一种基于 APP 无缝集成的应用流自动协同框架

陈世展[1),2)] 王 茹[1),2)] 冯志勇[1),2)] 薛 霄[1),2)] 何 强[3)] 林美辰[1),2)]

[1)](天津市认知计算与应用重点实验室 天津 300350)

[2)](天津大学智能与计算学部 天津 300350)

[3)](斯威本科技大学电子信息及软件工程学院 墨尔本 3122 澳大利亚)

摘 要 应用市场中存在着海量的移动应用(mobile Application,App),无论是在个人日常生活还是在专业领域都发挥着非常重要的作用.每个移动应用都是由不同的提供者独立开发的,提供特定的服务,在设计之初并没有考虑与其他移动应用之间的协同,这种彼此隔绝使其呈现出碎片化特征.面对日渐复杂的用户需求,这种碎片化特征导致用户在多个移动应用间频繁切换才能完成特定任务,用户体验较差.本文重点关注异构自治移动应用间以及移动应用与用户间的协同交互,提出一个基于 Android 的移动应用自动协同框架,允许在相关的移动应用之间信息共享与复用,并按照应用流的方式逐一执行,从而实现无缝连接满足复杂用户需求.该框架的关键技术包括三个部分:(1)将移动应用的内部数据表示为〈存储,语义,语法〉三元组,并利用静态分析和动态分析技术生成移动应用抽象服务化模型,为异构应用间数据复用奠定基础;(2)使用 Nowcasting 和 Forecasting 结合的移动应用预测方法及客户定制方法来构建应用流(mobile Application Flow,App Flow);(3)基于参数注入和 Android 进程间通信机制设计实现了移动应用间自动协同按需跳转执行引擎.在 Android 7.1 系统上实现了框架原型,并针对旅行外出典型应用场景展示了异构自治移动应用自动协同的效果.

关键词 移动应用流;自动协同;移动应用推荐;用户定制;Android

中图法分类号 TP391 **DOI 号** 10.11897/SP.J.1016.2019.02631

An Automatic Collaboration Framework for Integrating Mobile Apps into Flow Seamlessly

CHEN Shi-Zhan[1),2)] WANG Ru[1),2)] FENG Zhi-Yong[1),2)] XUE Xiao[1),2)] HE Qiang[3)] LIN Mei-Chen[1),2)]

[1)](*Tianjin Key Laboratory of Cognitive Computing and Application*, *Tianjin* 300350)

[2)](*College of Intelligence and Computing*, *Tianjin University*, *Tianjin* 300350)

[3)](*School of Software and Electrical Engineering*, *Swinburne University of Technology*, *Melbourne* 3122, *Australia*)

Abstract There are a large number of mobile applications (App for short) in the application market, which play a very important role in the daily life of individuals and in the professional application domains. Each App is independently designed and developed by a different provider, providing a specific service that was not designed to work with other Apps at the beginning. This isolation phenomenon from each other heterogeneous Apps leads to a fragmented feature of Apps. Faced with increasing the complex requirements of users, this fragmentation feature drives users to switch frequently and manually between multiple Apps to complete a specific task, resulting in a poor user experience. In this paper, we focus on the collaborative interaction between heterogeneous autonomous Apps and between Apps and users. In order to meet the complex requirements of

收稿日期:2018-05-11;在线出版日期:2019-04-03.本课题得到国家自然科学基金(61572350)、国家重点研发计划(2017YFB14012001)资助.**陈世展**,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为服务计算、面向服务的体系架构.E-mail:shizhan@tju.edu.cn.**王 茹**,硕士研究生,主要研究方向为服务计算.**冯志勇**,博士,教授,中国计算机学会(CCF)会员,主要研究领域为知识工程、服务计算、计算机认知.**薛 霄**(通信作者),博士,教授,中国计算机学会(CCF)会员,主要研究领域为服务计算、计算实验.E-mail:jzxuexiao@tju.edu.cn.**何 强**,博士,高级讲师,主要研究方向为软件工程、云计算、服务计算和大数据分析.**林美辰**,女,1992 年生,硕士,主要研究方向为服务计算.

users, it proposes an Android-based Apps Collaborative Framework to achieve the seamless connection of Apps, which allows information sharing between related Apps and executes them one by one according to the application flow automatically. The key technology of the framework consists of three parts: (1) Representing the internal data of the heterogeneous App as a set of 〈storage, semantics, grammar〉 triple, and using the Data-driven static program analysis and dynamic program analysis techniques to generate the abstract service model of Apps; (2) Using the App prediction method combined with Nowcasting and Forecasting and user customized profile (UCP) to build a mobile application flow(App Flow); (3) Designing the automatic collaborative execution engine based on parameter injection and the inter-process communication mechanism in Android to realize the on-demand switch between Apps. The advantages of this framework are mainly reflected in two aspects: (1) Mobile application prediction and recommendation based on application flow avoids the shortcomings of user-defined process, and human-computer interaction based on feedback and inductive learning further improves the accuracy of recommendation results; (2) Mobile application flow execution based on UCP avoids the complex calculation required for application prediction while preserving the flexibility and personalized features of mobile computing, and the sharing of the application process configuration profile among different users can overcome the cold start issue caused by missing user history to some extent. The framework prototype is built on the Android 7.1 system. Based on the prototype, the typical application scenario of travel out is used as a case study to demonstrate the automatic collaboration of the heterogeneous autonomous Apps. The results show that our App Collaborative Framework is effective and promising. In short, the framework proposed and constructed in this paper can effectively break the information barrier between different mobile applications, and greatly improve the user experience in the face of increasingly complexity of user requirements. In the future, we will focus on the following works: (1) The abstract model needs to be further enriched, distinguishing front-end UI, back-end cloud services, and third-party cloud services to be compatible with more interaction requirements and complex business processes; (2) The potential security risks in the implementation process flow will be analyzed and optimized to improve the security and robustness of application flow collaboration.

Keywords app flow; automatic collaboration; app recommendation; user customization; Android

1 引 言

近年来移动智能设备已经成为便捷的互联网入口,甚至处理以前只能通过 PC 完成的复杂任务.海量的移动应用(Mobile Applications,App)为这一繁荣做出了巨大贡献,无论是在个人日常生活还是在专业领域都发挥着非常重要的作用.通常移动应用都是由不同的提供者独立开发的,提供特定的服务,在设计之初并没有考虑与其他移动应用之间的协同,这种彼此隔绝使其呈现出碎片化特征.然而,面对日渐复杂、单一应用难以满足的用户需求,这种碎片化的状况导致终端用户不得不在多个相关移动应用间频繁切换、有序使用才能完成特定任务[1].

协同、组合或混搭已成为移动应用使用的一种趋势.以旅行外出场景为例,当用户做计划时会使用 Kayak① 查询航班信息,通过 Airbnb② 预订住宿,并将这些信息记录在备忘录③或日程安排④中.用户不仅需要找到合适的应用并各自启动它们,而且还要多次输入服务参数.例如,在 Kayak 中查询航班时输入目的地和出发日期,在 Airbnb 和备忘录中同样需要输入这些重复的参数.总之,多次在主屏幕点击启动不同的移动应用并重复输入这种以用户为中介的协同方式操作繁琐、使用体验不佳.

为改善用户体验、实现面向复杂需求的移动应

① https://www.kayak.sg/mobile
② https://zh.airbnb.com/mobile
③ http://app.mi.com/details?id=com.zhanlang.notes
④ http://www.zhwnl.cn

用自动协同，需要克服如下两个障碍：(1) 如何找到下一个合适的移动应用？在当前的使用模式下，用户的移动设备中安装了许多应用，当需要访问特定应用时需通过滑动屏幕寻找；(2) 如何实现不同移动应用间的协同？组合使用移动应用时，通常是由具有语义相关性的输入进行驱动的，但由于移动应用异构自治、彼此孤立，无法重用服务参数，需要用户重复手工录入.

使用上下文感知、挖掘用户历史使用行为的方法进行移动应用预测和推荐[2-3]，能够帮助用户找到合适的移动应用. 基于历史数据建模的 Forecasting 方式，忽视了移动计算所呈现的高动态性以及移动应用间的潜在关联关系、应用使用的流特征等，难以获得精确的预测结果. 更为主要的是，这些移动应用推荐在获取建议结果后即终止，需要用户自己手动进行后续操作. Rukzio 等人[4]证明对于移动用户来说，检查预先输入的参数比重新输入快大约四倍. 用户自定义配置文件(User Customized Profile，UCP)支持移动应用流(mobile application Flow，app Flow)定制，例如 Tasker①、IFTTT②、Microsoft Flow③等，在解决移动应用协同问题上进行了很好的探索，但由于语义歧义和异构问题使得所支持的移动应用数量上极其有限，而且需要用户事先编辑并保存应用流，使用门槛较高，对普通用户造成一定的障碍.

为了解决上述挑战，本文提出一个基于 Android 平台的移动应用自动协同框架，允许在相关的移动应用之间信息共享与复用，按照应用流的方式逐一执行，从而实现无缝连接满足复杂用户需求. 该框架的关键技术包括三个部分：(1) 将移动应用表示为〈存储，语义，语法〉三元组 item 的集合，并利用静态分析和动态分析技术生成移动应用抽象服务化封装模型，降低移动应用间异构性和语义歧义；(2) 整合移动应用预测推荐机制和客户定制构建个性化应用流. 基于 Nowcasting[5] 与 Forecasting 融合预测模型推荐下一个移动应用；UCP 则提供了应用流进一步复用和个性化编辑定制工具；(3) 针对异构自治移动应用的协同执行，基于参数注入和 Android 进程间通信(Inter-Process Communication，IPC)机制设计移动应用流执行引擎.

