基于日志挖掘的移动应用用户访问模型建模技术研究

陈三川[1,2,3] 吴国全[2] 魏 峻[2,3] 黄 涛[2,3]
(中国科学院大学 北京 100190)[1]
(中国科学院软件研究所软件工程技术研究开发中心 北京 100190)[2]
(中国科学院软件研究所计算机科学国家重点实验室 北京 100190)[3]

摘 要 提出了一种基于监控日志挖掘的移动应用用户访问模型自动构造方法,该方法包括监控代码注入和界面访问模型构造两部分。首先,提出了一种监控代码自动注入方法,即通过对移动应用代码的静态分析,自动地在相应位置插入监控代码以支持在运行时动态地监控用户的访问行为。其次,提出了一种基于状态机的移动应用用户访问模型构造方法。访问模型中状态机的节点和节点间跳转上的附加属性描述了UI界面之间的跳转行为和界面内控件的使用情况。对移动应用进行的实验表明,这种基于监控日志挖掘的移动应用用户访问模型自动构造方法能够成功地自动注入移动应用的监控代码,并能够有效获得移动应用用户界面访问行为。

关键词 移动应用,用户界面访问行为,界面跳转,自动注入
中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.11.006

User Behavior Modeling Method for Mobile Applications Based on Log Mining

CHEN San-chuan[1,2,3] WU Guo-quan[2] WEI Jun[2,3] HUANG Tao[2,3]
(University of Chinese Academy of Sciences, Beijing 100190, China)[1]
(Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[2]
(State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[3]

Abstract This paper presented a user behavior modeling method for mobile applications based on log mining. The method is two-fold, including monitoring instructions instrumentation and UI access modeling. We presented an automatic monitoring instructions instrumentation method that uses static analysis to automatically insert monitoring instructions at the appropriate site in order to dynamically monitor user behavior at run time. We also presented an automata based user behavior modeling method for mobile applications. Information attached to states and transitions of the automata in user behavior model describes transitions between UIs and the usage of each widget within UIs. The test results on real world mobile applications show that this method can both successfully instrument monitoring instructions and effectively obtain the UI access behaviors.

Keywords Mobile applications, User interface access behavior, UI transition graph, Instrumentation

1 引言

随着移动设备的大量普及,各种移动应用市场上出现了数以万计的移动应用。这些应用之间的竞争十分激烈。为了优化这些移动应用,使之更好地适应用户的需求,其开发者需要监控移动应用的行为。移动应用监控分析平台[1,2]可以收集用户数、会话长度、硬件类型、运营商等信息,分析用户忠诚度等信息。开发者可以根据分析得到的用户行为来提供更好的用户体验,获得更好的收益,保持一定数量的忠实用户。流行的移动应用监控分析平台包括:友盟[1]、Flurry[2]等。

目前,已有移动应用监控分析平台的主要不足有:

1)在监控日志的收集方面,已有移动应用监控分析平台为移动应用开发者提供移动端SDK,应用开发需要在应用源代码的相应位置手动注入监控代码来在运行时动态地收集用户访问信息,这增加了开发者的工作量,并且容易导致编码错误的发生。监控代码的自动注入将有效分离关注点,降低移动应用开发工作量,提高移动应用开发效率。

2)在用户行为分析方面,已有移动应用监控分析平台只支持简单的用户数量统计、会话时长统计等统计分析功能,缺少对移动应用复杂行为的分析功能。例如,应用开发者可能希望了解移动应用各种功能有怎样的访问路径,哪些功能被经常使用,哪些功能被较少使用。

为了克服已有工作的不足,我们提出了一种基于监控日志挖掘的移动应用用户访问模型自动构造方法,该方法包括监控代码注入和访问模型构造两部分。

首先,为了能够在运行时收集用户访问信息生成监控日

到稿日期:2013-09-16 返修日期:2013-11-16 本文受国家自然科学基金资助项目(61173005),国家自然科学基金资助项目(61003029)资助。
陈三川(1987-),男,硕士,主要研究方向为软件工程与分布式计算,E-mail:chenschuan@gmail.com;吴国全(1979-),男,博士,副研究员,主要研究方向为软件工程与分布式计算;魏 峻(1970-),男,博士后,研究员,主要研究方向为软件工程与分布式计算;黄 涛(1965-),男,博士,研究员,主要研究方向为软件工程与分布式计算。

