

一种上下文感知编程模型——IPM 的实施

李 明^{1,2}, 郝 旻², 牛玉洁², 赵万贺²

(1 西安理工大学 经济与管理学院, 陕西 西安 710054; 2 西安交通大学 电子与信息工程学院, 陕西 西安 710049)

摘 要: 提出了一种面向上下文的编程模型 IPM(Isotope Programming Model)并介绍了 IPM 的实施. IPM 对已有的面向对象模型进行扩展, 引入了新的语言元素, 将上下文感知应用的编程对程序员简化, 使其只关心程序的逻辑, 而不用关心上下文数据的变化. 将 IPM 与现有的面向对象的编程方法进行对比, 体现了 IPM 在上下文感知应用开发中的优越性.

关键词: IPM 的实施; 上下文感知应用; 面向对象

中图分类号: TP311.11

文献标识码: A

文章编号: 1000-7180(2011)03-0170-05

The Implementation of IPM —— a Kind of Programming Model of Context—Aware Application

LI Ming^{1,2}, XI Min², NIU Yu-jie², ZHAO Wan-he²

(1 School of Economy and Management, Xi'an University of Technology, Xi'an 710054, China;

2 School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: The paper proposes a context—oriented programming model which is named Isotope Programming Model (IPM). The paper also discusses the implementation of IPM. IPM extends the object—oriented programming model and introduces new language elements, which simplifies the development of context—aware application and makes programmer only care about the logic of programming and free from the change of context data. The paper also compares the IPM with object—oriented programming model and embodies the advantages of IPM in the fields of the development of context—aware application.

Key words: implementation of IPM; context—aware application; object—oriented

1 引言

普适计算已逐步渗透到人们的生活中, 其开放性、移动性、环境多变的特征更加凸现出来. 传统的编程模型中环境因素与程序的控制主体的关系过分紧密, 不能适用于普适计算环境的多样、复杂、动态和多变等特点, 更难于进行系统的维护和扩展, 所以不能很好地适应普适计算环境. 例如, 面向对象编程中的方法与上下文紧密相关, 同一种操作根据不同的上下文的实现形式是不同的, 在面向对象中主要通过传递和上下文紧密相关的参数, 并编写大量的分支语句完成. 这些做法使程序的逻辑和上下文数

据紧耦合, 不利于程序的扩展与维护, 不适合上下文感知应用的开发.

研究人员已经认识到基于普适计算环境的程序设计时支撑技术的重要性, 展开了相关的研究, 并建立了相关的编程模型. 如: 分支模型(branching model)和触发模型(triggering model)是普适计算中常用的两类编程模型. 分支模型将上下文以及用户偏好相关的决策嵌入到应用程序逻辑中, 例如, Henricksen, K. 和 Indulska, J 提出将用户偏好加入到分支模型中^[1], 使用用户的偏好将上下文和可选动作联系起来. P. J. Brown and G. J. F. Jones 使用分支模型对上下文信息检索^[2]. 触发模型经常

收稿日期: 2010-06-13; 修回日期: 2010-08-11

基金项目: 国家“八六三”计划项目(2006AA01Z101)

使用在自适应的、上下文感知应用中,该模型随着上下文的改变或某一具体的事件出现自动触发行为动作,如 stick-e note 使用了此种编程模型^[3],Ingar Mæhlum Arntzen 提出的基于事件驱动的普适计算编程模型^[4],将用户的偏好与事件驱动结合起来,从而适应不同的上下文情况.这两类编程模型使上下文情况、用户的需求与程序逻辑紧密绑定,当变化产生时,程序不易扩展,升级,从而不能适应动态的环境.

IPM(Isotope Programming Model)是项目组提出的上下文感知的编程模型,该模型在面向对象的基础上对对象的概念进行扩展,将对象从属性和行为的集合转变为属性和一系列针对不同环境的行为代码文件的集合,每一个代码文件包含一段环境描述信息及在该环境下对象所呈现的行为,这种行为的分离可以提高程序开发的灵活性,降低环境变化对程序的影响程度. IPM 的实施是以面向对象模型为基础,对上下文感知应用方面进行扩展,添加新的语言成分,使程序员在开发上下文感知应用时,只需要关心应用的抽象操作,不必考虑当时的上下文情况.而抽象操作如何在不同的上下文中映射成具体的操作,事先由规则定义好,即在何种上下文的情况执行何种操作,后台根据上下文的情况根据规则自动匹配具体的操作. IPM 降低了程序逻辑与上下文数据的耦合性,比起传统的编程方法更利于程序的扩展及上下文感知应用的开发.