本文第 2 节对相关工作进行概述；第 3 节描述本文的自动协同框架；第 4 节详细介绍移动应用抽象模型、移动应用预测、批处理式执行关键技术及实现；第 5 节和第 6 节分别是移动应用推荐的实验结果分析及通过案例研究展示本文方法的有效性；接下来是讨论和全文总结.

2　相关工作

本节对应用流构建和多移动应用协同问题的相关工作进行回顾. 其中应用流构建包括移动应用预测推荐和 UCP，多移动应用协同主要涉及应用交互和移动服务组合方面的研究.

针对移动应用预测问题，Liao 等人[2]通过挖掘手机中移动应用的历史日志，构建移动应用及其使用时刻间的关系分布，进而计算其使用概率. Sun 等人[6]基于 MRU(Most Recently Used)和 MFU(Most Frequently Used)计算移动应用的使用概率，实现了 AppRush——对手机主屏幕上的移动应用快捷方式进行动态更改的自适应接口. Lee 等人[3]将使用日志抽象为包含时间、地点、行为、移动应用四个属性的 item，从而根据 item 间的属性相似度推断与用户上下文最接近的移动应用. Shin 等人[7]对收集的 37 种传感器数据进行全面分析，采用朴素贝叶斯方法结合用户上下文预测用户当前可能使用的移动应用. Lu 等人[8]综合用户的实际物理位置移动路径与虚拟移动应用使用路径之间的关联，提出了一种基于位置(Location Based Services，LSB)的移动应用推荐方法，根据用户实时物理位置推荐可用 App. Liao 等人[9]将显式特征和隐式特征结合，对移动应用使用情况进行预测，并使用特征选择算法进一步减小日志大小、缩短预测时间. Baeza-Yates 等人[10]提出了一种将基本特征与会话特征结合的方法，采用并行树朴素贝叶斯网(Parallel Tree Augmented Naive Bayesian Network，PTAN)来提升手机主屏移动应用的使用体验. 现有研究虽然从不同角度实现了移动应用预测，但忽略了对服务参数的动态性、周期性、可复用性以及移动应用间的语义关联等综合考虑，推荐特征仍可以进一步扩展.

UCP 方式属于工作流的静态编排，典型的代表有 Tasker、IFTTT 和 Microsoft Flow. Tasker 是一个基于上下文的应用，其中任务由动作、时间、位置等上下文信息组成，但是其操作所涉及的动作仅限于手机设置和系统类移动应用. IFTTT 为用户提供了菜单式的频道构建，分为触发频道和动作频道. 当触发频道的移动应用被触发后，IFTTT 会自动执行动作频道的相关动作. 与 Tasker 相比，IFTTT 支持更多的第三方应用. Microsoft Flow 是一个实现工

① https://tasker.joaoapps.com/
② https://ifttt.com/
③ https://flow.microsoft.com/zh-cn/

作流自动化的 SaaS 服务，与 IFTTT 在概念和设计上都很相似，能够快速发布到 Web 及移动设备上. 现有的对 UCP 的研究和应用，能够参与其中的移动应用是非常有限的，难以满足用户日趋复杂的个性化需求；另外基于配置文件的静态流程编排缺乏动态适应需求的能力，并且编辑配置文件也需要用户具有一定的动手能力.

访问开放式 API 或资源是解决应用交互问题的一种方案. MashDroid[11] 是一个数据驱动的、基于内容的移动移动应用组合方法，基于豌豆荚①移动应用内搜索(Inapp Search，IAS)机制，根据事先要求的形式对其内部资源进行描述，并允许豌豆荚从外部调用，从而打破移动应用间的界限. 接入 IAS 后，用户可以在豌豆荚内进行搜索，直达目标内容而不需要进入某个移动应用内再进行搜索. Project Flow② 项目试图进行移动应用的去壳化交互并将其连接到一个流中使用. 例如四个场景：搜索、音乐、导航、打车，但其本身是基于对外部开放资源或 API 的访问，不涉及第三方移动应用. 基于 AEM (Architectural Event Modules) 的多应用模块化组合[12] 系统支持系统级应用间的组合与协同. 如果集成了过多的资源或 API，臃肿的移动应用不仅启动速度慢，其运行效率也必然会受到影响，但是参数跨应用交互为应用自动协同提供了借鉴与指导.

在移动服务组合方面，Deng 等人[13] 从能耗角度切入，提出移动服务选择与组合的能量消耗聚合规则，并采用遗传算法和再规划机制处理用户行为和可变条件，使得移动服务组合能够在满足用户需求的同时保持低能耗. 因为移动端的各项性能受限，Deng 等人[14] 进一步提出一种基于移动服务共享社区架构、基于 KH(Krill Herd)算法进行服务组合的方法，从而在物理位置上实现社区的服务共享，为移动服务组合提供了更加多样化的选择. 该工作在移动服务组合方面做出了突出贡献，但在面临用户的某些复杂交互需求时尚未能提供一个最优方案. 针对旅行场景 Marwa 等人[15] 设计了一个适用于移动环境的旅行服务配置工具，用户只需在可视化 UI 中给定计划约束，该工具将比较各个服务网站所提供的服务，并将数据和服务通过 RESTful 接口进行混搭，结合用户偏好给出旅行计划. 虽然该工作与本文使用的案例场景类似，但它只针对 Web 应用，并不适用于大量第三方移动应用程序，也较少考虑用户的行为特征.

针对云计算中 Web 服务资源的“信息孤岛”问题，文献[16]对来自不同类型设备的大规模流数据进行服务化，进而实现数据的集成与处理，提升数据共享能力；文献[17]则从软件服务工程的角度，探讨了以 RGPS 需求元描述为基础的模型交互驱动架构 MiDA，实现多样化涉众需求制导的业务协作以及面向软件模型资源的服务供给侧可互操作构造.

3 自动协同框架

图 1 是本文移动应用自动协同框架示意图，包括分析、监控、预测、交互、UCP 及 App 知识库六个组件模块.

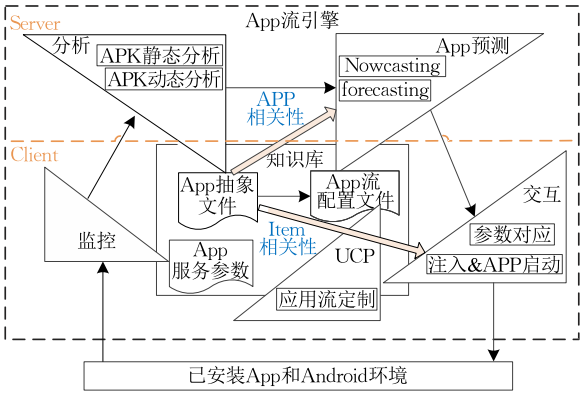


图 1 移动应用自动协同框架示意图

3.1 应用流分析

在服务器端，应用分析将实现异构移动应用的抽象封装，将移动应用表示为用户输入 item 的集合，其结构化图示如图 2 所示.

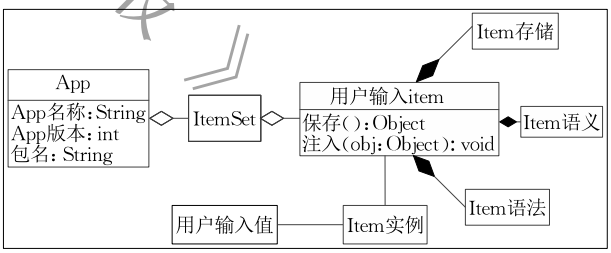


图 2 移动应用抽象模型的结构化图形表示

每个用户输入 item 表示为〈存储，语义，语法〉三元组形式，其中：

(1) Item 存储表示服务参数的物理存储位置；

(2) Item 语义是服务参数的含义；

(3) Item 语法是服务参数在移动应用中定义的数据格式.

每个 item 支持两种操作：保存和注入. 具体如下：

(1) 保存操作将服务参数进行存储以便在后续

① https://www.wandoujia.com/

② https://www.raventech.com/

移动应用中重用；

(2) 注入操作使用可重用的服务参数实现 item 实例的更新.

Item 实例驱动移动应用运行并与用户输入值产生关联，用户输入值则表示一个具有实际意义的词语. 然而，item 实例与用户输入值在表达上可能不完全相同，其中用户输入值是为了便于用户理解的，而 item 实例是方便移动应用后台处理的.

应用分析主要是关于应用程序包 APK 的静态分析和动态分析. 借助静态分析技术，分析移动应用的内部参数，实现从 APK 到移动应用抽象模型的映射. 由于静态分析得到的候选 Activity 集合较大，利用动态分析技术过滤部分 Activity. 考虑到程序自动化分析结果可能存在一定的偏差，本文采用了人-机结合的确认方式：以 Web 表单形式呈现自动解析得到的 App 抽象模型信息，供开发人员修正及确认，最终同步至知识库存储备用.

3.2　应用流构建

本文基于移动应用自动预测和用户定制(UCP)相结合的方式构建应用流. 除考虑上下文特征及历史数据外，还将移动应用相互间的关联关系(如相互配合完成更大粒度用户任务的协作关系、功能相似而导致的竞争关系等)以及应用流当前状态(不仅仅是当前应用，还包括了前序多个应用的信息)作为预测的输入特征，借鉴文献[5]中的 Nowcasting 气象学预测模型，实现 App 推荐进而自动构建应用流；与应用抽象模型处理的策略类似，用户也可以显式地对构建的应用流进行编辑和个性化定制，最终以 UCP 形式保存应用流.