志，我们设计了一种监控代码自动注入方法。通过对移动应用代码的静态分析，自动地在相应位置插入监控代码以支持在运行时记录、收集用户的访问行为。通过使用这一监控代码自动注入方法，开发者不用手动地注入监控代码，减轻了工作量，并且分离了关注点。

其次，基于监控日志信息，我们提出了一种基于状态机的移动应用用户访问模型构造方法。访问模型通过为状态机中的节点、节点间跳转附加各种属性等信息描述了 UI 界面之间的跳转关系、界面之间的跳转概率、跳转发生的触发条件和界面内控件的使用情况。根据这一访问模型，移动应用开发者可以了解移动应用的热点访问路径和移动应用所提供功能的使用情况等信息，进而对移动应用进行进一步优化。

本文第 2 节对已有相关工作进行分析和介绍；第 3 节介绍一种移动端监控代码的自动注入方法；第 4 节介绍基于界面跳转自动机的移动应用界面访问模型自动构造方法；第 5 节给出访问模型建模示例；第 6 节给出面向移动应用的监控平台设计；第 7 节对所提出的方法进行实验验证和分析；最后总结全文并给出下一步的研究工作。

2 相关工作

2.1 移动应用监控

文献[3]提出了一个移动应用性能监控工具，并指出移动应用异步、多线程的特性使监控变得困难。针对上述困难，文献[3]实现了 AppInsight，即向移动应用注入代码来自动确定用户事务中的关键路径和异常路径。AppInsight 记录异步调用和回调函数的开始时间、结束时间、调用位置、线程标识，从而把异步调用和回调函数对应起来，避免了记录执行系统代码的路径，有效降低了监控开销。用户使用注入的移动应用时，监控信息将被上传到服务器，这些信息会在分析后通过网页展示出来。

另一类相关研究是移动应用污点分析，文献[4]使用动态的污点分析来监控智能手机上的隐私数据泄露。文献[4]的污点分析提供了多种粒度的污点跟踪：变量级别、方法级别、消息级别、文件级别。该污点分析主要在虚拟机和进程间的通信库上实现，将包含隐私信息的信息源如通讯录作为污点源，可能导致把信息泄露的操作作为污点传播的终点。

2.2 移动端代码注入

文献[5]针对移动应用给出一种灰盒的 GUI 模型生成方法。文献[5]的工作分为两步：第一步，对移动应用的源代码进行静态分析，设计推理算法对应用 GUI 的行为进行推理，抽取出 GUI 中每个 UI 控件支持的行为；第二步，使用一个动态爬虫算法来系统地测试中移动应用的行为。第一步中的推理算法可以用来确定我们移动应用监控代码自动注入的位置。

文献[6]设计并实现了一个移动应用选择性代码注入框架 SIF(Selective Instrumentation Framework)。文献分析了最近需要定制化代码注入的相关研究，发现了这些注入工作对注入框架的要求：选择性、动态路径检查、高效性。文献拓展了 java 语言，提出了一种语言来提高描述的表现力，同时保证高效性，定义了代码点集合和路径集合以实现选择性代码注入和动态路径检查，同时使用静态分析和动态分析技术来估计注入代价。

2.3 界面访问模型

一类相关研究是 GUI 测试事件流图，文献[7]针对 GUI 测试中测试所有可能的事件组合和事件顺序会使测试用例数量呈指数增长的问题，提出了基于使用数据(usage profile)和 GUI 结构来自动构造事件流图(Event Flow Graph)，并使用事件流图指导测试用例生成。事件流图中的节点代表 GUI 的事件，事件 e1 和 e2 之间的边表示事件 e2 将紧接着事件 e1 之后被触发。事件流图中有一个公共的起点 INIT 和一个公共的终点 FINAL，这些从 INIT 节点到 FINAL 节点的路径构成了马尔科夫链，满足马尔科夫假设，进而计算每个节点的先验概率和后验概率并标识在节点上，构造得到概率事件流图，以此来刻画用户对 GUI 的实际使用。

