一种普适计算环境下自适应中间件

贺建立[1] 陈 榕[1] 康钦马[2]
(同济大学基础软件工程中心 上海 200092)[1] (同济大学电子信息与工程学院 上海 200092)[2]

摘 要 普适计算环境固有的内在复杂性对当前的基础软件提出了新的挑战,迫切需要一种具有感知和自适应能力的中间件。提出了一个由接口、框架和情境元模型组成的自适应中间件,给出了在CAR构件平台上的设计与实现。为获取构件信息和对外提供服务,接口元模型支持同步和异步接口。情境元模型在构件对象内建模情境信息,计算实体间以基于异步事件通知方式交互。框架元模型分类和管理构件,随着运行时计算环境的变化动态改变中间件的结构和行为。软件实体感知环境的变化,实体间以松耦合的方式交互,动态改变自身的结构和行为,满足普适计算环境下的动态自适应需求。

关键词 普适计算,自适应中间件,接口元模型,情境元模型,框架元模型

Self-adaptive Middleware in Ubiquitous Computing Environments

HE Jian-li[1] CHEN Rong[1] KANG Qin-ma[2]
(System Software Engineering Centre of Tongji University, Shanghai 200092, China)[1]
(School of Electronics and Information Engineering, Tongji University, Shanghai 200092, China)[2]

Abstract The inherent complexity of ubiquitous computing poses many new challenges for software infrastructure, which needs urgently the adaptive middleware. This paper proposed a self-adaptive middleware in ubiquitous computing environments, which is composed of the three meta-models, namely the interface, framework and context meta-models. The design and implementation on CAR component platform were presented. The interface meta-model provides access to the services and the internal information and state of a component in terms of asynchronous and synchronous interfaces. The context meta-model represents context information in component object and mask distribution problems in ubiquitous computing environment based asynchronous event notification. The framework meta-model classifies and manages component, makes changes to middleware structures and behaviors according to the changing runtime environments. At last, the software system meets the dynamic self-adaptive requirement in ubiquitous computing environments.

Keywords Ubiquitous computing, Self-adaptive middleware, Interface meta-model, Context meta-model, Framework meta-model

1 介绍

随着像智能手机、个人数字助理、数字照相机这一类手持设备和无线网络技术的发展,各种计算设备逐渐地被利用到人们的整个生活和工作中。普适计算强调把各种计算设备嵌入到环境中,建立一个充满计算和通信能力的环境,让计算设备本身从人们的视线消失,使人们注意的中心回归到要完成的任务本身,最终使人们与环境融为一体[1]。与主流的分布式计算模式相比,普适计算具有一些明显的特征,包括[2]:无所不在的,嵌入式的,游牧的,自适应的和永恒的。普适计算环境固有的内在复杂性对当前的软件技术提出了新的挑战。首先,软件运行的硬件平台资源和处理能力有限,软件应该是轻量级的。由于运行环境的异构、分布和多样性,应用软件在开始运行时,往往装载一些核心功能,随着运行环境的变化,需要动态组合一些其它功能。其次,由于移动设备需要主动适应运行环境(例如网络性能、电量、位置、设备容量等)的改变,应用软件需要感知环境的变化,从而调整自身的结构和行为。最后,因为普适计算环境下硬件设备连接的间断性,互操作应用之间应该是一种松耦合关系。

自适应中间件具有开放性、动态性、可重配置等特征。反射是自适应中间件的支撑技术,使中间件开放了内部部分实现细节。反射性是指系统能够提供自身状态和行为的自我描述,并且系统的实际状态和行为始终保持一致。从结构上考虑,反射中间件分为基层和元层。基层解决特定应用领域的问题,对相关的具体问题进行抽象描述。元层提供一些发现系统内部行为和结构的功能,并使系统在运行时随着外界环境以及自身状态的变化而改变,它的设计与实现是反射中间件研究的中心[3]。本文提出了一种由接口元模型、框架元模型和情境元模型组成的元层多模型结构的反射中间件;介绍了在CAR构件平台上的设计和实现。接口元模型以同步和