3.2.1　应用流的预测

应用流预测可以被描述为综合当前上下文环境与应用流状态$\{App_0, App_1, \cdots, App_{i-1}\}$，综合 Nowcasting 与 Forecasting 模型预测用户当前时刻的期望 App_i.

移动计算具有高度的动态性，会随用户地理位置及使用时间发生变化，单纯基于历史数据的应用推荐方式并不一定能得到满意的推荐结果. 同时，碎片化特征导致移动应用的使用呈现出典型的流特征——多个相关移动应用按序使用完成粗粒度任务、满足用户需求. 此外，经常被混搭使用的移动应用相互之间也必然存在某种信息共享等关联关系. 在传统的基于用户历史记录、时空分布等上下文特征预测基础上，针对上下文和用户意图之间存在着很强的时间性和相关关系的特点，本文进一步利用移动应用间的潜在合作(移动应用间可重用服务参数，扩展使用流，支持更加复杂的任务需求)与竞争(推荐的移动应用与已执行结束的移动应用具有相似的功能，用于用户横向比较)关系以及体现用户的行为流特征，将 App_i 视为当前用户意图，以用户智能终端中已安装的应用构成 App_i 的候选集，利用协同 Nowcasting 模型来预测用户意图，即预测用户在当前时刻最可能用到的应用 App_i.

3.2.2　应用流的定制(UCP)

经由用户执行确认的应用流自动以 UCP 形式存储到知识库中，或者用于下次直接触发应用流执行，或者供用户使用模式挖掘，改善应用预测性能. 同时，UCP 还作为用户接口对已经存储下来的应用流进行管理，包括个性化的编辑定制或删除.

与应用流预测相比，直接从知识库中读取完整的应用流不需要经过复杂的计算，在面对移动用户重复性、模式化的业务活动时更有优势.

3.3　应用流执行

当检测到系统开机(BOOT_COMPLETED)、屏幕亮(ACTION_SCREEN_ON)、解锁(USER_PRESENT)或退出一个移动应用到系统主界面(ACTION_MAIN)等 Android 事件时，会触发应用流预测或者直接从 UCP 读取对应的配置文件，交由应用流执行模块调度实现移动应用批处理式执行. 应用流执行的关键包括三个方面：(1) 监控当前移动应用退出；(2) 基于参数注入方式的移动应用间数据共享复用；(3) 启动下一个应用. 如图 3 所示，应用流只能处于两种状态：等待和执行. 初始时应用流处于等待状态，处于流程首位的移动应用是应用流触发器，其启动事件将触发应用流转入执行状态. 应用流中的每个移动应用将按序执行、借助交互模块实现流内数重用和共享，直到应用流中最后的移动应用退出，整个应用流结束进入等待状态. 在应用流执行过程中，除非用户显式终止，否则执行状态将不会被打断.

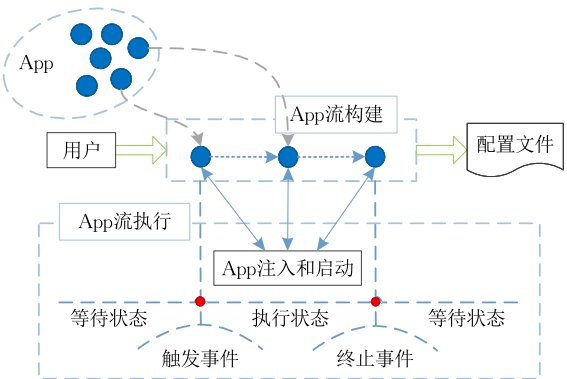


图 3　应用流执行原理

应用流执行包括参数提取、注入和启动. 参数提取基于应用分析获取的抽象文件中的 Item 语义信息,获取应用流的前序移动应用参数实例数据;参数传递利用注入方式,将前序移动应用参数数据共享给下一个移动应用,得到一个已经注入了服务参数的 UI,完成数据共享及应用流式按序启动. 其中,借鉴归纳编程[18-19]思想和众包方式,基于反馈-扩散(feedback-diffusion)机制,记录和学习移动应用参数复用.

4 关键技术与实现

4.1 应用流分析

框架将移动应用抽象为 item 的集合,本节将对 item 的存储、语义和语法细节进行描述,随后是应用分析流程涉及的关键技术.

4.1.1 Item 存储

本文只考虑 Android 开发中的 SQLite 数据库和 SharedPreferences 存储方式. 对于 SQLite 数据库方式,用户的一个输入值可能存储在一个字段或一个行记录中,称为 DbField 和 DbRow. 用户输入值也可能存储于通过键值访问和更新的 SharedPreferences 对象 SpObject 中,如表 1 所示.

表 1 DbField、DbRow 及 SpObject 间的比较

存储方式	支持多实例	键值形式	保存操作	注入方式
DbField	否	是	支持	更新
DbRow	是	否	返回 null	添加
SpObject	否	是	支持	更新

item 存储被设计为层次结构,顶层与 item 实例相关,较低的层次负责 item 实例与用户输入值的转换. 顶层支持表 1 所述的三种存储模式,较低层均使用 DbField 的形式存储. 对超过一层的 item 存储结构进行保存操作时,遵循存储层级结构从高到低的顺序转换为用户输入值. 相反的,在进行注入操作之前,用户输入值应该根据存储层级结构从低到高转换为相应的 item 实例.

图 4(a)和(b)分别表示当顶层采用 DbField 和 DbRow 模式时,保存和注入操作中的三层存储结构和数据流方向.

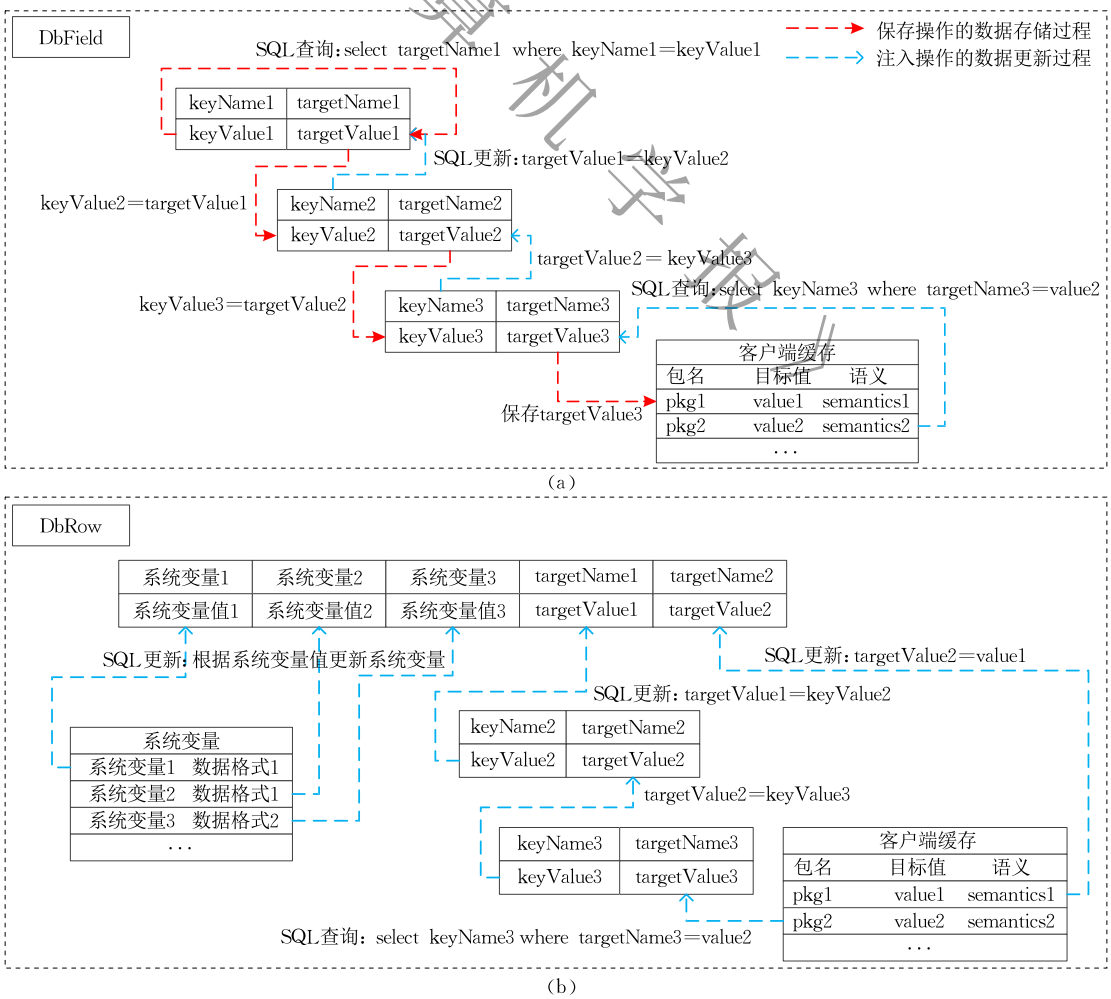


图 4 Item 存储的层次结构示意图

4.1.2　Item 语义

Item 语义用来进行不同应用间的 item 匹配，在实际分析验证大量的移动应用输入输出后，考虑到如果注入一个错误的用户输入值，移动应用将无法正常运行甚至会出现崩溃的情况，本文基于本体标注建立 item 语义间的关系，重点关注语义间的等价和隶属两种关系. 如果两个 item 语义概念具有等价关系，则意味着在某些情况下具有相同含义；如果是隶属关系，则代表两个概念间是包含关系. 例如，“猫途鹰”(TripAdvisor)①提供了旅行建议相关服务，“去哪儿网”提供了酒店预订服务，当用户指定旅行计划时，“猫途鹰”的目的地与“去哪儿网”的城市可能会构成等价关系；而“猫途鹰”的目的地又可能成为天气类移动应用“墨迹天气”的位置信息. 通过众包式的人工确认，能够确保以正确的用户输入值驱动移动应用的运行.