另一类相关研究是活动调用图，文献[8]指出简单地运行移动应用而没有适当的 UI 界面交互不能暴露移动应用的一些安全问题，进而提出使用静态分析和动态分析结合的方法，构造应用的活动调用图(Activity Call Graph)和函数调用图(Function Call Graph)来找出移动应用中调用敏感的 API 的函数及活动调用路径，给出路径上相关的 UI 元素。文献[8]中应用的活动调用图中的节点是活动(Activity)，边是活动之间的跳转关系，节点和边没有权值，活动调用图是由静态分析得到的。

还有一类相关研究是电子商务网站的负载刻画，文献[9]指出传统的按照每秒点击数等指标刻画负载的方法不适合刻画电子商务网站的负载，电子商务网站的用户通过一系列的顺序相关的请求与网站交互，这些请求构成用户会话。因此，文献[9]提出一种刻画和产生电子商务网站负载模型的方法。首先，文献[9]定义了用户行为模型图(Customer Behavior Model Graph)来描述有相似访问模式的用户行为，构造负载模型并提出针对不同 CBMG 的聚类算法。文献[9]中的 CBMG 节点是如购买、选择、付款等用户的行为，边是这些行为的跳转关系，边上的权值表示跳转概率。

3 监控代码自动注入

为了能够在运行时收集用户访问信息生成监控日志，减轻开发者手动注入监控代码的负担，减少手动注入产生的编码错误，我们设计了一种监控代码自动注入方法。

这些监控代码分为两类：一类代码监控界面之间的跳转；另一类监控界面内 UI 控件的使用情况以及这些控件是否触发了界面之间的跳转。

一种直观的方法是静态分析移动应用代码，找到所有 UI 控件的监听器，在这些 UI 控件的监听器(listener)代码中注入对使用情况的监控代码；再分析所有这些监听器代码，找到这些代码对 intent 的使用，记录这些使用，进而注入监控触发界面跳转的代码。

这种做法的代码分析代价较大，我们采取以下方法：1)分析移动应用代码；2)在监听器中注入监控代码；3)设置监控变量；4)在监控界面的 OnPause 函数注入监控代码的方法。

这样做的理由是：移动应用的一个 UI 界面通常由一个类来实现，如安卓平台中一个 UI 界面由开发者继承 Activity 类来实现，这个类中包含很多重要的回调函数，如 OnPause 函数，一个 UI 界面在退出前台显示之前会执行这个函数。因此，我们在 Activity 类中设置一个监控变量来收集 UI 界面上的 UI 控件的使用情况，在执行 OnPause 函数的时候，表明 UI 界面即将跳转，我们获取所有监控变量中保存的 UI 控件使用情况，并将最后一个使用的 UI 控件作为触发 UI 界面的

控件上传至服务器端，以达到监控界面跳转和界面内控件的使用这两个目的。这种方法减少了代码静态分析的工作量，我们不必分析每一个监听器来查找是否有 intent 信息来导致 UI 界面的跳转，不必分析 intent 信息的参数含义，减少了注入代价。

具体的算法如算法 1 所示。首先，我们找出代码的入口点，找出所有表示界面的 Activity 类，分析代码的类层次，得到用户自定义的类并从中找出监听器类。其次，我们从每个入口点建立 CFG，找到监听器类的实例初始化的位置，沿 CFG 正向传播找到 CFG 上注册事件监听器的位置，记录注册的监听器和控件的对应关系。再次，根据这些对应关系，在对应的监听器处添加监控相应控件的代码。

算法 1 控件使用监控代码注入算法

```
Input S: app source code
Output S′: instrumented app source code
1. UIClass←getUIClass(S);
2. CH←getClassHierarchy(S);
3. HClass←getHandlerClass(CH);
4. EntryPoints←getEntryPoints();
5. FOREACH HC∈HClass
6.   FOREACH C∈UIClass
7.     FOREACH EP∈EntryPoints in UIClass
8.       CG←makeCallGraph(S,EP);
9.       l1←getHandlerClassInit(CG,HC);
10.      hInstance←recordHandler(l1,HC);
11.      l2←LocateRegistration(CG,HC);
12.      wInstance←recordwidget(l2);
13.      widgetAndHandler←recordPair(hInstance,wInstance);
14.    END
15.    FOREACH WH∈widgetAndHandler
16.      wInstance←getWidget(WH);
17.      hInstance←getHandler(WH);
18.      LF←getListenerFunctions(hInstance);
19.      instrumentAtHandlerClass(LF,WH);
20.      instrumentAtResumePause();
21.      instrumentMemberVariable();
22.    END
23.  END
24. END
```