2 IPM 简介

IPM 是项目组提出的一种面向普适计算环境的编程模型. IPM 的核心思想在于将对象的代码从一个类文件扩展到多个类文件,每个文件表示一种环境对应的执行过程,多个类文件共存于系统中,共同提供服务.这种模型所表示的对象类似于化学元素中的同位素:占据相同的周期表位置,具有类似的功能,但具有不同的形式,这种模型称为 IPM. IPM 扩展了面向对象的概念,在 IPM 中,对象不仅包含属性的声明,同时包含一组同位素元素(isotope element). 每一个同位素元素代表对象在一种特定环境下的行为.对象在表示其不同环境中的行为时,不需要将所有行为混合在一个对象的声明中,取而代之的是若干个独立的同位素元素,对象的声明过程也简化为只包含属性的声明. IPM 的主要特征:

(1) 对象包含对象属性的声明以及行为的声明.行为声明部分和对象属性声明部分分离.定义属

性部分称为 main element.

(2) 每个对象包含多个行为描述部分,称之为 isotope element,每个部分代表一个特定环境下对象所呈现的行为.

(3) 每个行为描述部分具有一个上下文声明块 context block,每一个 context block 由一组上下文信息条件表达式组成,只有当前环境完全满足 context block 所声明的条件时,行为描述部分才有可能生效.

图 1 展示了面向对象模型和 IPM 中对象的基本结构.这是一种基于分离/匹配机制的编程模型.分离指:将程序中的与动态环境相关的因素从程序主体中分离出来,程序的主体只描述应用的逻辑,在编程时集中在程序的逻辑主体,而与动态环境相关的信息使用不同的文件描述,这样当环境改变时,可以只改动与环境相关的文件的描述,从而最大限度降低环境信息与处理代码的耦合度,提高程序的灵活性和适应性.匹配指,高度抽象的程序主体逻辑在不同的环境实施时,按照一定的规则、上下文情况以及环境资源的情况与具体的实体和服务匹配,形成可执行代码.

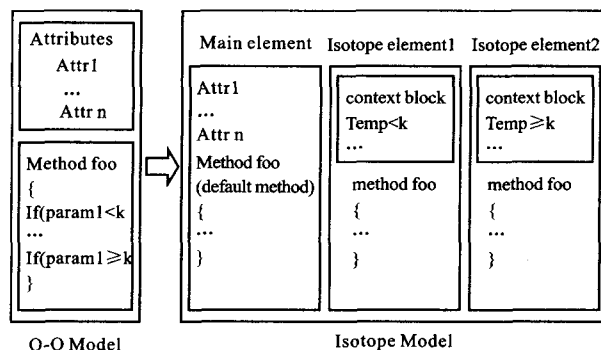


图1 面向对象模型和 IPM 的对比

3 IPM 的实施

IPM 的实施是以面向对象模型为基础,针对上下文感知应用方面进行扩展,添加新的语言成分,使程序员在开发上下文感知应用时,只需要关心应用的抽象操作,不必考虑当时的上下文情况.而抽象操作如何在不同的上下文中映射成具体的操作,事先由规则定义好,即在何种上下文的情况执行何种操作,后台根据上下文的情况根据规则自动匹配具体的操作.

3.1 IPM 对面向对象的语言扩展及实施

在开发上下文感知的应用时,面向对象的类中方法与上下文紧密相关,即同一种操作根据不同的

上下文的实现形式是不同的,在面向对象中主要通过传递和上下文紧密相关的参数,并编写大量的分支语句完成.这些做法使程序的逻辑和上下文数据紧耦合,不利于程序的扩展,例如某个方法中与上下文相关的参数增加了,所有调用此方法的实参都需要增加,而且需要修改获取上下文的相关代码以获取相关参数的值.

