面向用户使用习惯的安卓应用自适应方法

沈立炜 宁柯丞 赵文耘

(复旦大学软件学院 上海 201203) (上海市数据科学重点实验室(复旦大学) 上海 201203)

摘 要 使安卓应用具有适应用户个性化使用习惯的能力可提升其用户体验。针对该目标,提出了一种面向用户使用习惯的安卓应用自适应方法。该方法结合软件自适应技术,以用户使用习惯为自适应的上下文因素,同时将自适应目标聚焦在动态管理应用的活动跳转序列上。为了支持开发者实现这样的应用,归纳了一组设计需求,并以两个用户使用场景为例展现了安卓应用构造过程中对设计需求的实现细节。

关键词 用户使用习惯,自适应,安卓应用,个性化

中图法分类号 TP31 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.023

Usage Habit-oriented Self-adaptive Method for Android Applications

SHEN Li-wei NING Ke-cheng ZHAO Wen-yun

(School of Software, Fudan University, Shanghai 201203, China)

(Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 201203, China)

Abstract Endowing Android applications with the capability of adapting to users' personalized usage habits is able to improve their user experience. Aiming at this objective, this paper proposed a usage habit-oriented self-adaptive method for Android applications. Based on the conventional self-adaptive mechanism, this method takes the usage habits as the context factor. Meanwhile, the adaptation implementation of the method focuses on the ma-nagement of the transition sequence of the activities in an application. The paper further concluded a set of designing requirements in order to support developers to construct this kind of applications. In addition, two usage scenarios are illustrated to explain the implementation details towards the designing requirements in the application development phase.

Keywords User habit, Self-adaptive, Android application, Personalization

1 概述

根据调研机构 Strategy Analytics 的统计,安卓(Android)在 2014 年末具有 81.5%的移动设备操作系统的市场占有率。由于安卓操作系统的开放性以及发布的便捷性,基于安卓的 App 正以非常快的速度不断涌现,这使得市场中同一品类的安卓 App 之间的竞争也变得愈发激烈。在这种背景下,具有更好的用户体验且能满足用户个性化使用需求的安卓 App 才会获得更多青睐。

当前的安卓应用开发者往往会预先设计用户的使用流程而后进行实现,这类固化的 App 具有统一的使用方式,要求用户掌握正确的操作流程,因而在某些场景下无法满足用户的个性化使用需求。为了将这种"用户围绕 App 转"的现实格局转换为"App 围绕用户转",我们认为安卓应用开发者应赋予 App 根据用户使用习惯改变自身操作流程的能力。例如,某一用户仅常用 App 的某项功能,而与该功能相关的活动(activity)的触发需要用户进行多层次的菜单导航,用户很可能会对多余的菜单项点选操作感觉厌烦。若该 App 针对用户使用习惯在应用启动后直接跳转至相应的活动界面,则能够改善其使用体验。

针对这一目标,本文提出了一种面向用户使用习惯的安卓应用自适应方法。该方法结合软件自适应技术,将用户的使用习惯视为导致软件自适应的上下文因素,同时将自适应的目标聚焦在动态管理安卓 App 的活动跳转序列之上。为了支持开发者实现具有自适应能力的安卓 App,本文归纳了一组设计需求,给出了简要规范,并以两个用户使用场景为例展现了安卓 App 构造过程中对设计需求的实现细节。本文所提供的方法仅是对用户操作流程自适应的初步探索,由于不同场景下用户的操作具有异构性,目前还未抽象出通用的实现框架,它是在设计需求的指导下针对特定的使用场景实现相应的需求。

本文第 2 节列举了相关背景工作以及其他研究成果;第 3 节在传统自适应技术的基础上归纳了面向用户使用习惯的自适应设计需求,并对每一个组成部分进行了阐述;第 4 节展示了安卓 App 的两个应用场景,以及为了达到自适应目标而对设计需求进行的具体实现;最后是对全文的总结与展望。

到稿日期:2015-11-30 返修日期:2016-02-27 本文受国家自然科学基金(61402113),国家高技术研究发展计划(863)(2013AA01A605)资助。

沈立炜(1982—),男,博士,副教授,主要研究领域为软件产品线、自适应软件系统;宁柯丞(1993—),男,主要研究领域为安卓应用开发技术;赵文耘(1964—),男,教授,博士生导师,主要研究领域为电子商务、移动应用软件开发技术。