4 移动应用用户访问模型构造

根据移动端收集到的大量的监控日志，移动应用监控分析平台挖掘这些监控日志信息，自动构建出用户的访问模型。我们的移动应用用户访问模型提供了丰富的信息，这些信息包括两部分：一部分是界面之间的跳转信息，包括界面之间的跳转关系、界面之间的跳转概率、跳转发生的触发条件；另一部分是界面内 UI 控件的使用信息。界面之间的跳转信息通过模型中表示界面的节点和节点之间跳转的权值及附件属性等信息给出；界面内 UI 控件的使用信息通过模型中节点上附加相应界面的 UI 控件的使用情况给出。

我们的访问模型自动构建方法首先将移动端收集到的大量的原始数据，即用户的事件序列转换为用户的界面跳转图，再对用户进行聚类并给出该组用户的访问行为。下面两小节将分别介绍我们提出的界面跳转图构造算法和用户聚类算法。

4.1 界面跳转图构造算法

界面跳转图构造算法将移动端收集到的大量的原始数据，即用户的事件序列转换为用户的界面跳转图。这一算法基于经典的 Ktail 算法[10]，Ktail 算法的基本思想如下：

Ktail 算法从事件序列中逐一读入事件，向前看 k 个事件（当前事件序列的长度为 k 的尾部），将具有相同的长度为 k 或者更短的尾部的前缀构造为一个等价类，这个等价类被映射成 FSM 中的一个状态，读入的事件使 FSM 在不同状态间转换。这样便由时间序列构造了一个描述事件间相互转换的自动机。

使用上述 Ktail 算法可以自动地由事件序列构造自动机，例如：对于如下收集到的事件序列 ABEBFBEBA，使用上述 Ktail 算法可以构建得到如图 1 所示的自动机。

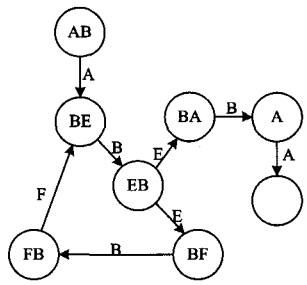

图 1 输入序列 1 生成的自动机

如果将使用上述 Ktail 算法得到的自动机作为移动应用的访问模型，这一访问模型的主要不足是没有考虑到移动应用的界面跳转特点，不能直观地反映移动应用界面之间的跳转。

为此，我们结合移动应用 GUI 的特点，对使用 Ktail 算法构造出的自动机进行进一步的优化和修正。首先将自动机转化为 UI 自动机，即将自动机的每个状态用 UI 界面来表示；之后，我们在 UI 自动机上添加相应的跳转概率得出界面跳转图。

具体的算法如算法 2 所示。第一步，我们首先对 UI 跳转事件上附加的相关信息进行处理，如保存这些信息中 UI 界面上控件的使用情况，之后根据 Ktail 算法构造自动机；第二步，我们针对上一步生成的自动机中的每一个节点，获得这个节点的出边和入边，将自动机转换为以界面为节点的自动机；第三步，计算事件序列中事件间转移的概率，在自动机的边上添加相应的跳转概率得出界面跳转图。