到稿日期:2008-08-11 返修日期:2009-05-11 本文受国家"863"计划项目(2001AA113400)资助。
贺建立(1974—),男,博士生,主要研究方向为普适计算和中间件技术,E-mail:hejianli74@gmail.com;陈 榕(1957—),男,教授,博士生导师,主要研究方向为普适计算、操作系统和中间件技术;康钦马(1969—),男,博士生,主要研究方向为嵌入式系统、虚拟仪器。

异步接口的方式获取构件信息并提供内省设施，允许应用开发者与特定环境中动态发现的构件交互。情境元模型在构件对象内建模情境信息，计算实体间以异步事件通知的方式交互。情境元模型屏蔽了应用的分布特征，使得交互对象间以一种松耦合的方式合作，开发者仅仅关注于特定应用领域的语义。针对普适环境的异构、动态、分布、移动等特性，框架元模型定义、分类和管理构件，根据情境的变化动态组装适配其它构件，从而改变中间件的结构和行为。软件实体感知环境的变化，实体间以松耦合的方式交互，动态改变自身的结构和行为，满足普适计算环境下的动态自适应需求。

2 相关工作

当前主流的面向对象中间件技术和标准有 J2EE/EJB，CORBA/CCM，COM/DCOM 和. Net。Enterprise JavaBeans (EJB)[4] 构件是 EJB 容器管理的 Java 类。EJB 容器管理 beans 的执行，处理安全、事务管理、命名和目录接口查询以及远程连接。有 3 种企业 beans：实体 beans 建模商业数据，缓存数据信息；会话 beans 建模商业过程，执行业务逻辑的代理 Java 对象；消息驱动 beans 作为消息监听对象建模消息相关的业务过程。COM 构件是二进制的代码，构件源代码是任何支持函数指针的编程语言，客户通过函数指针调用构件服务。接口是 COM 构件的基础，是构件地址空间的二进制结构，它指向构件函数地址表的指针。一个 COM 构件可以实现很多接口，IUnKnown 是所有构件的根接口，它有 3 个方法。其中两个用于控制构件的引用计数，另一个方法实现构件运行时的接口反思机制，客户通过接口查询获取同一构件上所需的接口。CLR 是. NET 的运行环境，负责装载、执行和管理中间语言(IL)的. NET 类型。. NET 构件是被 CLR 所支持的二进制组装单元[5]，可以被任何所支持语言实现并被编译成 IL 代码。. NET 构件由元数据和 IL 代码组成，元数据包括组装和类型信息以及属性。CORBA 构件元类型扩展和规范了对象元类型，封装了构件内部表示和实现[6]，构件对象的运行环境称为容器(container)。尽管这些标准和技术在传统的分布式应用中获得了成功，但软件技术的目标作用域局限于传统的桌面计算环境。中间件是粗粒度的且缺乏柔性，并不适合开发普适计算环境下的应用软件。因此，迫切需要一种既能感知环境变化且构件间以松耦合的方式交互，又能适应环境的变化，动态改变自身的结构和行为的中间件。

G. Blair 和 G. Coulson[3] 提出了一个通用的中间件反射元层模型 OpenORB，其中的元空间包括接口元对象、截获消息的拦截元对象、管理中间件内部结构的元对象以及资源元对象。OpenCom 是基于这一反射模型实现的一个开放式的构件原型。所有的 OpenCOM 应用构件必须注册到 OpenCOM 的运行环境中，由它负责加载和实例化。运行环境基于相关的元信息造出应用构件的依赖关系，并依此来衡量构件的增加、删除和替换等操作的有效性。OpenORB 强调中间件的自适应，却忽略了软件实体间需要一种松耦合异步的交互模式。