语义处理方面，利用基于反馈的 Web 服务语义标注方法[20]，将人工参与和标注重用相结合，兼顾标注效率的同时提高标注的质量，实现异构移动应用的数据共享. Item 语义作为抽象模型的元素之一，具有模板的性质，通过预定义 item 的语义匹配规范，一定程度上为移动应用间的 item 交互奠定了基础. 在首次执行移动应用流进行参数注入时，也引入用户对基于标注结果生成的参数实例进行确认的过程，进一步确认 item 标注结果，基于反馈-扩散机制优化标注质量的语义标注和数据共享过程将在 4.3 节具体论述.

4.1.3　Item 语法

除了语义消岐外，正确的语法格式对于确保 item 用户输入值的正确性也是极其重要的. 一些应用有它们自己的可接受数据格式，例如，对于日期来说可以有“yyyy-MM-dd”或者“yyyy/MM/dd”等格式. 在注入来自其它移动应用的用户输入值前，必须确保其格式已转换为可接受的格式并通过人工确认.

假设 $item_1$ 和 $item_2$ 来源于同一个移动应用的抽象，有如下两种特殊情况：

(1) 语义等价但存储位置不同，表示为

$$item_1.sematics == item_2.sematics \;\&\&\; item_1.storage \neq item_2.storage.$$

由于存储结构不同使得数据格式也不同，存在一个用户输入值会存储在不同物理位置的情况. 如在“携程”中，城市名称和城市代码虽然都表示出发或目的地城市，但通常存储在不同的位置.

(2) 语义不等价但存储位置相同，即同一位置保存多个 item 实例，表示为

$$item_1.sematics \neq item_2.sematics \;\&\&\; item_1.storage == item_2.storage.$$

某个物理位置也可能保存了多于一个的 item，例如在火车票预订移动应用中，出发城市和目的地城市保存在以“-”分割的同一物理位置.

移动应用的内部输入/输出被表示为〈存储，语义，语法〉三元组，以 XML 格式存贮于知识库中，描述录入该应用可接受及返回的数据形式，客户端通过 RESTful API 进行获取. 图 5 为抽象文件示例，在该示例中仅包含一个语义概念为“Location”的 item，其存储结构为两层，包括顶层的 SpObject 存储和底层的 DbField 转换，且在语法上没有格式要求. “complexoptions”标签表示其符合上述“同一位置保存多个 item 实例”的特殊情况.

```
<?xml version="1.0" encoding="UTF-8"?>
<app EorF="E" packageName="com.nuomi">
  - <item>
    - <storage style="sharedprefsLabel">
      - <sharedprefslabel complex="true" from="true" spTargetType="String" spKeyValue="cityselect_select_codes" spName="com.nuomi_preferences">
        - <targetfrom>
          - <databasefiled version="4" tableName="city" dbName="nuominew.db">
            - <dbkey>
                <dbfield fieldType="String" fieldName="city_id"/>
              </dbkey>
            - <dbtarget complex="false" from="false">
                <dbfield fieldType="String" fieldName="short_name"/>
              </dbtarget>
            </databasefiled>
          </targetfrom>
          <complexoptions addOrUpdate="false" separator=":" position="0"/>
        </sharedprefslabel>
      </storage>
      <semantic concept="Location"/>
      <syntax style="null"/>
    </item>
</app>
```

图 5　抽象文件示例图

4.1.4　APK 分析

本文基于 APK 静态分析的污点分析技术[21]及动态分析的自动化测试技术[22]构建 APK 分析过程，将移动应用映射为抽象信息. 整个 APK 分析过程分为三个阶段. 具体如下所示：

(1) 阶段 1，静态分析. 查找移动应用是否包含合

① https://www.tripadvisor.cn/

适的 item 信息.输入输出参数反映在 Android 的 Activity中,当 Android 操作系统重新创建 Activity 时会通过 onRestoreInstanceState 或者 onCreate 方法恢复先前的输入信息和状态数据,因此,从读取存储值到设置给界面中的组件这条数据流中包含着潜在的 item. FlowDroid[23-24] 是 Android 中的一个上下文敏感、流敏感、字段敏感、对象敏感、生命周期明确的静态污点分析工具.将从本地存储读取数据的 API 视为 source,而将设置界面中的组件值的 API 视为 sink,本文利用 FlowDroid 的 source API 和 sink API 探测移动应用中的数据流,可得到一个刻画移动应用输入输出的数据流集合.

图 6 是 APK 静态分析方法示意图. Activity 类和布局通过 setContentView 进行关联,并通过全局 ID 对界面中的组件进行操作.

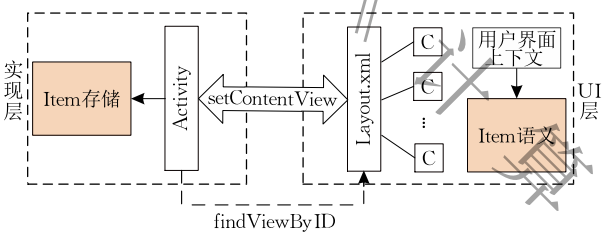


图 6 APK 静态分析方法示意图

(2)阶段 2,动态分析.减小阶段 1 所得数据流集合的规模.每个移动应用基本都包含许多 Activity,但其中部分 Activity 是不经常出现的.为了降低分析的复杂性,需要优先考虑常见的、主要的 Activity,而过滤掉需要多次交互才可以到达的 Activity.受自动化测试技术的启发,使用 Android 调试桥(ADB)、Monkey、View Server 等测试方法和工具模拟用户真实操作,且仅选择少于三次用户操作即可到达的页面.

首先,通过 ADB 命令启动模拟器,安装 APK 并启动该 APK.使用 Monkey 测试使移动应用达到常规运行状态,完全退出移动应用并使用 ADB 重启移动应用.发送"DUMP ffffffff"命令请求 View Server 以获取当前 Activity 的布局树信息,布局树的叶节点能够接收用户事件从而转到其他 Activity,使用 adb input tap 命令模拟点击每个叶节点.重复该过程直到所有三次内可达的 Activity 都访问结束.进一步减小了阶段 1 所获得的数据流集合的分析过程如图 7 所示.

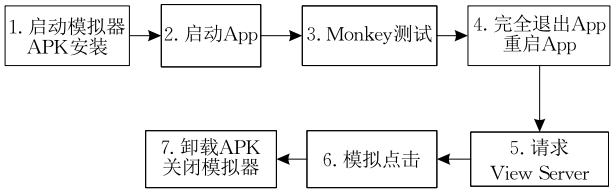


图 7 动态分析过程

(3)阶段 3,通过对前两阶段所得的数据流进行分析,获取其三元组信息,即 item 存储、语义和语法.图 8 展示了存储结构的分析过程:(a)表示采用前向代码分析得到的数据流分析过程,结果如(b)所示.(c)中展示了从 sink API 开始,采用反向控制流顺序分析获取参数,直到没有等价的表达式为止.对有实际意义的属性值作为参数进行 item 语义标注.在分析层面难以获取 item 语法信息,此时以基于众包机制的手工标注为主.

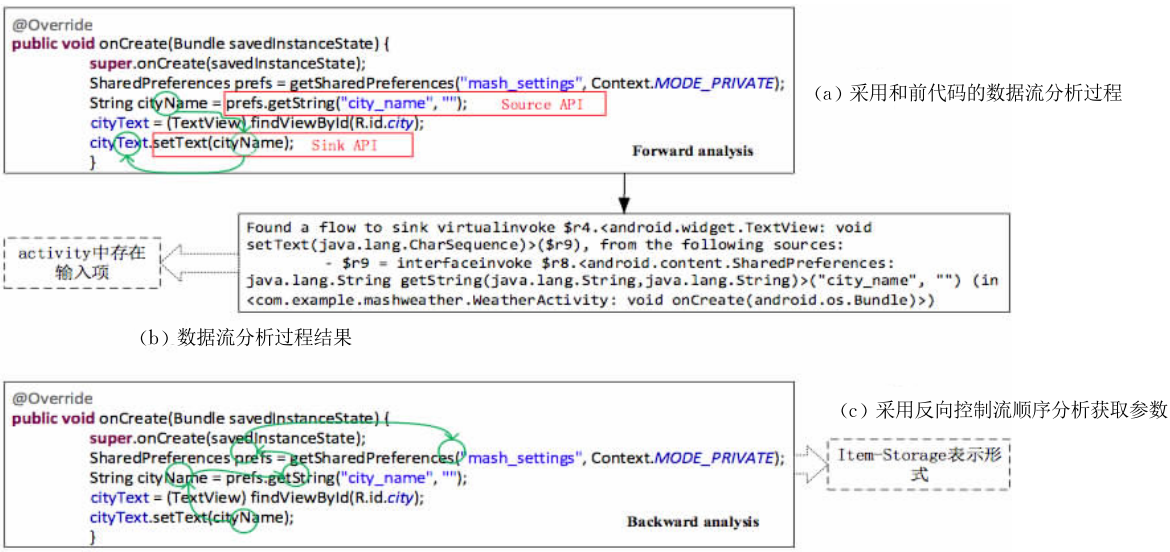


图 8 存储结构的分析过程

4.2　应用流构建

应用流表示为应用序列$\{App_0,App_1,\cdots,App_{i-1}\}$，基于移动应用预测的应用流构建就可看作是基于上下文、移动应用间关系及流特征，寻找用户最有可能使用的下一个应用即App_i的过程.