2　相关工作

软件自适应是指软件在运行过程中能够实时收集系统的各种变化信息，并根据预先设定好的策略在必要时对自身进行自动调整以更好地为用户提供服务的能力[1]。在软件自适应领域，2001 年 IBM 首次提出了自主计算(Autonomic Computing)的概念，将其归纳为软件的自配置、自优化、自修复以及自保护 4 种能力，同时还提供了一种自主控制循环模型即 MAPE-K Loop 模型来指导其实现[2]。

Dehlinger 等[3]在其对移动应用软件工程的展望中将设计上下文感知的、具有自适应能力的移动应用作为主要的挑战。当前，已有诸多工作聚焦于此。例如，Mc-Kenley等[4]通过将自适应能力以程序分支的方式硬编码在原始应用程序中来管理 App 在非预期场景中的行为。Nakagawa 等[5]将自适应机制应用于软件维护领域，使用 MAPE 模型在不改变应用代码的前提下实现了嵌入式系统的运行时行为变更。Cugola 等[6-7]则提出了一种声明式的方法来定义 App 的自适应需求与行为，并依赖于中间件来实现 App 在运行时的动态调整。另外，Datta 等[8]提出了一个自适应的开发框架，赋予了安卓应用在手机电池电量较低的时候进行动态功能调整(通过暂停部分功能使得电池续航时间变长)的能力。上述工作都以实现移动 App 功能的运行时变更为研究目标，他们所关注的上下文变化因素往往是外部第三方服务的可用性、QoS、移动设备的电量、性能等方面，同时包括在 App 的构件重配置、功能模块的局部改造等层次上执行自适应的行为。与之相比，本文方法所关注的上下文变化因素是用户使用习惯，即用户使用 App 的常用交互流程，另外所规划的自适应行为能够持续地动态管理安卓 App 的活动跳转序列，即自动执行某些 activity 之间的跳转。

3　方法架构与设计需求

3.1　面向用户使用习惯的自适应方法架构

本文提出的面向用户使用习惯的安卓应用自适应方法架构示意图如图 1 所示，方法中的流程借鉴了软件自适应技术理论中的 MAPE-K 模型。

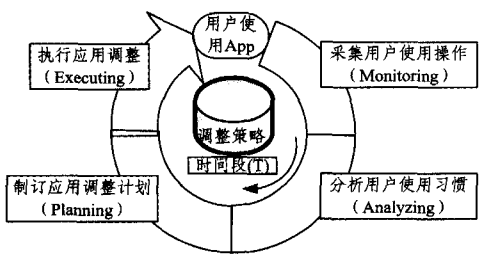

图 1　方法架构示意图

在用户使用 App 时，应用采集(Monitoring)其每次操作的数据信息；一段时间(T，由开发者在设计 App 时预定义)后，启动对采集数据的分析(Analyzing)，得出表征用户当前使用习惯的量化数据；随后，根据分析结果和预先设定好的调整策略制订应用的调整计划(Planning)，告知应用对其活动序列进行怎样的变化；最后，执行调整计划(Executing)，对应用的活动流程进行相应的更改。

3.2　具有自适应能力的安卓应用的设计需求

基于安卓应用的自适应方法架构，提出一套针对应用开发者的安卓应用设计需求，如图 2 所示。这些需求主要分为 4 个部分：策略库定义、用户使用操作数据的表示及捕获、数据分析及计划制订、自适应计划实施。开发者需在基本功能需求的基础上扩展实现自适应需求，使安卓应用具有动态调整的能力。下文将对每一项设计需求进行介绍。

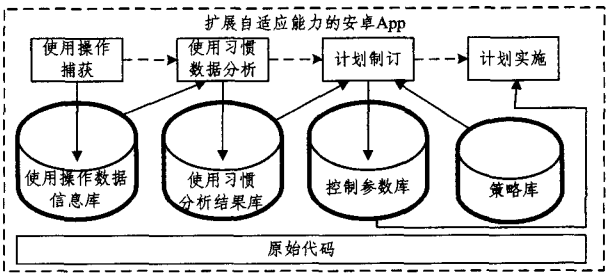

图 2　具有自适应能力的安卓应用的设计需求

3.2.1　策略库定义

策略库即安卓应用进行自适应时所要依据的策略的集合。为了定义准确、有效的策略库，需进行足够的信息调查及用户实验，并对其规律作详实准确的总结。策略库中具体策略的表示方式可以采用 ECA 规则[9]等。策略可按展现形式及其作用域进行分类，其分类和标签是为了提升计划制订子活动的检索效率。