3 中间件模型

反射的本质是开放性，它提供了一种能力，使得中间件对于上层应用适当地暴露其内部实现细节，软件实体了解中间件的运行状态和行为信息，从而修改其结构和行为。通常把反射系统分为结构反射和行为反射两类，它是一个自描述、自观察、自修改的系统。反射技术使得系统在结构上保持关注点分离，一个反射系统分为元层和基层。基层解决特定应用领域的问题，对相关的具体问题进行抽象描述。元层是反射中间件研究的中心，它使系统具有反思和适应能力。一些反射系统的经验已经证明了元层结构带来软件模块化和可伸缩性的好处[7]。

图 1 为反射中间件模型。元层由接口、框架、情境元模型和自描述信息组成。自描述是反射构件技术的核心思想，构件需要把自身的信息提供给构件开发环境和运行环境中的其它构件。自描述信息包括构件提供的服务、构件需要的服务以及构件复用时需要的所有信息。我们认为构件是完整的可独立部署实体，自描述信息不应该以存储的文本文件来描述，而是包含在构件内部，以便于构件部署、升级以及运行时可配置，从而使构件具备自适应能力。

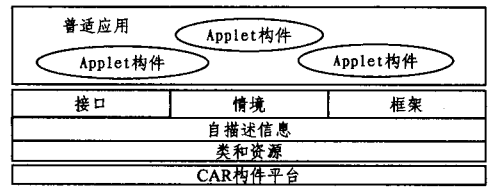

图 1 中间件模型

接口是构件地址空间内的二进制结构，是可以被客户访问的一组操作集合，每个操作有规定的语义，它的布局是由函数指针组成的表，构件通过接口相互连接。接口元模型以接口的方式获取构件信息并提供一些内省设施，允许应用开发者与特定环境中动态发现的构件交互。为获取构件提供的服务，构件实现了一个或多个接口。接口元模型管理构件生命周期，提供内省机制，与外界动态交互。内省设施确保构件客户在运行时获取所需接口。

普适环境下的情境信息域划分为用户域、系统域和环境域 3 个子域。情境感知应用能够感知用户特点、系统行为和物理环境的状态，因此能够根据情境的变化动态调整自身行为。情境建模处理如何收集、组织、表示和存储情境信息，面向对象建模能够很好地处理不完整的、意义不明显、动态的数据。物理环境中的实体建模为面向对象软件中的对象，自描述的数据类型表示实体的位置、状态和时间等信息。“事件驱动体系结构上是对反应式系统的抽象，通过“推”模式的事件通信提供并发的反应式处理，特别适合松耦合通信和支持感知的应用，是解决 SOA 松耦合通信与协调处理的理想解决方案[8]”。情境元模型在构件对象内以面向对象建模情境信息，软件实体间以基于事件通知异步的松耦合的交互方式。情境元模型屏蔽了应用的分布式特征，开发者仅仅关注特定应用领域的语义。

由于普适环境的异构、动态、分布、移动等特性，随着运行条件的变化，普适环境中的动态适应显得异常普遍，框架元模型需要根据情境的变化动态组装适配其它构件，它允许构件在编译和运行时插入其它构件，从而改变功能和结构。主流的构件技术的目标作用域局限于传统的桌面计算环境，并不适用于设备异构的、资源受限的动态普适计算环境。在框架元模型中，我们提出异构构件概念模型，软件系统由 Applet

构件、方面构件和普通构件组装而成。Applet 构件建模领域问题，方面构件表示与服务质量相关的非功能属性，普通构件建模领域边界，静态或动态地绑定方面构件。框架元模型为应用开发者透明地分类、定义、管理、组装和静态或动态地绑定不同类型的构件。方面是分离地表示实体特定属性和功能的代码段，可以被编织到语境的实际执行代码中，从而使语境构件对象共享方面的属性和功能。方面的重用与语境的结合产生了一个构件族。动态方面技术取得关注点分离的同时，方面代码装载和卸载并不需要中断系统的运行。图 2 为基于框架元模型的软件开发过程。