在 IPM 开发中,在程序中会同时存在上下文相关的和上下文无关的类.上下文无关的类定义和目标语言中普通类定义没有区别.上下文相关类的主体定义时,要在声明的最前面使用关键字 isotope 标识.一旦类被定义为 isotope,那么说明该类的对象中包含 isotope 方法,其使用关键字 isp 标识. isotope 方法指与上下文相关的抽象操作,而所有与上下文相关的具体操作称为 isotope element,在程序执行时,根据不同的上下文的情况将 isotope 方法映射为不同 isotope element,即根据不同的上下文执行不同的操作. IPM 在面向对象的基础上,对上下文相关的方法进行语言扩展,使其支持动态上下文切换与获取,通过预编译技术实现扩展语言向宿主语言的转换. isotope 方法表示如下所示:

```
isp retType MethodName(paraList): context-SubjectList
```

```
{ ... (Default body of the method) }
```

isp: 定义 isotope 方法的标记;

retType: 方法返回值的数据类型;

MethodName: 方法名称;

paraList: 业务参数列表(与上下文无关的参数);

contextSubjectList: 给出该方法敏感的上下文主题列表: 如设备属性、位置、用户偏好等;

Default body of the method: 描述一种缺省的行为方式.

以国际会议为例,参会人员事先注册个人信息,其中包括国籍和专业偏好,当参会人员进入会场时,会议系统根据其母语及其专业偏好向其手机发送感兴趣的专题会议的安排.我们使用 IPM 完成以上的上下文感知应用.图 2 中,左边的程序代码是一个包含有上下文相关方法的 isotope 类 CellPhone.java,其中,上下文相关方法 isp Display(): language, preference 表示会议系统根据上下文,即参会人员的母语(language)及专业偏好(preference),向其发送感兴趣的会议安排.这是一个抽象操作,需要在具

体的上下文环境执行具体的动作.图 2 的右边程序代码描述了上下文感知方法 Display() 的调用,在调用时,程序员不需要考虑上下文情况,直接调用 isotope 方法,如何进行上下文的匹配,以及调用哪个 isotope element,交给 IPM 的软件体系框架完成.图 3 列出所有与 Display() 相关的 isotope elements.规则文件(图 4 所示)定义了映射规则,使用本体语言 OWL 描述,将抽象操作与具体的 isotope elements 根据不同的上下文进行匹配.程序运行之前,首先对 isotope 类预编译,转换成 java 程序.在执行 O. Display() 语句时,先获取相关上下文,然后查询规则文件,得出相关的 isotope element 并调用.

<pre>isotope Cellphone() { Attribute1; Attribute2; ... Cellphone(); isp Display():language, preference; }</pre>	<pre>... Cellphone O=new... O.Display()</pre>
---	---

图 2 isotope 类及 isotope 方法的调用

<pre>void Display1() { printf("上午在102会议室SOA专题"); } void Display2() { printf("OS session in Room 213 at 9:00 a.m"); }</pre>

图 3 与 Display() 相关的所有 isotope elements

<pre><Display rdf:ID="Dispaly1"> <followedBy> <Person rdf:ID="Person1"> <Preference>SOA</preference> <language>Chinese</language> </Person> </followedBy> <actionName>Display1</actionName> </Display></pre>	<pre><Display rdf:ID="Dispaly2"> <followedBy> <Person rdf:ID="Person2"> <Preference>OS</preference> <language>Chinese</language> </Person> </followedBy> <actionName>Display2</actionName> </Display>.....</pre>
--	--

图 4 规则文件

3.2 IPM 的软件体系框架

IPM 的软件体系框架主要描述如何将 isotope 方法表示的与上下文无关的抽象操作映射到在特定上下文环境中具体操作,即将 isotope 方法与 isotope element 映射.图 5 展示了 IPM 软件体系框架的类图,其中,contextServiceImpl 为核心类,类似于匹配器,主要根据当前的上下文查询规则文件,得出匹配的方法名,然后使用反射机制调用.每一个上下文相关的抽象操作都对应一个监听器 Listener.

Monitor 是用来获取当前上下文的类,不同的 Monitor 可以获取不同的上下文,如位置上下文、设备上下文、用户上下文等,Listener 对应的抽象操作对哪一类上下文感兴趣,就可以将自己注册到相应的

Monitor,当感兴趣的上下文发生变化时,Monitor 将这一变化包装成一个 ContextEvent,通知给注册的 Listener,Listener 将根据这一变化调用 context-ServiceImpl 中的相关方法完成匹配和方法调用。

