动作类型为 intentswitch 的按钮。通过检索工具的使用，获得表 2 所列举的三个层次中的满足条件的 Activity 结果，并展示了每一个层次中 Layout 相似性度量的计算值。

表 2 准确性验证案例的检索结果

检索层次	Activity 名称	Layout 相似度	检索排序
第一层次	com.bignerdranch.android.geoquiz.QuizActivity	1	1
第二层次	com.iazasoft.footguy.Prefs	0.731	2
	org.yaaic.activity.MainActivity	0.731	3
第三层次	com.matteopacini.katana.KatanaActivity	0.669	4
	fr.renzo.wikipoff.ui.activities.DeleteDatabaseActivity	0.648	5
	com.onest8.onetimepad.MainActivity	0.601	6
	com.github.xloem.qrstream.Launcher	0.439	7
	com.github.wdkapps.fillup.GasRecordActivity	0.272	8

5.2 有效性验证案例

为验证方法的有效性，自定义界面模式检索的需求，并构造如图 7 所示的基于 PQT 的查询。该需求表示用户需要一个由 3 个菜单项组成的菜单界面，其中每个菜单项的事件类型为 item_selected，动作类型为 intentswitch。

```
query = {
        "viewType":"ViewGroup",
        "viewName":"android.view.Menu",
        "subViews":[
                {
                  "viewType":"Widget",
                  "viewName":"android.view.MenuItem",
                  "events":[
                          {
                            "eventType":"item_selected",
                            "enentAction":{
                                          "actionType":"intentSwitch",
                                          "targetWidget":"",
                                          "influenceMethod":""
                                          }
                          }
                         ]
                  },
                  ……  //另外两个菜单项与上一个 JSON 对象类似
                ]
        }
```

图 7 基于 PQT 的有效性验证案例查询需求

同样经过检索工具的运行，得到表 3 所列检索结果。根据检索机制，所有层次中的查询结果均符合 view 类型与数量的条件，因此在其界面中均包含具有三个菜单项的菜单控件。就 Widget 的事件行为而言，第一层次中的结果在菜单的设计与交互实现上更符合用户的期望需求，用户可直接复用 Widget 的资源定义与事件行

为代码。用户也可选择第二层次或第三层次中的查询结果，在知晓其事件行为并不符合预期的前提下仅复用其布局而忽略相关的行为实现。这种结果反馈方式提高了界面模式检索的有效性与可用性，且为用户提供了更加广泛的反馈建议。

表 3 有效性验证案例的检索结果

检索层次	Activity 名称	Layout 相似度	检索排序
第一层次	cx.hell.android.pdfview.ChooseFileActivity	0.926	1
	org.grapentin.apps.exceer.gui.MainActivity	0.892	2
	email.schaal.ocreader.ItemPagerActivity	0.841	3
第二层次	com.hayaisoftware.launcher.activities.SearchActivity	0.658	4
	com.bleyl.recurrence.activities.ViewActivity	0.419	5
	org.cprados.wificellmanager.ui.WifiPreferences	0.391	6
第三层次	com.luorrak.ouroboros.deepzoom.DeepZoomActivity	0.352	7
	ru.meefik.busybox.MainActivity	0.294	8
	org.xbmc.android.remote.presentation.activity.MusicArtistActivity	0.187	9
	jwtc.android.chess.PlayHubActivity	0.122	10

6 结论与展望

安卓开发者需要通过不断地学习来提高其设计 Activity 界面及其交互的能力。为了节省选择、使用 App 的时间且降低学习的成本，本文提出的基于布局与交互分析的安卓应用界面模式抽取与检索方法为开发者提供了一种有效的途径。该方法在预处理阶段分析单个 Activity 的界面元素构成及交互行为，在检索阶段支持用户通过查询语言搜索出与其需求最为接近的界面模式。

本文所提出的方法以及实现的工具链能够完成基本的安卓界面模式抽取与检索能力，是在针对界面的安卓代码推荐方面的初步尝试。然而，方法及工具本身仍然存在有待改进的地方，这些关注点可作为未来研究工作的重点。