算法 2 界面跳转图构造算法

```
Inpute ESequences: set of event sequences
Output UITGraph: UI transition graph
1. #STEP 1. Construct automata
2. Automata←null;
3. pre←null;
4. FOREACH esequence ∈Esequences
5.   FOREACH event ∈esequence
6.     profile[event]←getProfile(event);
7.     K←lookAheadK(event);
8.     s←newState(K);
9.     IF s equals s′ ∈Automata
10.      connect(pre,s′);
11.      pre←s′;
12.    END
13.    ELSE
14.      connect(pre,s);
15.      pre←s;
16.      END
```

```
17.   END
18. END
19. #STEP 2. Transfer automata into UI Automata
20. UIAutomata←null
21. FOREACH s ∈ Automata
22.   inEdges←getInEdges(s);
23.   outEdges←getOutEdges(s);
24.   FOREACH inEdge∈inEdges
25.     A←newUIState(inEdge);
26.     FOREACH outEdge∈outEdges
27.       B←newUIState(outEdge);
28.       connect(A,B);
29.     END
30.   END
31. END
32. #STEP 3. Transfer UI Automata into UI transition graph
33. UIGraph←null;
34. FOREACH edge(e_i,e_j) ∈UIAutomata
35.   n_i←count(e_i,e_j);
36.   FOREACH count(e_i,e_k) ∈UIAutomata
37.     n_k←count(e_i,e_k);
38.   END
39.   P←n_i/(∑n_k);
40.   addWeight(i,j,P);
41. END
```

4.2 用户聚类算法

用户的界面跳转图给出了用户访问的移动应用界面之间的跳转关系和用户在这些界面之间的跳转概率，它刻画了这个用户的界面访问行为。我们通过对用户进行聚类将类似行为的用户归为一类，从而对用户进行精确的分组并给出该类用户的行为，从而指导移动应用开发者优化移动应用。

用户聚类算法如算法 3 所示。从表示应用入口的节点开始记录长度为 k'的路径 s'，这个路径每个节点都是上一个节点跳转概率最大的节点，路径 s'即为该界面跳转图的热点路径。创建该跳转图的热点路径等价类，将该跳转图放入这个等价类。将用户的事件序列加入所在用户分类的事件序列，得出该用户分类的事件序列。对这些事件序列，使用界面跳转图构造算法得出该用户分类界面跳转图。

算法 3 用户聚类算法

```
Input UIGraphs: n UI graphs for n users
Output: m user groups,their hot paths,and UI graph for each group
1. FOREACH UIGraph∈UIGraphs
2.   hotpath←computeHotPath(UIGraph)
3.   H←addToSet(hotpath)
4.   equivClass
5.   ←computeEquivalenceClass(hotpath)
6.   E←addToSet(equivClass)
7. END
8. FOREACH equivClass∈ E
9.   FOREACH user ∈equivClass
10.     S←addUserEventSquences(user)
11.   END
12.   GroupGraph←computeUIGraphFor(S)
13. END
```

5 访问模型建模示例

下面以一个实际的移动应用为例，说明我们的界面跳转图构造算法和用户聚类算法。

移动应用 MyLife 是一个生活应用，提供旅行、地图、同城等功能，旅行功能有机票预订和酒店预订子功能，地图功能有地点定位和公交查询子功能，同城功能有租房信息和交友平台子功能。应用的主要界面有 10 个，分别为主界面、旅行界面、地图界面、同城界面等，如表 1 所列，下文中将分别用大写英文字母 A、B、C 等表示。

表 1 生活应用界面功能示意

界面	Activity A	Activity B	Activity C	Activity D	Activity E
功能	主界面	旅行	地图	同城	机票
界面	Activity F	Activity G	Activity H	Activity I	Activity J
功能	酒店	地点	公交	租房	交友

输入序列 1：ABEBFBEBA

输入序列 2：ACHCHCHCADIDA

输入序列 3：ABEBA

取算法中的 $K=2$，则自动机中节点可用两个事件的时间序列表示。

序列 1 生成的自动机如图 1 所示。

序列 1、2、3 生成的界面跳转图分别如图 2—图 4 所示。

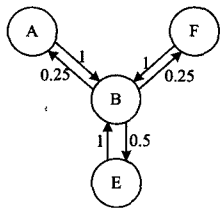

图 2 输入序列 1 生成的界面跳转图

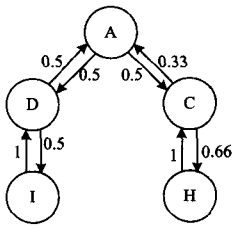

图 3 输入序列 2 生成的界面跳转图

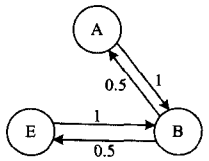

图 4 输入序列 3 生成的界面跳转图

k'取 3 时，序列 1、2、3 的热点路径分别是 ABE、ACH、ABE。我们按照聚类算法对序列 1 和序列 3 的用户进行聚类得出该用户组的界面跳转图，如图 5 所示。