4.2.1　应用间关系

类似于经典的 Web 服务组合，App_i可能会与当前应用流$\{App_0,App_1,\cdots,App_{i-1}\}$构成合作关系，也可能与当前应用$App_{i-1}$构成竞争关系. 两种关系均表示向量形式，作为移动应用预测的一个输入特征. 合作关系与竞争关系的计算规则分别如下.

合作关系：通过计算App_t与当前移动应用流的参数语义相似性得到合作关系，用$Scoop_t$表示，是可重用度的平均概率，如式(1)所示，Abs_t是App_t的抽象，$CurrentFlow$表示当前移动应用流的 item 集合，$Reused$操作量化了基于当前流和语义关系的 item 可重用度. 等式(2)和(3)描述了$Reused$操作的计算细节，S表示 item 集合，se是 semantics 的缩写. 所有移动应用的$Scoop$共同构成合作向量.

$$Scoop_t=\frac{\sum_{item\in Abs_t} Reused(currentFlow,item)}{|Abs_t|} \tag{1}$$

$$Reused(S,item)=\max(\forall S_i\in S,Reused\_single(S_i,item)) \tag{2}$$

$$Reused\_single(S_i,item)=\begin{cases}1, & S_i\text{与 }item\text{ 语义等价}\\ 0.9, & S_i\text{与 }item\text{ 语义相似}\\ 0.8, & S_i\text{语义包含于 }item\\ 0.6, & item\text{ 语义包含于 }S_i\\ 0, & \text{其他}\end{cases} \tag{3}$$

竞争关系：以App_i和App_{i-1}之间的语义相似性刻画竞争关系，用$Scomp_t$表示，从而提供相似的候选服务. $Scomp_t$等价于Abs_t与Abs_{i-1}的 item 集合间的相似性，使用 Jaccard 相关系数计算，如式(4)所示.

$$Scomp_t=\frac{\sum_{item\in(Abs_{i-1}\cap Abs_t)} Reused(App_{i-1},item)}{|Abs_{i-1}\cup Abs_t|} \tag{4}$$

$$Abs_{i-1}\cap Abs_t=\{item\in Abs_t\,|\,Reused(App_{i-1},item)>0\} \tag{5}$$

$$Abs_{i-1}\cup Abs_t=(Abs_t/(Abs_{i-1}\cap Abs_t))\cup Abs_{i-1} \tag{6}$$

基于App_{i-1}计算 item 的可重用度，式(5)所示$Abs_{i-1}\cap Abs_t$是Abs_t的子集，使得 item 可以重用App_{i-1}的用户输入值. 式(6)所示的$Abs_{i-1}\cup Abs_t$是Abs_{i-1}和Abs_t与$Abs_{i-1}\cap Abs_t$差集的并集.

所有移动应用间的$Scomp$构成竞争向量. 移动应用相关性计算过程算法 1 所示.

算法 1.　APP 相关性计算算法 CalCorrelation.

输入. AppSet is the candidate set composed of all Apps installed in User's smartphone; and $\{App_0,App_1,\cdots,App_{i-1}\}$ is the current Appflow

输出. Scoop assigns the cooperation correlation; Scomp assigns the competation correlation

1. For each App_i in AppSet do
2. $Scoop[i]=\dfrac{\sum_{item\in app_i} Reused(\{app_0,\cdots,app_{i-1}\},item)}{|app_i|}$
3. $Scomp[i]=\dfrac{\sum_{item\in(app_{i-1}\cap app_i)} Reused(app_{i-1},item)}{|app_{i-1}\cup app_i|}$
4. endfor
5. return $Scoop$, $Scomp$

4.2.2　特征选择

本文中将预测特征分为三类：移动应用相关性特征、上下文特征、流特征.

移动应用相关性特征基于上文中的合作和竞争关系计算移动应用的参数语义相似性.

移动设备的硬件传感器能够准确采集周围环境的真实物理位置信息，已有研究证实时空信息在移动应用预测中起着至关重要的作用[7,25]. 本文选择了时间、星期、位置和上一个使用的移动应用作为上下文特征.

将用户智能终端中多个应用的相继启动看作是服从泊松分布[9]的顾客到达过程，基于历史日志信息构建应用使用图(App Usage Graph，AUG)来描述不同应用在使用过程上的转换关系. 其中，AUG 中节点V_i代表移动应用App_i，有向边E_{ij}表示从移动应用App_i切换到App_j，即App_j是App_i的后续应用. 边E_{ij}的权重表示转移概率，是关于两个移动应用启动时间间隔的函数.

使用 AUG 计算流特征可以将应用流$\{App_0,App_1,\cdots,App_{i-1}\}$表示为$\langle p_{0,t},p_{1,t},\cdots,p_{i-1,t}\rangle$，其中$p_{n,t}$表示在应用$App_n$返回主屏后直接或间接(经过 1 个或 1 个以上的移动应用后再)使用App_t的概率. 预测app_i时首先假设Θ_t是$app_i=app_t$的概率并计算app_t的流向量，其初始值设为所有移动应用有均等的概率. 其次采用迭代更新算法计算app_i加权流向量直至Θ_t收敛. 具有相似语义参数的移动应用属于同一个应用流的可能性更高，为提高计算Θ_t的效率，本文使用归一化$Scoop_t$替代平均概率Θ_t进行迭代更新计算.

4.2.3　Nowcasting 和 Forecasting 集成预测

基于 Nowcasting 方法的移动应用预测流程如图 9 所示. App_i是用户的意图，移动设备中的每个

移动应用都有可能成为当前选择，并且同一时刻只能出现最多一个期望的移动应用. 将相关性特征、上下文特征、流特征及意图表示为时间序列，如图 10 所示. 考虑到应用程序的使用呈现以周为周期的特性[2,26]以及数据稀疏性问题，本文选择同一用户不同周的历史数据构建张量 X. 采用 CP 张量分解技术 $P=parafac\_als(X,R)$ 获取不同周的协同潜在因子(令 $R=4$)；使用卡尔曼(Kalman)滤波获取当前时刻的潜在协同因子；采用最小二乘回归，对当前的潜在协作因子与最近使用的移动应用进行线性回归；最后对当前时刻所有移动应用的预测意图估计值排序得到预测结果.

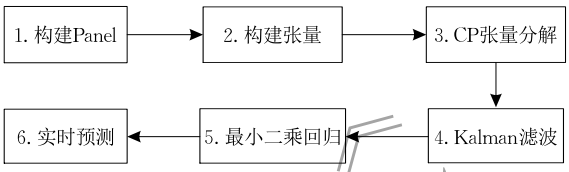


图 9 Nowcasting 方法计算过程

TIME STEP	10:30	11:00	11:30	12:00	NOW(12:20)
OFFICE	1	0	0	0	0
HOME	0	0	0	1	1
FLOWFEA(APP_X)	0	0.3	0.5	0.2	0.3
FLOWFEA(APP_Y)	0	0	0	0.6	0
LASTAPP(APP_1)	0	0	0	1	1
LASTAPP(APP_2)	1	0	0	0	0
APP1	0	0	0	0	??
APP2	1	0	0	0	??

图 10 Panel 示例图

本文将涉及大量矩阵和向量运算的 Nowcasting 算法以 MATLAB 实现，并封装成成 Jar 包在 Eclipse IDE 开发环境中调用.

在 Forecasting 部分，采用 K 近邻分类算法，并将移动应用本身作为分类标签，K 取 20%进行移动应用预测，算法 2 描述了 Forecasting 的计算过程.

算法 2. Forecasting 计算算法 CalForecasting.

输入. TS is the Training Set (80% App usage record)
cr is the current usage record

输出. forecasting App is the forecasting recommendation Set

1. foreach App_i in TS do
2. dis_i=distance(App_i,cr); //特征相似度计算
3. endfor
4. RecordSet is top 20% nearest dis;
5. Map〈App,sum〉map; //build a map which is the //CUSUM of the distance's reciprocal between App //and feature
6. foreach App_i in RecordSet do
7. map($App_i.app$)+=(1/dis_i);
8. end
9. $FApp$=ordering map.key in map.value descending //sort the key of map according to the desc of //map' value and set to $FApp$
10. return $FApp$

综合 *Nowcasting* 和 *Forecasting* 的推荐结果，并以弹框的形式展示，供用户选择确认.

4.3 应用流执行

4.3.1 参数对应

当用户从弹出的推荐结果列表中选择了一个移动应用图标后，应用流自动协同框架基于自定义的语义库和语义相似性计算并搜索出合适的参数注入到选中的移动应用界面中，实现移动应用间的参数对应和数据共享. 由于标注可能存在问题且可能存在多参数匹配的情况，此处需要由用户进行交互确认服务参数. 此处对于参数语义相似性计算，仍然利用移动应用预测中的 *Reused_single*，即式(3)的规则. 对于参数语义库，本文在对大量移动应用服务参数进行分析标注的基础上，构建了包括相同、等价、包含在内的参数映射关系，并且它们的关联性依次减弱.

