两种基于分离/匹配机制的普适计算编程模型比较与研究*)

李 明[1,2] 齐 勇[1] 侯 迪[1] 郗 旻[1]

(西安交通大学电子与信息工程学院 西安 710049)[1] (西安理工大学信息管理系 西安 710054)[2]

摘 要 普适计算的环境是高度动态的,资源种类丰富,设备交互复杂,而基于封闭环境的传统的编程模型已不能适应这种变化。因此,基于普适计算环境的编程模型已成为软件工程领域重要的研究内容之一。就目前两种基于分离/匹配机制的普适计算编程模型——基于本体及操作符的编程模型(Olympus)及基于 XML 及动态代码嵌入的编程模型(COP(Context-Oriented Programming))进行比较与研究,分析了两者的共性及各自的特点、优势及局限性,并对今后的研究进行展望。

关键词 普适计算,编程模型,分离/匹配,Olympus,COP

Comparison and Research of Programming Model Based on Separating/Matching Mechanism for Pervasive Computing

LI Ming[1,2] QI Yong[1] HOU Di[1] XI Min[1]

(School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)[1]

(Department of Information Management, Xi'an University of Technology, Xi'an 710054, China)[2]

Abstract The environments in pervasive computing are highly dynamic. The resources in these environments are rich and interoperability of equipments is very complicated. Traditional programming models based on closed environments cannot adapt to these changes. Therefore, programming model for pervasive computing has become essential in software engineering. Compared the two programming models based on separating/matching mechanism for pervasive computing——Olympus which is based on ontology and operators and COP (Context-Oriented Programming) which is based on XML and embedding dynamic codes. Analyzed the advantages and limitations of the two programming models and proposed the future works of this field.

Keywords Pervasive computing, Programming model, Separating/matching, Olympus, COP

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。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,故对其展开了相关的研究,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与用户偏好相关的决策嵌入到应用程序逻辑中,例如 K Henricksen 和 J Indulska 提出将用户偏好加入到分支模型中[2],使用用户的偏好将上下文和可选动作联系起来,P. J. Brown 和 G. J. F. Jones 使用分支模型对上下文信息进行检索[3]。触发模型(triggering model)经常使用在自适应的、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[4],Ingar Mæhlum Arntzen 提出的基于事件驱动的普适计算编程模型[11],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。近年来,基于普适计算环境编程模式研究有了新的进展,主要集中在使支撑普适计算的平台更好适应动态环境,并产生了一种新型普适计算的编程模型: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,从而最大限度地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[1]。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、环境资源的情况与具体的实体和服务相匹配,形成可执行的代码。这种编程模型中比较成熟的有:使用本体及操作符的 Olympus [5] 和基于 XML 及动态代码嵌入 COP(context-oriented programming)[6,7],它们从不同的角度实现了基于分离/匹配机制的普适计算编程模型。本文将集中讨论与比较这两种编程模型的共性、特征、优势及局限性。论文的以下部分是这样安排的:第 2 部分介绍 Olympus 及 COP 编程模型;第 3 部分分析 Olympus 编程模型及 COP 编程模型的优势及局限性;最后是展望。

*)国家 863 计划基金资助项目(2006AA01Z101),西安理工大学校科学研究计划资助项目(107-210509)。李 明 讲师,博士研究生,研究方向为分布式系统、普适计算;齐 勇 教授,博士生导师,研究方向为分布式系统、移动计算、普适计算。

2 Olympus 编程模型及 COP 编程模型

2.1 Olympus 编程模型

UIUC 大学的 Anand Ranganathan 提出基于普适计算的编程模型——Olympus[5] 模型，以 Gaia[8] 为基础，并对 Gaia 的编程模式进行了扩展。Olympus 使开发者使用“活动空间(active space)”中的抽象实体(abstract entities)及其中的通用操作(common operators)，描述程序的主体逻辑。在程序实施时，Olympus 通过发现机制，根据开发者定义的约束文件、实体的本体描述、当前空间可获得的资源、空间级的规则、上下文环境及效用函数将这些抽象实体解析为活动空间中实际的实体。此外，Olympus 给开发者提供了 operators，表示活动空间中的通用功能，它包括开始、结束、移动部件等。实施时，operators 映射为具体函数。因此，开发者不必考虑空间中上下文情况、环境的资源、程序是怎样实施的，所有这些细节都是由 Olympus 自动完成，这样可使开发者专注于编程中的逻辑设计，使程序更好地适应普适计算的高度动态环境。图 1 是 Olympus 的程序的一个简单的例子。

```
ActiveSpace as1; / * refers to a virtual active space entity * /
as1. hasProp("containsPerson","Sal");
as1. instantiate( ); / * as1 now refers to the Active Space where the
                        user Sal is located * /
Application appl;/ * refers to a virtual application entity *
appl. hasProp("class","Slideshow");
appl. hasProp("file","Olympus. ppt");
appl. start(as1); / * appl is started in Active Space as1 in the "best"
                    possible configuration * /
```