3.2.2　用户使用操作数据的表示及捕获

针对特定的自适应场景，应该定义所需的用户使用操作数据。在开发阶段，开发者可依赖预先定义的用户操作类型列表(见表 1)在其中进行选取，以便为应用附加相应的数据采集功能。

对用户使用操作数据进行捕获时需要保证采集到的用户操作信息准确、全面和精练。在实现上其可分为构建用户操作数据库和实现用户操作数据捕获装置两部分。用户操作数据库用于储存收集到的用户操作数据，同时要具备向数据分析子活动发出分析信号并输出模块化数据的能力；用户操作数据捕获装置则需要嵌入到安卓应用的基础业务功能中，过滤并处理原始的用户行为信息，生成有效的数据。

表 1　用户操作类型列表

类型序号	操作类型描述	操作类型属性
T1	在菜单页面中选择特定菜单项，跳转至特定功能页面	(1)操作发生时间 (2)源头/目标页面(活动)
T2	点击页面上的返回按钮或者手机返回键，返回前一个页面	(1)操作发生时间 (2)源头/目标页面(活动) (3)从进入该页面直至跳出用户所持续停留的时间
T3	从页面通过任意操作跳转至其他页面	(1)操作发生时间 (2)源头/目标页面(活动) (3)从进入该页面直至跳出用户所持续停留的时间

3.2.3　数据分析及计划制订

数据分析的主要任务是完成对用户使用操作数据的分

析，提炼出用户操作数据所表现的用户使用习惯，并将用户的操作历史转化为可供决策的参数。在具体实现上其分为设计用户使用习惯分析算法、实现用户使用习惯分析器及构建用户使用习惯分析结果库这 3 个部分。用户使用习惯分析算法的设计依赖于特定的自适应场景，一般而言，算法的输入是所捕获的用户使用操作数据类型以及捕获时间周期等信息，算法的输出则是表征是否需要进行自适应调整的指标数据。我们将在后续章节通过场景介绍特定的分析算法。用户使用习惯分析器决定了数据分析任务的效率，其负责在收到数据分析信号后启动准确且尽量少的数据分析算法并输出结果，然后触发计划制订子活动。数据分析结果库则负责存储分析结果，在分析结果比较多时需要能支持模块化的读取。

计划制订的主要任务是根据通过用户使用习惯分析的结果和面向用户使用习惯的安卓应用自适应策略推荐合适的针对安卓应用活动序列的调整指令。该需求主要是进行策略匹配，旨在根据用户使用习惯分析结果库中相应指标数据的变化情况来选取相应的自适应策略，指导 App 进行自身调整。

3.2.4 自适应计划实施

计划实施需求的主要任务是完成安卓应用的各类调整，以达到改变应用用户体验的目的。在本文所关注的自适应场景中，计划的实施主要是对控制参数库中的参数进行更改，并依赖于活动管理控制器进行动态的活动序列管理。活动管理控制器需要嵌入到安卓应用的基础业务功能中，在运行时可读取控制参数库中的参数以实现不同的交互流程。

3.2.5 小结

在应用开发阶段，由于需要针对设计需求开发多个自适应相关模块，因此本文方法相比普通安卓应用开发方法需要更大的工作量。不过我们认为这些工作量是可控的。另外，这些模块的添加也能赋予安卓 App 满足用户个性化使用需求的能力，提升应用的用户体验。

4 自适应场景示例

4.1 场景 1 及其应用实现

4.1.1 场景设计

在现今大而全的安卓应用中，很多用户往往只集中于使用某几个常用的功能，此时则会出现这类场景：用户要使用某个功能时往往需要在一个具有很多按钮的菜单界面中找到相应的功能，点击菜单项后进入目标功能页面。例如，在使用美团搜索附近的餐馆时需要先点击“美食”才能查看附近的餐食信息；在使用印象笔记进行文本记录时需要先点击界面上的“加号”图标再选择“文本”后，才能开始录入文字。当用户需要频繁地使用特定的应用功能时，他们需要重复在菜单页面中选择功能菜单项的操作。针对这种情况，若应用能够感知用户在某段时间内频繁使用安卓应用特定功能的习惯并动态调整交互流程，使得应用能直接跳转至所需的功能页面，则可省去用户多余的选择步骤，提升用户体验。在实现层次上，这类场景表示基于用户选择从一个展现菜单的活动跳转到其他的对应特定业务功能的活动，自适应的目标是在应用运行阶段根据用户的操作历史推断用户的使用习惯（即对特定功能的偏好），在进入菜单活动时自动为用户选择并跳转至相应的功能活动。