首先，目前 GATOR 分析的结果仅指明了 View 的层次结构，但未涉及各个 View 本身的属性，例如一个 Button 的宽度、高度及位置等信息。借鉴 Sun 等人提出的 DroidEagle 工具[15]，我们认为更丰富的界面特征能够允许户构造更为细节的搜索请求，并得到更为精准的搜索结果。因此，下一步将尝试改进 GATOR 的分析策略，扩展界面模式元素的属性范围。

其次，在现阶段仍然需要依赖用户自行构造基于 PQT 的查询请求并且预先验证 JSON 格式的正确性，这为不熟悉界面结构的用户带来了困难。针对这一问题，计划在后续的工具研制中采用可视化的方式支持用户对查询需求进行表述，即允许用户通过拖拽控件、设置属性行为等所见即所得的方式将用户的布局设计自动转换为基于 PQT 的查询。

最后，用户当前能够查看的检索结果包括 Activity 的代码与资源文件，通过这些反馈暂时难以让用户获得对界面模式最直观的感受。因此，计划在后续工作中增强界面模式的展现效果。其中一种途径是通过可视化的方式（例如针对 Activity 的截屏或者行为的录像）让用户直接理解界面。为了达到这个目标，将尝试在 GATOR 分析结果上生成 Appium[16]脚本赋予其自动化跳转至该 Activity 界面并对其进行截屏的能力。

References:

[1] “Gartner Says Worldwide Sales of Smartphones Grew 9 Percent in First Quarter of 2017”. http://www.gartner.com/newsroom/id/3725117

[2] Junwei W, Liwei S, Wunan G, et al. Code Recommendation for Android Development: How does it Work and What can be Improved?[J]. SCIENCE CHINA Information Sciences.

[3] Thung F, Wang S, Lo D, et al. Automatic recommendation of API methods from feature requests[C]//Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering. IEEE Press, 2013: 290-300.

[4] Jiang H, Nie L, Sun Z, et al. Rosf: Leveraging information retrieval and supervised learning for recommending code snippets[J]. IEEE Transactions on Services Computing, 2016.

[5] Kumar R, Satyanarayan A, Torres C, et al. Webzeitgeist: design mining the web[C]//Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2013: 3083-3092.

[6] Deka B, Huang Z, Kumar R. ERICA: Interaction Mining Mobile Apps[C]//Proceedings of the 29th Annual Symposium on User Interface Software and Technology. ACM, 2016: 767-776.

[7] Nguyen T A, Csallner C. Reverse engineering mobile application user interfaces with remaui (t)[C]//Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on. IEEE, 2015: 248-259.

[8] Alharbi K, Yeh T. Collect, decompile, extract, stats, and diff: Mining design pattern changes in Android apps[C]//Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services. ACM, 2015: 515-524.

[9] Sahami Shirazi A, Henze N, Schmidt A, et al. Insights into layout patterns of mobile user interfaces by an automatic analysis of android apps[C]//Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems. ACM, 2013: 275-284.

[10] Rountev A, Yan D. Static reference analysis for GUI objects in Android software[C]//Proceedings of Annual IEEE/ACM International Symposium on Code Generation and Optimization. ACM, 2014: 143.

[11] Yang S, Yan D, Wu H, et al. Static control-flow analysis of user-driven callbacks in Android applications[C]//Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on. IEEE, 2015, 1: 89-99.

[12] Yang S, Zhang H, Wu H, et al. Static window transition graphs for android (t)[C]//Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on. IEEE, 2015: 658-668.

[13] Arzt S, Rasthofer S, Fritz C, et al. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps[J]. Acm Sigplan Notices, 2014, 49(6): 259-269.

[14] Klieber W, Flynn L, Bhosale A, et al. Android taint flow analysis for app sets[C]//Proceedings of the 3rd ACM SIGPLAN International Workshop on the State of the Art in Java Program Analysis. ACM, 2014: 1-6.

[15] Sun M, Li M, Lui J. DroidEagle: Seamless detection of visually similar Android apps[C]//Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks. ACM, 2015: 9.

[16] Appium. http://appium.io/

[17] Tekli J, Chbeir R, Yetongnon K. An overview on XML similarity: Background, current trends and future directions[J]. Computer science review, 2009, 3(3): 151-173.