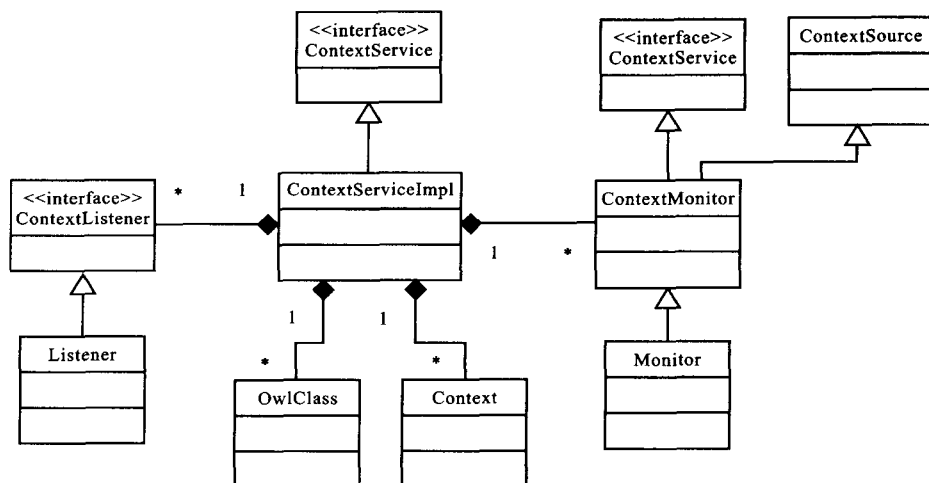


图5 IPM 软件体系框架类图

4 面向对象和 IPM 开发模式的比较

使用面向对象的编程模式,程序员可以按照比较自然的思维方式组织应用系统.在面向对象编程模式中,一个实体采用一个对象表示,对象是实体属性和方法的集合.属性用于表示对象的当前状态,方法则表示这个对象能够执行的行为.对象可以接收其他对象对它的调用,通过方法调用改变对象内部状态或者为调用者提供服务.传统的面向对象设计主要基于互联网环境,在这种环境中,程序执行的上下文相对比较稳定,可能的变化也是可以预知的,因此,面向对象模型的设计没有提供适应上下文变化的能力.

使用面向对象模型编写上下文感知应用存在如下两个问题:

(1) 环境的改变会引起应用程序的重新编写、编译、部署和运行.现有的面向对象的方式采用分支语句进行环境代码的分离,如果环境发生变化,需要增加或修改分支的执行过程以及判断的依据,另一方面,环境的变化可能引起执行过程的变化,对程序的改动程度会更大.重新编写的代码需要再次经过编译、部署的过程才能生效,增加了更新系统的复杂度.

(2) 通过参数方式传递环境信息导致程序的适应性较差.面向对象系统在设计时需要明确定义接口及程序交互方式,如果有新的环境引入系统,则需要通过引入新的参数,同时改变程序的接口来适应这种变化.由于对象之间通过接口交互,因此接口的

变化会引起一系列的改动,从而增加修改的复杂度.

产生这些问题的重要原因是:面向对象系统没有提供将不同环境的不同处理行为分离出程序主体的方法,导致环境信息的改变直接影响程序代码,从而降低了程序的适应能力.

IPM 编程模式采用了逻辑代码与环境代码分离的模式.最大限度降低环境信息与处理代码的耦合度,提高程序的灵活性和适应性.对于普适环境中泛在设备的动态性、异构性以及环境的复杂性和多变性方面都有较好的适应.

普适计算环境中,上下文的改变是非常有可能发生的,这种变化包括上下文空间的变化(上下文维度的增加、减少、修改)、上下文值的变化引起逻辑的变化、上下文与行为间映射关系的改变等.在这些情况下,对于传统的编程模式来说,环境依赖代码和逻辑代码是混杂在一起的.虽然对于上下文的变化我们只想修改环境依赖代码,但是这种混杂将迫使我们不得不先去分析哪些是环境依赖代码,哪些是逻辑代码,这就需要程序修改者了解逻辑代码,而逻辑代码往往最复杂,这加重了程序员的负担.但对于 IPM 来说,由于逻辑代码和环境依赖代码分离,程序员可以迅速找到环境依赖代码,对其作出调整,系统的业务逻辑层代码则完全不需要了解.当有新的上下文或行为加入系统时,程序员只需要新添加一个 isotope element 来描述新加入的行为,同时修改配置文件即可;当某一个上下文环境对应的业务逻辑