参数传递类似于一个中介，在打破移动应用边界实现数据共享中起到了关键作用. 即使用户与移动应用的交互次数不多，但类似的人机交互过程可进一步解决语义冲突、避免语义层面的歧义. 借助人机交互中的迭代确认和反馈学习可从结构化或小规模数据集中归纳总结出规律，并据此产生出新的数据.

参数重用中的归纳过程如图 11 所示. 经归纳编程构建的每一条规则，如 $Reuse(P_1,P_2,P)$，表示参数 P_1 重用参数 P_2 的值的可能性为概率 P. 用式(7)计算参数对应值，选择使得 PS 值最大的 P_2 作为参数 P_1 的对应参数.

$$PS_{P_1,P_2}=P\times Reused\_single(P_1,P_2) \quad (7)$$

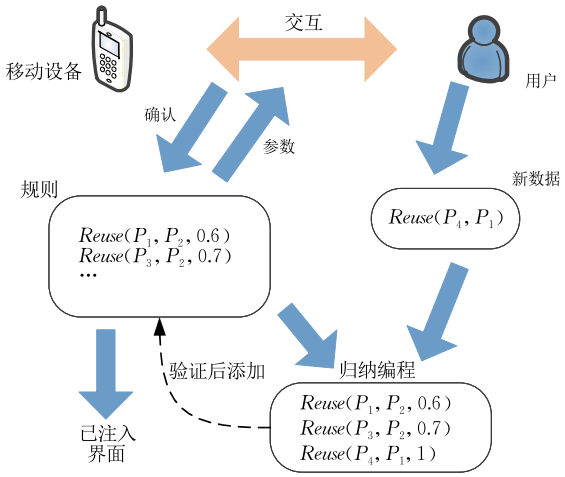


图 11 参数重用中的归纳过程

在预填之前，询问用户是否需要修改参数值匹配并确认，P 值由此进行更新. 规则集体现了用户的短期使用习惯.

4.3.2　参数传递

部署于用户设备内的应用流自动协同框架客户端类似于传统 Web 服务组合中的中介器，隔离移动应用间的直接交互，借助移动应用的抽象模型实现参数传递，参数传递示意图如图 12 所示.

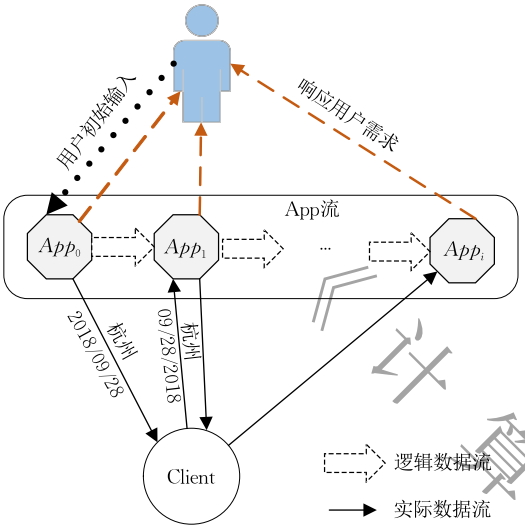


图 12　参数传递示意图

图 13 展示了参数提取、注入和启动的完整过程. 参数提取如图 13(a)所示，当监听到用户退出或者特定事件时，客户端获取应用 App_{i-1} 内的用户输入值以备后续共享重用. 针对 App_{i-1} 抽象信息中的每一个 Item 执行提取操作：按照由顶至底的顺序读取 Item 实例，转化为以 Key-Value 对的形式. 其中，Key 为标注 Item 的语义概念，Value 为用户数据输入实例.

注入过程如图 13(b)所示，实质是用缓存中的用户输入值更新 App_i 启动所需要的 item 实例内容，实现异构移动应用数据共享，所以，注入可看做是参数提取的反过程. 首先确定期望的应用，即 App_i，这可通过应用流预测推荐或直接读取预存的 UCP 实现. 接着，在获取 App_i 抽象信息基础上构造输入数据，也就是利用客户端缓存中与之语义相似度最高的用户输入值更新实例. 即依据 App_i 抽象信息对用户输入值进行语法格式转化，并按照 Item Storage 由底向上的顺序更新 Item 实例. 若有多个语义相似的概念，本文采用的策略是选择时间上最近一次保存的用户输入值. 最后，当所有 item 都注入后通过重载 ClassLoader，实现如图 13(c)所示的代码启动 App_i.

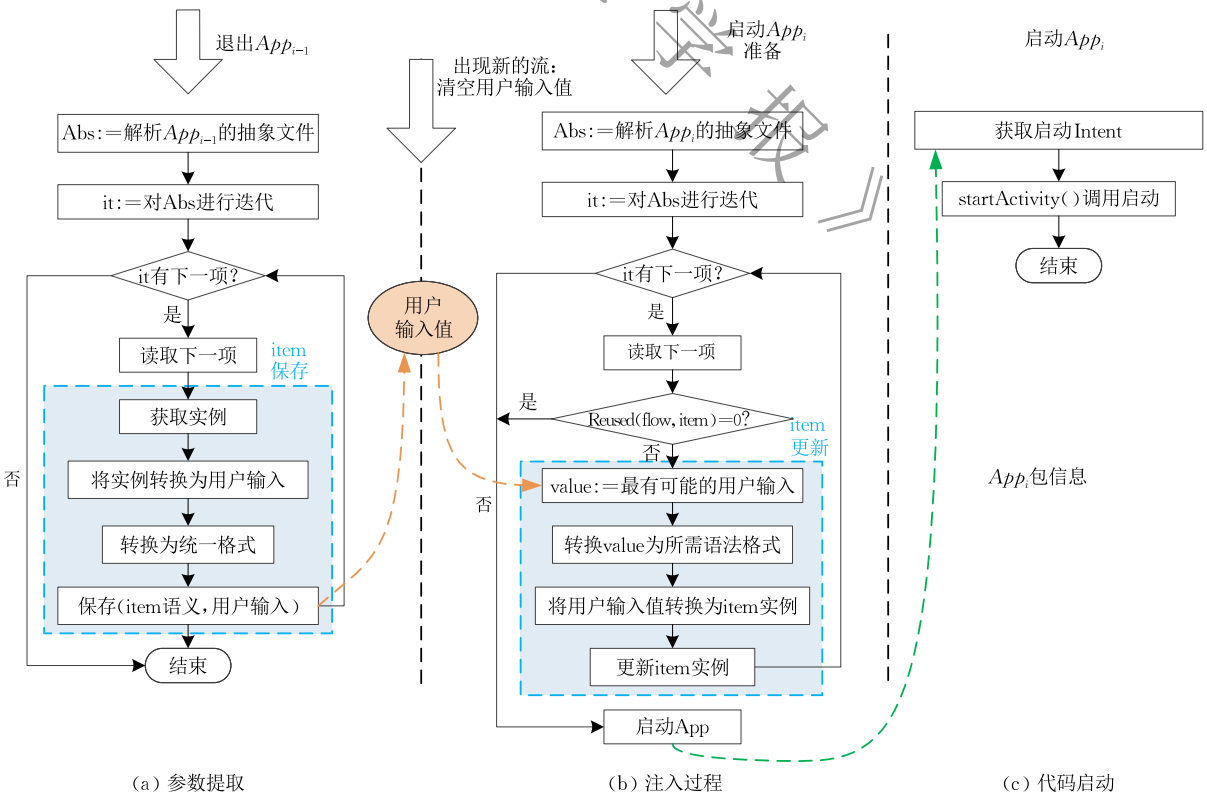


图 13　参数提取、注入和启动过程

应用流执行伪代码如算法 3 所示.

算法 3. 应用流执行算法 FlowExecution.

输入. Flow, the existed App flow

```
1. foreach App_i in Flow do
2.     foreach item_i in App_i do
3.         read; //get value of item_i
4.         save; //save item_i's semantics and value
5.     end
6.     get App_{i+1} of Flow;
7.     foreach item_i in App_{i+1} do
8.         item correspondence;
9.         inject item_i mainly based on item storage;
10.    end
11.    get package and class information of App_{i+1};
12.    construct Intent of App_{i+1};
13.    start Activity to launch App_{i+1};
14.    if App_{i+1} is the last App of Flow then
15.        break;
16.    end
17. end
```

5 App 流推荐实验

为评估移动应用流推荐方案的有效行,本文设计了一个简单的 Android 后台应用,收集了课题组 10 名志愿者三周内的手机使用情况,共 6233 条记录,格式为〈$AppName$,$Date$,$Time$,$Location$〉. 对数据的预处理主要有:

(1) 由志愿者对地理位置进行标注,添加相应的标签,如 Lab、Dorm 等;

(2) 根据时间上最邻近的上一条记录标注上一次使用的移动应用;将记录格式化为〈$AppName$,$Date$,$Time$,$LocationTag$,$LastApp$〉;

(3) 筛选时间间隔小于某一阈值 Δt(本文中取 $\Delta t=1$ m)的连续启动,标注为应用流,如 $App_0-App_1-App_2-\cdots-App_{i-1}$. 其中 App_0 即为前序应用,App_1、App_2 等就是相应的后继.

算法由 JAVA 和 MATLAB 混合编程实现,运行于 2.8GHz Intel CPU、12GB 内存、64 位 Windows 8 操作系统的实验环境,以前两周数据作为训练集,第三周数据为测试集.