4.1.2 自适应设计需求的实现

(1)策略库定义

针对该示例场景，定义的策略规则如表 2 所列，该规则采用 ECA 的方式进行描述。

表 2 使用 ECA 方式描述的自适应规则

规则	E(事件)	C(条件)	A(动作)
R1	用户在某段时间内通过页面按钮访问功能页面/返回菜单页面	存在用户对某一功能页面的偏好指数大于对其他所有功能页面的偏好指数之和	当用户进入菜单页面时，直接跳转至其偏好的功能页面
R2	用户在某段时间内通过页面按钮访问功能页面/返回菜单页面	不存在用户对某一功能页面的偏好指数大于其他所有功能页面的偏好指数之和	当用户进入菜单页面时，停留在该页面，等待用户选择

这两条规则表明安卓应用需实时监控用户的操作行为，捕获用户在某时间段内在应用中的操作轨迹，包括点击页面按钮访问特定功能页面以及在功能页面上通过点击按钮返回菜单页面的情况。通过预先设计的算法对这些操作数据进行分析，可规划出两条自适应规则。1)：规则 R1，若推断得到用户对某一个功能页面的偏好高于其他所有页面的偏好时（用户在该时间段内明显习惯使用某特定功能），则应适时更改安卓应用的活动跳转序列，使得用户进入菜单页面时应用不需要再次依赖用户的选择操作而自动跳转至相应的功能页面。2)规则 R2，若无法推断出用户对某一个功能页面的偏好高于其他所有页面的偏好（用户在该时间段内对多于一个功能页面的访问次数较为接近），则应用的活动跳转序列需要恢复至停留在菜单页面等待用户选择的交互流程。

(2)用户使用操作数据的表示及捕获

为了实现针对该场景的自适应需求，应用需要监控的用户使用操作包括表 1 中所列举的 T1 与 T2，即用户在菜单页面中选择特定菜单项跳转至特定功能页面的操作，以及用户点击页面上的返回按钮或者手机返回键从而返回至前一页面的操作。在实现上，具体做法是监听从菜单页面进入各个具体功能页面的函数，每进入一次某一功能页面就记录一次用户操作及其发生时间。同时，为了捕获用户对原先偏好的改变，即用户在新的时间段内不再习惯使用直接跳转的功能，则他会快速返回菜单页面，选择进入新的偏好页面，那么需要在进入功能页面后开始计时，如果短时间内返回了菜单页面，则记录一次立即返回动作及其发生时间。这个较短的时间需根据具体应用情况设定，本场景示例中在 5s 内返回这一指标。

(3)数据分析及计划制订

实现用户使用习惯分析的算法如算法 1 所示。通过该算法能够得到在某一个时间段内用户对各个功能菜单页面的偏好指数。对算法中的具体步骤的解释附加在相应的算法流程后。

算法 1　针对场景示例 1 的用户使用习惯分析算法

Import：用户使用操作数据信息库中所记录的用户操作信息（何时使用了什么功能）；有效时间长度 T

Output：用户对各个功能页面的偏好指数

Method UsageHabitAnalyse()

1. 定义与初始化

定义 $K_{\{1..n\}}$ 表示用户对各个功能页面的偏好指数的数组，n 为功能页面的数量。

Km（$1 \leqslant m \leqslant n$）=0

2. 读取用户操作数据

从用户使用操作数据信息库中读取有效时间 t 内的用户行为信息数据（t 视具体应用场景而定，本示例采用 10 分钟），记为 S[]。

3. 统计用户使用各个功能的频次，记录在对应的 K 值中

发生进入一次功能页面 m 的操作，该功能页面对应的 Km 值加 1；

发生一次立即返回操作，源头功能页面对应的 K 值减 1；

最终 K 数组中记录了在有效时间内用户使用对应功能的频次。

4. 考虑用户的操作分布对 K 值进行修正（根据时间越近数据取信度越高的原则对用户偏好指数 K 进行修正）

4.1. 将有效时间分为 P 段，P 的取值视具体情况而定，太大或太小会导致各段特点不突出，特点较突出的时间段的比例越高越好，一般取值应在 3 及以上，本示例取值为 3。