辑发生改变时,程序员只需要编写一个新的 isotope element,用它来替换旧的 isotope element,系统甚至不需要重启;当仅仅映射关系发生改变时,程序员只需要修改规则文件即可,其他的一切都不用改变。由此可见,对于上下文频频发生变化的普适计算环境下,IPM 的适应性是极高的,它为开发人员减轻了大量的负担,频繁的上下文变更仅需要程序员一个简单的动作即可以完成,整个系统不需要二次开发、重新部署等,代码的维护量大大减少。

IPM 消除了程序中冗繁的条件分支语句,程序员不再需要用大量的分支语句来对上下文环境和方法之间的匹配进行关联,这些关联都由规则文件来完成,易于书写,也易于修改,不仅从结构上更加合理,更加清楚明晰,而且使得程序各模块各部分之间解耦,相互之间的依赖性降低。

综上所述,IPM 较之传统语言在普适计算编程环境中拥有很大的优势,对于普适环境中泛在设备的动态性、异构性以及环境的复杂性和多变性方面都有较好的适应。

5 结束语

文中主要介绍了 IPM 的实施,IPM 的实施是以面向对象模型为基础,针对上下文感知应用方面进行扩展,添加新的语言成分,使程序员在开发上下文感知应用时,只需要关心应用的抽象操作,不必考虑

当时的上下文情况。IPM 使上下文数据与业务逻辑松耦合,比起传统的面向对象编程,更利于上下文感知应用的开发。在以后的工作中,需要对 IPM 的支撑技术如上下文信息的管理、推理、聚合等方面加大研究力度。

参考文献:

- [1] Henricksen K, Indulska J. A software engineering framework for context-aware pervasive computing[C]// Proceedings of the Second IEEE Annual Conference, USA; IEEE, 2004:77-86.
- [2] Brown P J, Jones G J F. Context-aware retrieval: exploring a new environment for information retrieval and information filtering[J]. Personal and Ubiquitous Computing, 2001,5(4):253-263.
- [3] Pascoe J. The stick note architecture: extending the interface beyond the user[C]// Proceedings of International Conference on Intelligent User Interfaces, Orlando, USA; ACM, 1997:261-264.
- [4] Ingar Mæhlum Arntzen. A programmable structure for pervasive computing[C]// Proceedings of the IEEE/ACS International Conference on Pervasive Services, San Diego; IEEE, 2004:29-38.

作者简介:

李明女,(1973-),博士研究生,讲师。研究方向为移动计算、上下文感知。

(上接第 169 页)

像素和部分剪切边界像素所造成的另一个非剪切像素;如反过来,先像素值加密后像素位置变换,则不能得到正确恢复的像素可认为是剪切像素的两倍,图像恢复质量将很差。这是应用文中算法要注意的关键问题。

6 结束语

文中算法有如下特点:(1) 采用以块为单位的类似 CBC 模式加密图像像素,并可随机改变每一块图像的加密初始值,使算法安全性大为提高;(2) 在像素置乱均布和像素值均布两大前提基础上,提出了基于邻域相邻像素特性的加密图像抗剪切攻击恢复算法,并有效实现。

基于邻域相邻像素特性的解密图像的抗剪切恢复算法,随着被剪切面积的增加,被剪切像素点的还原准确性逐渐下降,需进一步改进。

参考文献:

- [1] 李太勇,贾华丁,吴江. 基于三维混沌序列的数字图像加密算法[J]. 计算机应用,2006,26(7):1652-1654.
- [2] 成亚萍,吕巍,张太忠. 一种新的抗剪切的数字图像加密算法[J]. 计算机工程与应用,2008,44(27):130-131.
- [3] 张雪锋,范九伦,康宝生. 一种抗剪切变换攻击的数字图像加密算法[J]. 系统仿真学报,2006,18(8):2260-2263.
- [4] 李鹏,田东平. 基于超混沌序列的数字图像加密算法[J]. 微电子学与计算机,2008,25(3):4-7.
- [5] 谢颢,范九伦. 一种基于位运算的数字图像加密算法[J]. 微电子学与计算机,2009,26(12):117-119.
- [6] 罗军. 基于混沌系统的数字图像加密算法[J]. 计算机工程与设计,2009,30(8):1844-1845.

作者简介:

廖琪男男,(1964-),研究生,高级工程师。研究方向为数字图像处理、数字图像加密、数字图像信息隐藏和数字水印。