图 1 Olympus 程序例子[5]

Olympus 的特点如下：

(1)在 Olympus 中引入了 Active Spaces 概念。Active Spaces 指某一个具体的、高度动态的物理空间，其空间中的上下文及资源是经常变化的。Active Spaces 的资源有设备、对象(object)、用户(user)、服务(service)及应用(application)，Gaia 对此空间中的所有资源进行管理。

(2)引入本体的思想。针对每一个活动空间，设计本体层次，抽象地描述 Active Spaces 实体，使用 Olympus 中的发现机制，找到特定的空间中与此抽象实体相匹配的具体实例(instance)。

(3)引入通用操作符的概念(common operators)。使用通用的操作符代表常用的动作及行为，在运行时根据上下文的情况将操作符映射到相应的函数。

(4)Olympus 提出程序主体逻辑主要由抽象实体及抽象操作构成。Olympus 使开发者只需对“活动空间(active space)”中的抽象实体及抽象操作进行描述，其目的让程序员只需要关心程序的逻辑，不需要关心具体的资源，上下文情况，用户的偏好。在实施时，程序智能地解析成具体 Active Spaces 的应用及服务。

(5)使用发现机制(discovering process)和效用函数(utility function)。使用发现机制(discovering process)使程序框架中抽象实体与 Active Spaces 中的具体的实体匹配；如果同时发现多个实体可以匹配，则使用效用函数(utility function)来找到最优的匹配实体。

2.2 COP 编程模型

Andry Rakotonirainy 提出了面向上下文的编程模型 COP(Context Oriented Programming)[6,7]，其程序由主体框架 skeleton 及与上下文相关的 open term 构成。Open term 使用 XML 代码描述目的(goals)及上下文的含义。COP 使用 context-filling operation 根据上下文的情况及 open term 的必要描述，从候选库(repository of candidates)中选择 stubs，动态地填写程序 skeleton 中 open term。context-filling operation 分为两个阶段，第一阶段进行匹配(matching)，从候选库中选择合适的 stubs 填写 open term；第二个阶段进行绑定(binding)，将 stubs 中的参数与 skeleton 中的变量绑定，从而使代码可以执行。Context Oriented Programming 可以使编程人员不用定义和管理上下文及相关的适应机制，增强了程序的适应性及扩展性。图 2 为 COP 的 context-filling 的例子。COP 编程模型的特点为：

(1)建立一个与上下文无关的程序框架(context-free skeleton)，这个程序框架主要体现了整个程序的逻辑。

(2)以 XML 的形式描述与上下文及动态的环境信息相关的 open term。

(3)context-filling 属于运行时的动态代码嵌入，并分为两个阶段：第一阶段，匹配(matching)，从 stubs 库中选择具体的 stubs 填充 open-term；第二阶段，绑定(binging)，实质就是将 stubs 的 parameters 与程序主体(program skeleton)中的参数绑定，使程序可以运行。

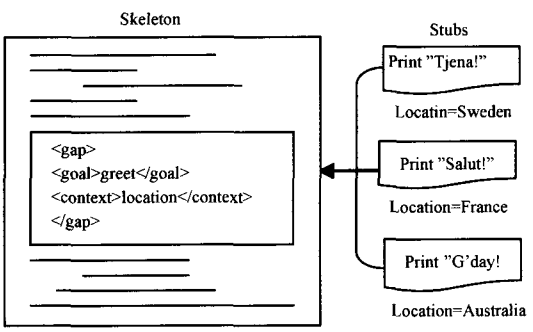

图 2 context-filling 例子[6]