4.2. 使用与统计总频次相同的方法分别统计出各个时间段内的各个功能的使用频次，找出各个时间段内使用频次最高的功能 m，在其对应的 Km 值上乘上修正值 X=（1+Cp）。其中，Cp 为各时间段对应的影响力参数，p 为时间段序号且 $1 \leqslant p \leqslant P$。一般而言，$C_1 < Cp < C_P$，其取值视具体情况而定，本示例采用 $Cp = p * C_1$，即等差数列关系，而 C_1 通常与有效时间 T 及总段数 P 成反比，即 $C_1 = V/(N * T)$，V 是一个自定义的参数，其目标是为了平衡操作分布与总频次各自的影响力，总频次的影响力默认为 1，一般操作分布的影响力 Cp 应该在 10%～30%之间，V 即用来保证这一点。

5. 存储分析结果

将最终计算得到的 K 存入用户使用习惯分析结果库中。

根据算法所计算得到的用户对各个功能页面的偏好指数，进一步分析并推断出是否存在某特定的功能页面，用户对该页面的偏好指数大于对其他所有功能页面的偏好指数之和。基于该推断，从策略库中根据规则的条件定义匹配得到相应的自适应动作，从而制定自适应的计划。

（4）自适应计划的实施

本场景示例中的自适应计划的实施通过调整控制参数库中的参数加以实现，不同的控制参数告知安卓 App 在进入菜单页面时可立即跳转的目标功能页面。若该参数为 0（满足 R2 规则后），则 App 能够停留在菜单页面。控制该 App 活动跳转序列的代码如下所示，其中 switchFunc 方法控制活动的跳转。

```
@Override
protected void onCreate(Bundle savedInstanceState){
  super. onCreate(savedInstanceState);
  setContentView(R. layout. myMenu);
  inforDB=new InforDB(this);
  dbWriter=inforDB. getWritableDatabase();
  switchFunc();
  initView();
}
private void switchFunc(){
  controlDB=new ControlDB(this);
  dbReader=controlDB. getReadableDatabase();
  cursor=dbReader. query(ControlDB. TABLE_NAME, null, null,
  null, null, null, null);
  //check controlDB and jump
  switch (cursor. getString (cursor. getColumnIndex (ControlDB.
  WHI-TCH)){
    case "0":
        break;
    case "1":
        jumpToFunc1();
        break;
    case "2":
        jumpToFunc2();
        break;
    case "3":
        jumpToFunc3();
        break;
    default:
        throw new Exception(
        "Something Wrong in ControlDB!");
  }
}
//jump and record
private void jumpToFunc1(){
  addToInforDB("1");
  i=new Intent(Menu. this, Func1. class);
  startActivity(i);
}
private void jumpToFunc2(){
  addToInforDB("2");
  i=new Intent(Menu. this, Func2. class);
  startActivity(i);
}
private void jumpToFunc3(){
  addToInforDB("3");
  i=new Intent(Menu. this, Func3. class);
  startActivity(i);
}
```

4.2　场景 2 及其应用实现

4.2.1　场景设计

存在一种类型的 App 在用户使用过程中通过一个过渡页面展示附加的或补充的信息，例如，涉及支付的支付成功页面、涉及主动记录的预览页面（记事本、相机等）等。这类过渡页面对于部分用户是需要的，对于部分用户则是无关紧要的。若能根据用户的使用习惯对不同需求的用户采取不同的策略，即应用对不关注该过渡页面的用户直接跳过该页面，则可

在一定程度上提高用户的交互效率。本文将这类场景抽象为在功能页面跳转流程中具有可选但非必需页面(过渡页面)的情况,自适应的目标是在应用运行期间根据用户在该页面的停留时间决定是否跳过该过渡页面并自动跳转至后续的功能页面。

4.2.2 自适应设计需求的实现

对该场景示例自适应设计需求的实现与前一个场景的实现较为类似,但在数据分析算法、自适应策略等方面略有不同。

首先,自适应规则指明需要记录的是用户在过渡页面停留的时间以及用户在其他页面返回至该过渡页面的操作,以确定用户是否需要展示该页面。停留时间过短的次数越多说明用户越倾向于不需要这个流程,返回到该页面的操作越多说明用户越倾向于需要这个流程。在具体规则上,当不需要这一过渡页面的偏好指数占绝对优势(即是需要这一页面的偏好指数的 3 倍及以上)时,通过控制参数变更页面的跳转序列,使得应用能够直接跳转至后续的页面。