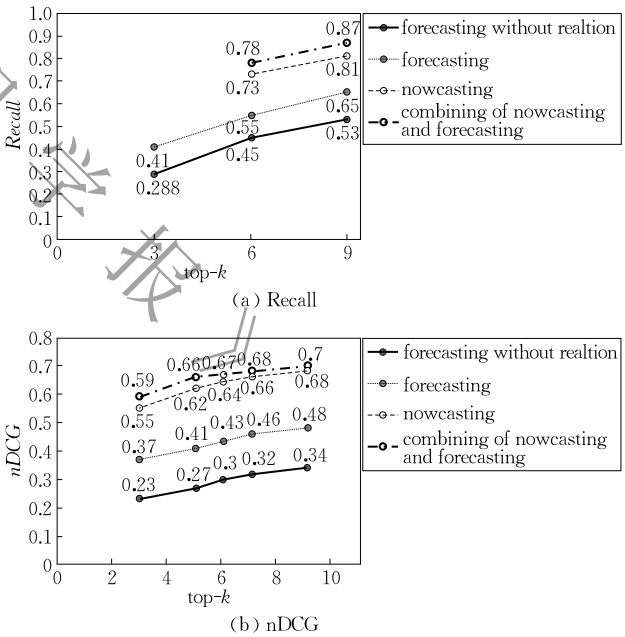
实验评价指标采用文献[9]中平均召回率和归一化折损累积增益 $nDCG$,平均召回率用来衡量系统的预测结果的命中率; $nDCG$ 用来衡量结果排序质量. 计算公式如式(8)、(9)所示:

$$AverageRecall=\frac{\sum_{C_i} I(Rec_i \cdot app_i)}{|C|} \tag{8}$$

$$nDCG=\frac{\sum_{c_i\in C}\frac{1}{\log_2(p+1)}}{|C|} \tag{9}$$

其中,C_i 为测试用例集合 C 中的某一个测试用例; $I(Rec_i \cdot App_i)$ 函数在推荐列表 Rec_i 命中真实结果 App_i 时取 1,否则为 0. p 表示 App_i 在推荐列表中的排序位置,没有命中时 $nDCG$ 为 0.

实验比较了“不使用相关性特征的 forecasting 预测”、“forecasting 预测”、“nowcasting 预测”以及“forecasting 和 nowcasting 结合预测”四种情况下,推荐 top3、top6、top9 个移动应用的实验结果,如图 14 所示. 可以看出将 forecasting 和 nowcasting 结合的方法在平均召回率(图 14(a))和归一化折损累积增益(图 14(b))均好于其他方法,其中平均召回率可以达到 0.7 以上. 考虑智能手机的屏幕特征,本文选择 top6 结果展示给用户.



(a) Recall

(b) nDCG

图 14 比较实验结果

6 案例研究

本文在 Android 7.1 平台上构建了应用流自动协同框架原型系统,针对旅行外出典型场景进行案例研究并展示原型系统的使用效果.

假设课题组某同学要到杭州参加 2018 年 11 月 12～15 日的 ICSOC 学术会议,在出发前做如下准备:

(1)查询并预定 11 月 12 日从当前位置“天津”到目的地“杭州”的高铁车票；

(2)查询并预定杭州市的某一酒店，入住时间为 11 月 12 日，离店时间 11 月 15 日. 额外约束：距离会议地点凯悦酒店 2 公里范围内；

(3)查询推荐餐厅、特色周边等信息；

(4)查询 11 月 12～15 日杭州市的天气情况；

(5)将订票信息、酒店预订信息、天气情况等保存到备忘录中以供随时查阅.

从豌豆荚应用市场下载安装了 9 个与旅行外出相关的应用. 其中，“携程旅行”和“智行火车票”提供高铁车票预订服务，“艺龙酒店”和“去哪儿网”负责提供酒店预订服务，“美团团购”和“百度糯米”负责提供团购服务，“墨迹天气”和“2345 天气”提供天气服务，“高德地图”则提供地图导航服务. 以上 9 个第三方移动应用安装到 LG 移动设备中，通过 APK 分析将其抽象信息保存在移动设备上，经过人工确认并补充语义信息.

当用户执行旅游计划时，首先启动智行火车票移动应用搜索并预订火车票信息，图 15 中的 Step1 展示了“智行火车票”所需的用户输入包括出发城市、目的城市、出发日期.

图 15　基于预测的移动应用协同效果展示

用户退出“智行火车票”后，应用流自动协同框架被激活进行后续移动应用推荐. 推荐结果以弹框方式展示给用户. 一旦用户点击某个图标，如“艺龙酒店”，复用来自“智行火车票”中的用户输入或输出直接启动到“艺龙酒店”的预订界面，如图 15 的 Step2 所示. 后续 Step3 及 Step4 也是如此，分别是预测并选择使用“墨迹天气”和“高德地图”的情形.

除基于应用流的移动应用推荐外，用户也可对已有的 UCP 应用流进行个性化编辑或者完全手动定制应用流，如图 16(a)所示. 图 16(b)显示用户新建的一条应用流以“携程旅行”为触发器，按序包含“去哪儿网”、“美团团购”、“2345 天气”和“高德地图”5 个移动应用. 每个应用启动前都将弹出图 16(c)所示的窗口，由用户决定是否继续应用流的执行. 一旦用户点击是则去哪儿网将自动启动，并且因为已经通过参数注入从“携程旅行”共享了必要的信息如目的地，“去哪儿网”的启动执行不需要用户手工的重复输入信息. 于是，基于参数注入和数据共享，应用流中后续的其他移动应用如“美团团购”、“2345 天气”和“高德地图”即依次启动批处理式执行. 演示视频可在 https://pan.baidu.com/s/19YhKbNRsl3ziLXt_4DBEQ(提取码 drdt)查看.

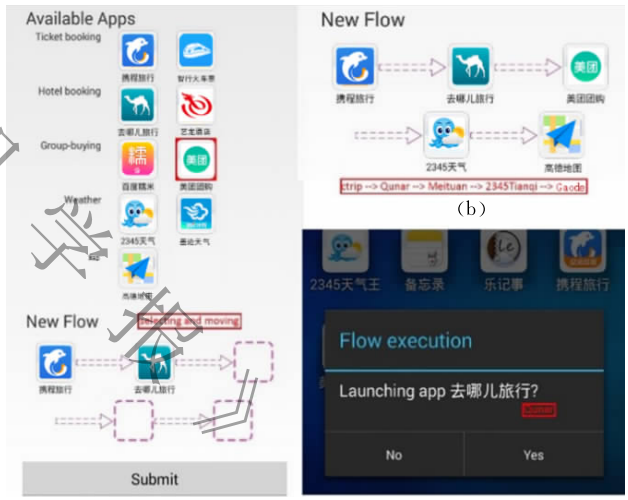


图 16　UCP 效果展示

考虑到移动设备的计算能力与电池供应问题，本文在实现时选择的策略是：UCP 方式优先于应用流推荐，即当某次推荐后用户选择的移动应用存在与 UCP 中已有的某一个或者多个流程当中，则下一轮的推荐结果直接由 UCP 中当前应用的后继应用组成. 并且当 UCP 中的预存应用流执行次数超过某一设定阈值后，移动应用推荐进入睡眠模式直接读取执行已有的应用流.

与传统的用户为中介的手动输入驱动相比，基于协同框架重用移动应用参数的自动协同，如图 17 所示，有效地减少用户操作、改善用户交互体验、优化了移动应用流和信息流的自动化构建.

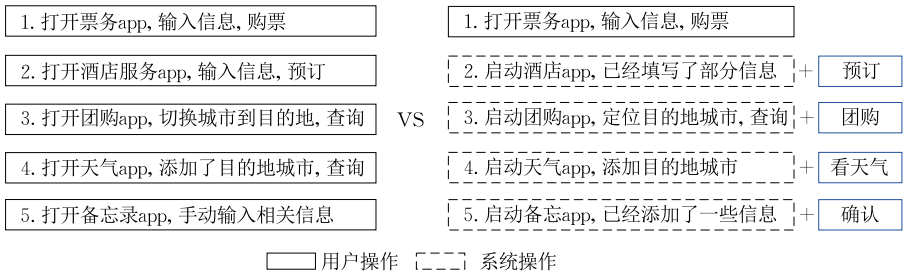


图 17 用户交互方式对比

7 讨 论

本文提出了一种基于移动应用无缝集成的应用流自动协同框架,不仅支持 UCP,还支持在移动应用流配置文件缺乏的情况下的移动应用预测. 该框架在很大程度上改善了用户体验,但在实验中也发现了一些不足之处,这将是我们今后研究工作的重点内容:

(1) 复杂工作流问题. 在常见的服务组合及用户场景中,不仅存在顺序流,往往还会出现较复杂的调用流,如循环流等. 由于移动设备本身的限制,用户在使用移动应用过程中一般只涉及顺序流. 因此,本文目前仅侧重于对顺序流的研究,今后将逐步考虑加入对其他调用方式的研究. 循环流的研究,将是后续研究的重点,通过分析用户的实际需求,找到循环流的入口移动应用,进而分析移动应用间的协同情况,以支持复杂的调用流程及复杂的实际应用场景;

(2) 人工辅助问题. 目前,本文中的应用分析工作还需要人工参与及确认. 但是,至少该过程是一次性的,只需在分析结束后提供给开发人员确认即可立即交付用户. 为了减少人工依赖,本文将静态分析与动态分析过程结合为一种工具,借助 PC 端与移动端的映射来完成两个阶段的交互过程,并在分析过程中根据移动应用参数的语义映射关系生成语义关系库,基于众包和反馈-扩散机制学习和复用人工确认的成果,这些措施使本文的研究工作更具有扩展性与通用性;

(3) 数据安全问题. 由于该研究需要监控获取服务参数,移动应用间的通信采用进程间通信方式,在一定程度上涉及到用户隐私及数据安全问题. 首先,移动应用间数据共享与交互在用户客户端进行,无额外数据被发送到服务器端;其次,应用流中的移动应用在交互时,必然会跨越移动应用自身边界,本文提出的应用流协同框架只是将原本人工交互的过程自动化,未跨越移动应用原有的限制,造成数据泄露;最后,本文在处理一次应用流交互时使用了缓存数据,但随着交互过程的结束,缓存数据也随之失效,并不会保存在客户端或服务器端. 本文从各方面保证了用户的隐私,但随着目前学术界对于信息安全问题的深入研究,本文将在后续工作中不断吸收该领域的最新研究成果,进一步保证用户的隐私安全.