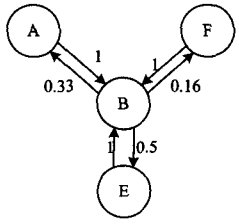

图 5 输入序列 1、3 所在用户组的界面跳转图

由上面的用户聚类算法结果可知，用户 A 和用户 C 属于

同一用户组，具有相似的使用行为，他们大多使用该应用 3 大功能中的旅行功能，这个监控结果为开发者提供了关于用户的有意义的参考。

我们的用户访问模型还包括界面之间跳转的触发条件（用触发跳转的 UI 控件表示）和界面 UI 控件的使用情况，这些信息都作为界面跳转图中节点的附加属性给出，本文对这些信息进行了简化，没有在上述图中标出。

6 监控平台的设计与实现

6.1 监控平台架构

我们设计并实现了一个面向移动应用的监控分析平台，其系统架构如图 6 所示。可以看出监控平台的系统架构涉及手机端、服务器端、浏览器端 3 部分。手机端将嵌入到应用程序的 SDK 中作为通信渠道，收集数据并上传到服务器。服务器端监听请求，进行相应的处理并写入数据库，在收到浏览器端请求时读取数据库并发送。浏览器端请求并得到服务器端数据，分析监控数据，将其显示在仪表盘上。其中手机端主要包括用户信息收集模块、手机信息收集模块、界面访问信息收集模块；服务器端包括路由控制模块、信息处理模块、数据库模块；浏览器端包括分析显示模块等。

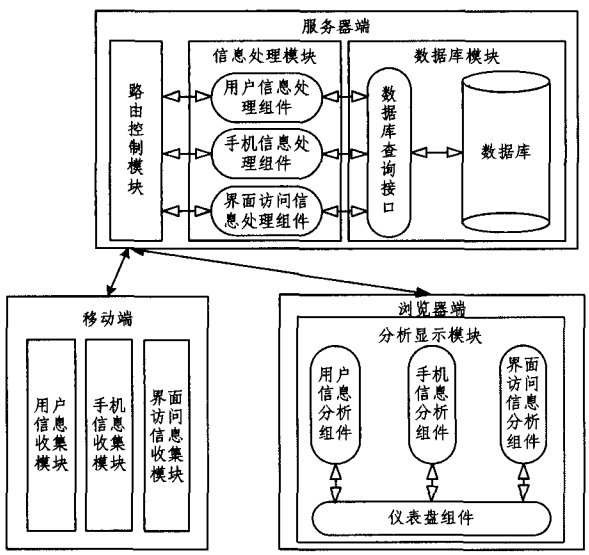

图 6 监控平台系统架构

6.2 移动端

移动端由用户信息收集模块、手机信息收集模块和界面访问信息收集模块组成。用户信息收集模块负责收集用户登录、会话时间等信息；手机信息收集模块负责收集手机终端信息，包括设备型号、设备操作系统、移动运营商等信息；界面访问信息收集模块负责收集界面跳转的时间、跳转来源界面和跳转目的界面、界面内控件使用情况等信息。应用开发者使用监控代码自动注入工具注入相应的监控代码，这些代码实现用户信息收集模块、手机信息收集模块和界面访问信息收集模块的功能。

如第 3 节所述，为了实现移动应用监控代码自动注入，我们静态分析移动应用的代码，设计分析算法，获得需要注入监控代码的位置，并注入相应的语句，自动注入工具的工作流如图 7 所示。