2.3 两种编程模型的分离/匹配实施

(1)分离

Olympus 使用 C++描述程序主体逻辑，COP 使用 python 描述程序主体逻辑(sketlon)。在 Olympus 中，动态的环境资源、用户的偏好信息以及上下文情况使用约束、规则、实体的本体层次(ontology hierarchy)及效用函数描述。在 COP 中，使用 XML 语言创建 open term，描述动态的环境资源和上下文信息。

(2)匹配

在 Olympus 中，程序主体中抽象的实体通过发现机制映射成动态环境中的具体实体；抽象行为根据上下文被解析成具体操作函数。在 COP 中，运行时，stubs 需要与 open term 进行匹配，并绑定相关的参数。

3 两种编程模型的优势及局限性

3.1 优势

两种编程模式旨在使程序的逻辑更抽象化，使程序的逻辑和具体的环境资源、上下文情况脱离，其目的是使程序适应性(adaptable)、可移植性(portable)大大提高，并且使程序员能专注于程序的逻辑。这是这两种编程模式的共同优点，此外两种编程模型还具有各自的优点。

Olympus：

(1)对 OOP 思想的扩展及提升，使程序更加灵活(flexible)。在 OOP 中，程序的逻辑主要由对象及对象实施的动作构成。而对象及其发出的动作是具体的，降低了应用程序的自适应、灵活性及可移植性。在 Olympus 中，在程序主体逻辑中由抽象的实体(entities)及抽象的操作(operations)构成，脱离了具体的细节。程序主体逻辑在不同的活动空间(active space)呈现出的具体的表现形式是不同的，这依赖于匹配过程中的上下文情况、所定义的本体层次及实体的描述、开发人员制定的约束以及效用函数，从而使程序的灵活性、适应性大幅度提高。

(2)使应用程序智能化。引入人工智能的方法，使用 prolog 语言描述约束规则，并能基于上下文的情况进行推理(context-sensitive)。例如，当用户在开会时，不会使用有声音的提醒设备来通知用户，可以使用震动或闪烁的方式提醒。

(3)匹配精确度的量化。在有多个候选实体的实例可以匹配时，Olympus 引入多维的效用函数(multi-dimensional utility function)来选择最优的实例。基本的四个维度包括：Location of the entity, Tasks supported by the entity, state of the entity, context of the space。一些维度可以被量化(如 location of the entity)，另外一些维度只可被定性(例如基于上下文的用户偏好)，定性的维度在用户的策略中可以用偏序的形式描述。此外，约束规则给出不同环境中的各种实体匹配的精确度要求，以提高应用程序的实施效率。

COP：

(1)高度抽象与多样性的统一。首先对同一类型程序的功能进行高度抽象，以 XML 的格式体现在程序主体中，将它们所表现出来的多样性存放在 stubs 库中。在程序运行时，根据不同的上下文匹配并绑定特定的程序功能。

(2)使用 XML 表示抽象的上下文及程序的功能，简单易操作、可读性强。

(3)实施程序的难度较低。使用成熟的脚本语言 Python 及匹配算法 tree-to-tree(t2t)进行，使程序员及程序管理者易于掌握此种编程模型，从而使程序实施的难度降低。

3.2 局限性

两者都没有说明如何解决程序实施过程中服务的冲突，尤其没有涉及不同设备上的服务冲突的解决方法[9,10]。因为程序实施时所处的环境和上下文是高度动态的、多样的、复杂的，针对不同的上下文环境实施某一个服务可能有不同的策略，这样就有可能产生策略上的冲突。例如，两人(Alice 和 Claire)之间使用 PDA 进行对等(peer-to-peer)通信[9]，信息交换的方式有以下几种策略：charMsg，每次传送一个字符；plainMsg，以明文的形式交换信息；compressingMsg，以压缩形式交换信息；encryptedMsg，以密文的形式交换信息。每端(peer)都根据上下文的情况建立自己的信息交换策略集(如图 1 所示)。对于 Alice，电池的能量低于 40%，使用明文(plainMsg)形式交换信息；电池的能量高于 40%时，使用加密的方式交换信息。对于 Claire，如果带宽高于 50%，使用明文(plainMsg)的形式交换信息；带宽低于 50%时，使用压缩(compressingMsg)的格式发送信息。而双方只有在信息交换策略达成一致时才能进行通信。上下文的情况是多样性的，有可能某时段的上下文情况不能满足使双方能达成一致的信息交换策略的条件，这样就产生了冲突。