其次,针对该场景的分析算法,采用对有效时间内一定次数的样本进行分析的方式,即每次获取最近一定数量的操作数据记录,通过分布修正偏好指数时的分段是依据数量而不是时间来分的,即每组含有操作的数量是一定的,时间跨度则可能每次都不一样,但修正方法类似。采用这种分析算法的目的是在实验中尝试更多类型的分析方式。

最后,与前一个场景中涵盖多个功能页面的情况不同,本场景下需要计算分析偏好指数的对象只有两个,即需要此过渡页面亦或不需要,这使分析算法的效率相对较高。

4.3 自适应效果分析

上文描述了如何根据设计需求解决实际场景中的问题,那么其效果如何呢?由于这里尚不能对用户体验进行量化表征,因此我们只能通过其他方面从侧面表现实验前后场景中用户体验的变化,下面以场景 1 为例。

对于场景 1,假设在加入自适应组件之前某一用户每天会使用功能 A 大约 20 次,其他功能 5 次,每次使用功能时他需要先在菜单中找到对应按钮,点击后启动,那么他一天中需寻找按钮并点击 25 次;而在加入自适应组件之后,该用户需寻找按钮并点击(包括启动功能和返回菜单两种)8～16 次,平均可以减少一半的操作量。用户操作量的减少意味着用户可以更"懒"、更舒服,安卓应用的用户体验则会更好。

同时,按照用户的使用习惯进行自动跳转可能会帮助用户在某些需要快速使用相应功能的情况下节约寻找及点击按钮的时间,这个时间虽然很短,但在相对紧急的情况下其重要性会被放大很多,相应地,省去这一步骤对安卓应用的用户体验也会有很大提升。这里可以称之为关键点效应,即关键时刻的表现往往能使人感受更深刻。

结束语 用户使用安卓 App 时会在一定的时间段内形成其个性化的使用习惯,根据特定的习惯更改应用中的页面(活动)跳转序列能够更好地契合用户的操作流程,提高应用的用户体验。为了实现该目标,本文提出了一种面向用户使用习惯的安卓应用自适应方法,并归纳了一组自适应设计需求,应用开发者需要在构造应用时附加实现这些需求,从而赋予应用在运行时进行自动调整的能力。

本文通过两个场景示例列举了对设计需求的实现细节。针对这两个场景的实验结果表明在这些场景下对用户操作习惯的自适应均能实现,在用户首次使用和改变用户操作习惯后应用都能及时作出相应的调整。本方法在实验中也体现出多方面的缺陷,其中最重要的是对应用的分析效率不够高,存在分析延迟等问题。另外,本文研究仍处于初步探索阶段,所关注的用户使用习惯仍较为简单,同时还未能抽象出通用的自适应机制以满足各类自适应调整的需求。未来的工作将以解决这些问题为主要目标。

参 考 文 献

[1] WANG Q X, SHEN J R, MEI H. An introduction to self-adaptive software[J]. Computer Science, 2004, 31(10): 168-171. (in Chinese)
王千祥,申峻嵘,梅宏. 自适应软件初探[J]. 计算机科学,2004,31(10):168-171.

[2] KEPHART J O, CHESS D M. The vision of autonomic computing[J]. Computer, 2003, 36(1): 41-50.

[3] DEHLINGER J, DIXON J. Mobile application software engineering: Challenges and research directions[M]// Workshop on Mobile Software Engineering. 2011.

[4] MCKINLEY P K, SADJADI S M, KASTEN E P, et al. Composing adaptive software[J]. Computer, 2004, 37(7): 56-64.

[5] NAKAGAWA H, KUDO T, SEI Y, et al. Towards Software Evolution for Embedded Systems Based on MAPE Loop Encapsulation[C]// 2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems (SASO). IEEE, 2014: 203-204.

[6] CUGOLA G, GHEZZI C, PINTO L S, et al. Adaptive service-oriented mobile applications: A declarative approach[M]// Service-Oriented Computing. Springer Berlin Heidelberg, 2012: 607-614.

[7] CUGOLA G, GHEZZI C, PINTO L S, et al. SelfMotion: a declarative language for adaptive service-oriented mobile apps[C]// Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. ACM, 2012.

[8] DATTA S K, BONNET C, NIKAEIN N. Self-adaptive battery and context aware mobile application development[C]// 2014 International Wireless Communications and Mobile Computing Conference (IWCMC). IEEE, 2014: 761-766.

[9] DITTRICH K R, GATZIU S, GEPPERT A. The Active Database Management System Manifesto: A Rulebase of ADBMS Features[C]// LNCS 985. Springer, 1995: 3-20.