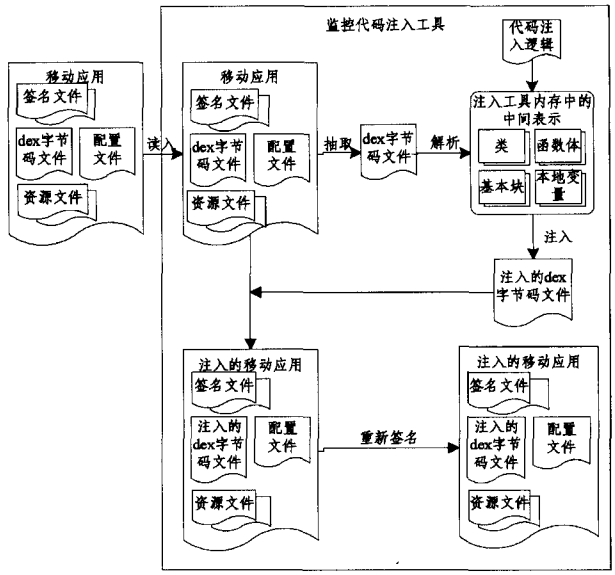

图 7 自动注入工具的工作流

6.3 服务器端

服务器端由路由控制模块、信息处理模块和数据库模块组成。服务器端提供监控数据的存储、查询、初步处理功能。服务器端提供两类 API：输入 API 和输出 API。

路由控制模块监听手机端和浏览器端的请求，接收发来的信息并对信息进行解析。如果请求实现输入 API，根据解析的内容发给信息处理模块的相应组件进行处理，处理之后的数据交给数据库模块存入数据库。如果请求实现输出 API，根据解析的内容发给信息处理模块的相应组件进行处理，处理之后的数据交给数据库模块读取数据库，将结果返回发起请求的手机端或浏览器端。

6.4 浏览器端

浏览器端由分析显示模块组成。分析显示模块向服务器端请求并接收监控信息，对监控信息进行分析，并将其显示在仪表盘界面。分析显示模块主要包括用户信息分析组件、手机信息分析组件、界面跳转信息分析组件、仪表盘组件。

分析显示模块各组件之间的交互可以通过下面的用例来说明。

应用开发者登录仪表盘，查看分析结果，仪表盘组件会调用相关分析组件。相关分析组件从数据库中获得信息并使用相关的业务逻辑进行分析，分析结果交给仪表盘组件进行进一步处理。仪表盘组件得到分析数据后以丰富的图表（饼状图、柱状图等）将其显示在仪表盘。

7 案例验证

根据上述移动应用监控分析自动构建方法，我们选择阅读应用 ZAKER 和在线音乐应用 Jamendo 进行实验来验证我们的方法。

ZAKER 是一款互动分享和个性化定制的阅读软件。它将微博等内容，按照用户个人意愿聚合起来，用户可以个性化定制阅读。Jamendo 是一款在线音乐软件，Jamendo 有音乐分类列表、搜索、下载、在线播放等功能。

首先，我们对 ZAKER 和 Jamendo 进行监控代码自动注入，实验环境为 PC：Dell Optiplex990，CPU：intel 8-core i7 3.40 Hz，RAM：4GB，OS：windows 32bit，注入实验结果如表 2 所列。

表 2　注入实验结果

应用名称	注入情况	用时(s)
ZAKER	成功	21
Jamendo	成功	19

注入后监控代码对移动应用性能的影响如表 3 所列。由实验可知，注入后的性能下降可以接受，不会导致用户体验不佳进而放弃使用。

表 3　注入后性能损耗

应用名称	注入情况	平均性能下降(%)
ZAKER	成功	2.1
Jamendo	成功	1.5

其次，我们将注入了监控代码的移动应用分发给少量用户(10 人)并收集监控信息建立用户访问模型。这两个应用的使用情况如表 4 所列。

表 4　应用使用情况

应用名称	用户数	总会话数
ZAKER	10	221
Jamendo	10	233

如表 5 所列，针对应用内控件的使用，我们发现在 10 个 ZAKER 用户的 221 次会话中只有 3 个用户的 10 次会话使用了 ZAKER 提供的添加资讯功能，该控件位于主界面的右下角，图标较小。Jamendo 使用较少的控件是播放界面下方的拖拽菜单。这些控件都是用户可能因不熟悉等原因而没有注意的控件。

表 5　控件使用情况

应用名称	用户数	最少使用的控件	使用次数
ZAKER	10	AddPrefButton	10
Jamendo	10	SlidingMenu	13

如表 6 所列，针对用户的访问行为，我们发现 ZAKER 的 10 个用户中有 6 个用户经常在浏览新闻之后评论这些新闻，Jamendo 的 10 位用户中有 7 位经常除了在线音乐之外选择收听在线电台。