8 结 论

移动应用功能的碎片化严重影响用户的使用体验,本文提出一种 Android 平台移动应用的自动协同框架. 首先利用 APK 静态分析和动态分析技术定义了一个基于服务的移动应用抽象模型;其次根据该模型设计了面向应用流的移动应用预测方法,在上下文基础上考虑移动应用相关关系和应用流作为预测特征,将 Nowcasting 技术结合到 Forecasting 预测中;最后是基于参数注入方式重用前序移动应用数据并唤醒后续移动应用,实现异构移动应用间信息共享和批处理式执行的动态协同,兼有应用预测和客户定制两种方案的优点:

(1) 基于应用流的移动应用预测和推荐,避免了用户参与的自定义过程,同时基于反馈和归纳学习的人机交互也进一步改善了推荐结果的精确度;

(2) 基于 UCP 应用流执行,在保留移动计算的灵活性和个性化特征前提下避免了应用预测所需的复杂计算,同时在不同用户之间分享应用流程配置文件,在一定程度上克服因缺少用户历史记录而导致的冷启动问题.

总之,本文所提出并构建的框架可有效地打破移动应用间无交互的障碍,在面对日渐复杂的用户需求极大改善用户体验. 未来仍然有一些工作值得扩展:(1) 抽象模型有待进一步丰富,区分前端 UI、后台云服务以及第三方云服务等,以便兼容更多交互需求和复杂业务流程;(2) 对应用流执行过程中可能存在安全隐患问题进行分析和优化.

参 考 文 献

[1] Yan T, Chu D, Ganesan D, et al. Fast app launching for mobile devices using predictive user context//Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services. Lake District, UK, 2012: 113-126

[2] Liao Z X, Lei P R, Shen T J, et al. Mining temporal profiles of mobile applications for usage prediction//Proceedings of the 2013 IEEE 13th International Conference on Data Mining Workshops. Atlantic City, USA, 2012: 890-893

[3] Lee M, Bak C, Lee J W. A prediction and auto-execution system of smartphone application services based on user context-awareness. Journal of Systems Architecture, 2014, 60(8): 702-710

[4] Rukzio E, Noda C, Luca A D, et al. Automatic form filling on mobile devices. Pervasive & Mobile Computing, 2008, 4(2): 161-181

[5] Sun Y, Yuan N J, Xie X, et al. Collaborative Nowcasting for contextual recommendation//Proceedings of the 25th International Conference on World Wide Web Steering Committee. Montréal, Canada, 2016: 1407-1418

[6] Sun C, Zheng J, Yao H, et al. AppRush: Using dynamic shortcuts to facilitate application launching on mobile devices. Procedia Computer Science, 2013, 19: 445-452

[7] Shin C, Hong J H, Dey A K. Understanding and prediction of mobile application usage for smart phones//Proceedings of the 14th International Conference on Ubiquitous Computing. Pittsburgh, USA, 2012: 173-182

[8] Lu H C, Lin Y W, Ciou J B. Mining mobile application sequential patterns for usage prediction//Proceedings of the 2014 IEEE International Conference on Granular Computing. Noboribetsu, Japan, 2014: 185-190

[9] Liao Z X, Li S C, Peng W C, et al. On the feature discovery for app usage prediction in smartphones//Proceedings of the 2013 IEEE 13th International Conference on Data Mining. Dallas, USA, 2013: 1127-1132

[10] Baeza-Yates R, Jiang D, Silvestri F, et al. Predicting the next app that you are going to use//Proceedings of the 8th ACM International Conference on Web Search and Data Mining. Shanghai, China, 2015: 285-294

[11] Ma Y, Liu X, Yu M, et al. MashDroid: An approach to mobile-oriented dynamic services discovery and composition by in-app search//Proceedings of the 2015 IEEE International Conference on Web Services. New York, USA, 2015: 725-730

[12] Malakuti S. Modular composition of multiple applications with architectural event modules. Software Practice & Experience, 2017, 47(7): 1013-1025

[13] Deng S, Wu H, Tan W, et al. Mobile service selection for composition: An energy consumption perspective. IEEE Transactions on Automation Science & Engineering, 2017, 14(3): 1478-1490

[14] Deng S, Huang L, Taheri J, et al. Mobility-aware service composition in mobile communities. IEEE Transactions on Systems Man & Cybernetics Systems, 2016, 47(3): 555-568

[15] Boulakbech M, Messai N, Sam Y, et al. Visual configuration for RESTful mobile Web mashups//Proceedings of the 2017 IEEE International Conference on Web Services. Honolulu, USA, 2017: 870-873

[16] Wang Gui-Ling, Han Yan-Bo, Zhang Zhong-Mei, et al. Cloud-based integration and service of streaming data. Chinese Journal of Computer, 2017, 40(1): 107-125 (in Chinese)
(王桂玲，韩燕波，张仲妹等. 基于云计算的流数据集成与服务. 计算机学报，2017，40(1)：107-125)

[17] Wang Chong, He Ke-Qing, Wang Jian, et al. On-demand interoperability techniques for software model services: A standardized solution for the information islands crisis. Chinese Journal of Computer, 2018, 41(6): 1314-1331 (in Chinese)
(王翀，何克清，王健等. 化解“信息孤岛”危机的软件模型按需服务互操作技术. 计算机学报，2018，41(6)：1314-1331)

[18] Flener P, Schmid U. An introduction to inductive programming. Artificial Intelligence Review, 2008, 29(1): 45-62

[19] Mitchell N, Koopman P, Plasmeijer R, et al. Approaches and applications of inductive programming. 2010, 82(16): 5400-5403

[20] Sun X, Chen S, Feng Z, et al. A service annotation quality improvement approach based on efficient human intervention //Proceedings of the 2018 IEEE International Conference on Web Services. San Francisco, USA, 2018: 107-114

[21] Huang W, Dong Y, Milanova A, et al. Scalable and precise taint analysis for Android//Proceedings of the 2015 International Symposium on Software Testing and Analysis. Maryland, USA, 2015: 106-117

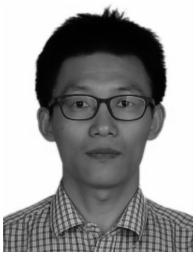
[22] Amalfitano D, Fasolino A R, Tramontana P, et al. Using GUI ripping for automated testing of Android applications//Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. Essen, Germany, 2012: 258-261

[23] Arzt S, Rasthofer S, Fritz C, et al. FlowDroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. ACM Sigplan Notices, 2014, 49(6): 259-269

[24] Fritz C, Arzt S, Rasthofer S, et al. Highly precise taint analysis for Android applications. EC SPRIDE & Technische Universitat Darmstadt, Germany: Technical Report: TUD-CS-2013-0113, 2013

[25] Parate A, Hmer M, Chu D, et al. Practical prediction and prefetch for faster access to applications on mobile phones//Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing. Zurich, Switzerland, 2013: 275-284

[26] Fan Y, Li H, Wu Q, et al. Mining and predicting smart device user behavior//Proceedings of the Asia-Pacific Advanced Network. Bandung, Indonesia, 2014: 65-77



CHEN Shi-Zhan, born in 1975, Ph. D., associate professor. His research interests include service computing and the service-oriented architecture.

WANG Ru, born in 1995, M. S. candidate. Her research interest is service computing.

FENG Zhi-Yong, born in 1965, Ph. D., professor. His research interests include knowledge engineering, service computing and computer cognition.

XUE Xiao, born in 1979, Ph. D., professor. His research interests include service computing and computational experiment.

HE Qiang, born in 1982, senior assistant professor. His research interests include software engineering, cloud computing, service computing and big data analysis.

LIN Mei-Chen, born in 1992, M. S. Her research interest is service computing.

Background

The framework discussed in this paper is about the integration of mobile computing and service computing. This paper focuses on the collaboration between heterogeneous Apps, including App composition and service parameters reuse. It can be considered as an integration problem of service composition, selection and recommendation. We put this old wine in a new bottle, but it does work this time.

In the past years, traditional service computing has made great progress. Due to the particularity of mobile domain, Apps cannot interact with each other. Many researches tried to break the boundaries between Apps, including the open resources and code modification of Apps. As for service recommendation, existing methods can predict users' real requirements but ignore the relations between services.

At the application level, an App automatic collaboration framework is proposed to solve the challenges of service composition, selection and recommendation in mobile computing. From the perspective of users, they do not need to repeat parameter input and transfer between Apps manually. From the perspective of App developers, they do not need to do code modification to achieve the interaction between different Apps.

This work is supported by the National Natural Science Foundation of China under Grant No. 61572350 and the National Key R&D Program of China under Grant No. 2017YFB1401201.