如图 3 所示，如果两人所处的上下文的环境如下所示：Alice 的 PDA 上的电池的能量低于 40%，Claire 的带宽高于 50%时，可以达成共识，使用明文的形式交换信息；如果 Alice 的电池的能量高于 40%，或者 Claire 的带宽低于 50%，两人应该使用何种信息交换策略？有关策略的冲突及冲突如何解决，尤其是不同设备间的策略冲突如何解决，在这两种编程模型中没有涉及。

Alice	Claire
MessagingService	MessagingService
plainMsg	plainMsg
battery<40%	bandwidth > 50%
encryptedMsg	compressedMsg
battery>40%	bandwidth < 50%

图 3 不同设备间策略冲突例子[9]

除此之外，对于 COP 编程模型还有如下局限性：

(1)没有对匹配程度进行量化。如果有多个候选者，无法确定最优匹配对象。在 Olympus 编程模型中，匹配时如果涉及到多个匹配对象时，采用效用函数(utility function)解决。

(2)没有涉及用户的喜好(preference)。用户的喜好也是影响匹配的关键因素。而在 Olympus 编程模型中，使用元组(〈entity〉〈property〉〈value〉)形式的约束(constraints)描述用户的喜好，从而在匹配时考虑到用户的喜好。

(3)没有考虑到用户对匹配的精确度要求不同。高匹配的精确度和低匹配的精确度对程序实施的要求应该是不同的，从而可以提高程序的运行效率。在 Olympus 编程模型中，关于匹配精确度的要求在用户及空间策略中进行定义，匹配时按照策略的定义执行。

结束语 综上所述，Olympus 和 COP 都采用了基于分离/匹配机制的普适计算编程模型。设计时，程序的主体逻辑与普适计算的环境相分离，使开发人员更好地专注于程序的抽象逻辑的设计；在程序实施时，根据不同的上下文环境使程序的逻辑动态地映射成适用于某一具体计算环境的应用。但两者实施的手段是不相同的。首先，这两种编程模型描述资源的方式是不同的：Olympus 采用本体描述；COP 采用 XML 描述。其次，程序的主体表示的策略不同：Olympus 的程序主体主要由抽象实体及抽象实体可以实施的抽象动作构成，COP 在程序主体中将与上下文相关的信息以 XML 形式表示。第三，在匹配时，Olympus 采用发现机制，基于约束、策略、用户的偏好等将抽象的实体映射为具体的实体，遇到多个候选匹配对象时，采用效用函数找到最优匹配对象；而 COP 采用 tree-to-tree 算法进行 XML 的匹配，不够全面。通过比较，Olympus 编程模型更加成熟，考虑得更加全面，但是，COP 编程模型比较简单，开发者比较容易实施。

从上述分析可知，在基于普适计算的编程模型中，分离/匹配机制可以更好地适应动态的普适计算环境。而如何更好地表示普适环境的资源，对于分离/匹配机制非常重要。在匹配的过程中，冲突总是存在的，冲突应该如何解决，是今后研究的内容之一。除此之外，普适计算的环境是开放的空间，用户的隐私、安全如何保护，信任策略、安全策略如何制定，也是普适计算编程模型考虑的重要方面。

参 考 文 献

[1] Xi Min. Isotope Programming Model: a Kind of Program Model for Context-aware Application // 2007 International Conference on Multimedia and Ubiquitous Engineering. 2007: 597-602

(下转第 266 页)

迁 complete，即整个过程的 BPEL 代码注释为 complete，因此，我们提供的迭代方法可以把工作流网转换为可读的 BPEL 模板代码。

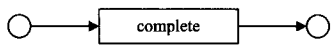

图 6 简化流(flow)结构为一个变迁后的工作流网

结束语 本文给出了有色 Petri 网协作模型的 BPEL 代码实现算法：首先，把有色 Petri 网协作模型转换为结构 WF-net 模型；接着，使用归纳法把 WF-net 模型的子结构替换成标注有 BPEL 代码的变迁。

在案例研究中运用转换算法，首先根据电话机故障修理的需求，创建了该案例的有色 Petri 网协作模型；接着经过分析和简化之后把原始的有色 Petri 网协作模型转换为结构 WF-net 模型；最后，实现了 WF-net 模型的 BPEL 代码实现，显示了转换算法的实用性。