表 6　用户访问行为

应用名称	用户数	热点路径
ZAKER	10	Main→Radio→Play
Jamendo	10	Main→News→Comment

这些实验结果给我们一个重要参考，即 ZAKER 和 Jamendo 的某些控件较小，不易辨识；ZAKER 和 Jamendo 用户的某些行为也可以指导开发者提供更好的用户体验，获得更好的收益，保持一定数量的忠实用户，如向经常收听在线电台的用户推荐各种电台软件。

结束语　本文提出了一种基于监控日志挖掘的移动应用用户访问模型自动构造方法，该方法包括监控代码注入和界面访问模型构造两部分。首先，我们设计了一种监控代码自动注入方法，即通过对移动应用代码的静态分析，自动地在相应位置插入监控代码来支持在运行时动态地监控用户的访问行为。其次，我们提出了一种基于状态机的移动应用用户访问模型构造方法。访问模型中状态机的节点和节点间跳转上的附加属性描述了 UI 界面之间的跳转行为和界面内控件的使用情况。通过对两种移动应用进行的实验表明，这种基于监控日志挖掘的移动应用用户访问模型自动构造方法能够成功地注入移动应用的监控代码，并能够有效地获得移动应用用户界面访问行为，包括界面间跳转热点路径和界面内控件使用情况。下一步将就移动应用监控分析工具的功能完善、性能优化、分析算法的优化、用户的精确细化分组等方面展开工作。

参 考 文 献

[1] 友盟统计分析平台[OL]. http://www.umeng.com/analytics

[2] Flurry Analytics[OL]. http://www.flurry.com/flurry-analytics.html

[3] Ravindranath L, Padhye J, Agarwal S, et al. AppInsight: Mobile App Performance Monitoring in the Wild[C]// Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation. Hollywood, USA, 2012

[4] Enck W, PGilbert W, Chun B-G, et al. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones[C]// Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation. Vancouver, Canada, 2010

[5] Yang Wei, Prasad M R, Xie Tao. A Grey-Box Approach for Automated GUI-Model Generation of Mobile Applications[C]// Proceedings of the 16th International Conference on Fundamental Approaches to Software Engineering. Rome, Italy, 2013

[6] Hao Shuai, Li Ding, Halfond W G J, et al. SIF: A Selective Instrumentation Framework for Mobile Applications[C]// Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services. Taipei, Taiwan, 2013

[7] Brooks P A, Memon A M. Automated GUI Testing Guided By Usage Profiles[C]// Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering. Atlanta, USA, 2007

[8] Zheng Cong, Zhu Shi-xiong, Dai Shuai-fu, et al. SmartDroid: an Automatic System for Revealing UI-based Trigger Conditions in Android Applications[C]// Proceedings of 2nd Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices. Raleigh, USA, 2012

[9] Menascé D A, Almeida V A F, Fonseca R, et al. A Methodology for Workload Characterization of E-commerce Sites[C]// Proceedings of the First ACM Conference on Electronic Commerce. Denver, USA, 1999

[10] Cook J E, Wolf A L. Discovering Models of Software Processes from Event-Based Data[J]. ACM Transactions on Software Engineering and Methodology, 1998, 7(3): 215-249

[11] Biermann A W, Feldman J A. On the Synthesis of Finite-State Machines from Samples of Their Behavior[J]. IEEE Transactions on Computers, 1972, 21(6): 592-597

[12] Gomez L, Neamtiu I, Azim T, et al. RERAN: timing- and touch-sensitive record and replay for Android[C]// Proceedings of the 35th International Conference on Software Engineering. San Francisco, USA, 2013

[13] Yang Li, Lin Zuo, Jun Wei, et al. Sequential Pattern-Based Cache Replacement in Servlet Container[C]// Proceedings of the 7th international conference on Web engineering. Como, Italy, 2007

[14] 黄翔，王伟，张文博，等. 面向性能剖析的 Web 应用自动性能建模方法[J]. 软件学报，2012，23(4)：786-801

[15] 张怡阳. 基于依赖注入的移动终端应用开发[J]. 计算机应用，2009，29(6)