由于主要的目标是产生可读的、易维护的 BPEL 代码，因此没有针对复杂的完全转换(例如，没有使用事件处理结构而是识别典型的 BPEL 结构)。

未来的工作将引入辅助变量或者借鉴程序设计方法学，把无结构的 WF-net 模型转换为 BPEL 代码。

参 考 文 献

[1] Arkin A，Askary S，Bloch B，et al. Web Services Business Process Execution Language Version 2.0. Working Draft. WS-BPEL TC OASIS，May 2005

[2] Jensen K. An Introduction to the Theoretical Aspects of Coloured Petri Nets. http://www.daimi.au.dk/~kjensen/papers_books/rex.pdf

[3] Object Management Group. Unified Modeling Language(UML) 2.1.1. http://www.omg.org/technology/documents/formal/uml.htm

[4] 袁崇义. Petri 网原理与应用[M]. 北京：电子工出版社，2005

[5] http://is.tm.tue.nl/~cgunther/dev/prom/

[6] CPN Group，University of Aarhus，Denmark. CPN Tools Home Page. http://wiki.daimi.au.dk/cpntools/

[7] Ullman J D. Elements of ML Programming. Prentice-Hall，1998

(上接第 235 页)

[2] Bertino E，Ferrari E，Atluri V. The specification and enforcement of authorization constraints in workflow management systems. ACM Transactions on Information and System Security，1999，2(1)：65-104

[3] Wainer J，Barthelmess P，Kumar A. W-RBAC-A workflow security model incorporating controlled overriding of constraints. International Journal of Cooperative Information Systems，2003，12(4)：455-486

[4] Enhydra Shark. http://www.enhydra.org/workflow/shark/index.html

[5] Fitzpatrick G. The Locales Framework：understanding and Designing for Cooperative Work. Australia：Ph.D. Thesis. The Univ. of Queensland，1999

[6] Fitzpatrick G，Mansfield T，Kaplan S. Locales Framework Exploring foundations for collaboration support // IEEE Proceedings of OzCH1. Hamilton，New Zealand，1996：34-41

[7] Crampton J. A reference monitor for workflow systems with constrained task execution // Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies (SACMAT 2005). Stockholm，Sweden，2005：38-47

[8] Crampton J. An algebraic approach to the analysis of constrained workflow systems // Proceedings of the 3rd Workshop on Foundations of Computer Security. Turku，Finland，2004：61-74

[9] Jboss jBMP. http://www.jboss.com/products/jbpm

[10] Yale CAS. http://www.ja-sig.org/products/cas/index.html

(上接第 250 页)

[2] Henricksen K，Indulska J. A Software Engineering Framework for Context-aware Pervasive Computing // Pervasive Computing and Communications，2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference. 2004：77-86

[3] Brown P J，Jones G J F. Context-aware retrieval：exploring a new environment for information retrieval and information filtering. Personal and Ubiquitous Computing，2001，5(4)：253-263

[4] Pascoe J. The stick-e note architecture：extending the interface beyond the user // Proceedings of International Conference on Intelligent User Interfaces. Orlando，USA，1997：261-264

[5] Ranganathan A，Chetan S. Olympus--A High-level Programming Model for Pervasive Computing Environments // Pervasive Computing and Communications，2005. PerCom 2005. Third IEEE International Conference. 2005：7-16

[6] Keays R，Rakotonirainy A. Context-oriented Programming // Proceedings of the 3rd ACM International Workshop. San Diego，CA，USA，2003

[7] Rakotonirainy A. Context-oriented Programming for pervasive space // Proceedings of the ACM Dynamic Languages Symposium. 2005

[8] Román M，Hess C，Cerqueira R，et al. Gaia：A middleware infrastructure to enable active space. IEEE Pervasive Computing Magazine，2002，1：74-83

[9] Capra L，Emmerich W，Mascolo C. CARISMA：Context-aware Reflective Middleware System for Mobile Applications. IEEE transactions on software engineering，2003

[10] Capra L，Emmerich W，Mascolo C. A Micro-economic Approach to Conflict Resolution in Mobile Computing // SIGSOFT. 2002

[11] Arntzen I M. A Programmable Structure for Pervasive Computing // Proceedings of the IEEE/ACS International Conference on Pervasive Services. 2004