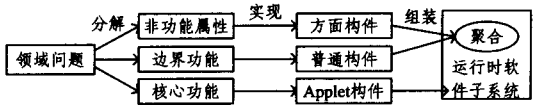

图 2　基于框架元模型的软件开发过程

4　设计和实现

构件中间件技术通过更高层次的抽象定义弥补了面向对象中间件技术的一些缺陷，中间件的构件化可以使中间件能够获得细粒度的灵活控制。CAR（Component Assembly Runtime）是用 C++语言实现的构件平台，构件自描述信息包含在构件内部，以便于构件部署、升级以及运行时自适应。构件自描述信息包括模块信息（ModuleInfo）、接口信息（InterfaceInfo）、类信息（ClassInfo）以及对其它构件依赖关系的导入信息（ImportInfo）。为了获取和操纵这些信息，在 CAR 构件平台上以 C 语言实现了相应的反射设施。

4.1　接口元模型

接口是面向对象构件技术的基础，接口元模型的设计支持 4 个基本原则：接口标识唯一性和接口规范不变性；实现一个或多个接口的构件标识是唯一的；运行时动态接口发现的内省机制；提供构件动态聚合接口。IInterface 是 CAR 构件接口的根接口，它包含 4 个方法：

CARAPI_(PInterface) Probe(REIID riid)，

CARAPI_(UInt32) AddRef()，

CARAPI_(UInt32) Release()，

CARAPI Aggregate (AggregateTypetype, PInterface pObj)。

Probe 方法动态查询构件是否支持某一接口，便于运行时的接口反思。构件被创建后存储于系统的堆空间，AddRef 和 Release 方法维护构件的引用计数，在适当的时候释放系统空间，管理构件生命周期。Aggregate 运行时动态组合其它软件实体。

构件由唯一的标识符标明其身份，新的构件可以由两种方式产生：①运行时主体构件动态绑定方面构件，产生新的构件；②通过以下步骤产生新的构件：(1)编辑 CAR 文件；(2)编译 CAR 文件，产生代码框架；(3)修改源代码，实现构件功能；(4)编译 cpp 文件。CAR 文件定义接口、构件和数据类型。接口元模型支持同步和异步接口，下面的 CAR 文件部分代码中定义了同步接口 IFoo 和异步接口 IBar，接口里的方法分别被客户同步和异步调用。

interface IFoo { Foo();} interface IBar { Bar();}

class CFoo { interface IFoo; asynchronous interface IBar;}

4.2　框架元模型

框架元模型负责分类、定义、管理、静态或动态地合成构件。框架元模型区分 3 种构件：普通构件、方面构件和 Applet 构件。方面构件表征普适环境下多个应用域共性的与服务质量有关的抽象体，在系统设计和部署时独立存储在构件库中，在系统运行时与普通构件组合成新的构件实体。独立的方面构件不提供任何服务，在编译或运行时与普通构件聚合成新的构件，通过接口对外提供服务。普通构件建模领域边界，静态或动态绑定方面构件。Applet 构件是面向应用域的可独立地部署和运行的软件对象。Applet 构件支持更大粒度的构件复用，一个 Applet 构件描述特定的领域问题，一个主 Applet 构件复用多个 Applet 构件、普通构件和方面构件，从而形成一个软件子系统。

构件框架为构件实例创建了环境条件，规定了它们间的交互。形式化的组装推理是语境相关的构件框架的理论基础，可以将一个支持被具有特定属性构件修饰的语境组装机制看作一个构件框架[9]。语境（context）相关的构件框架假定存在一个形式良好的语境以及相关的操作符，组装一些特定的元素，使语境获得某些预期的属性。方面构件是被特定属性修饰的组装元素。语境为普通构件动态聚合或拆卸一个或多个方面构件。此外，语境为构件对象提供运行时的环境，进入语境的对象具有语境的特征，一旦对象离开了语境，环境特征就失去了。图 3 为动态合成过程。

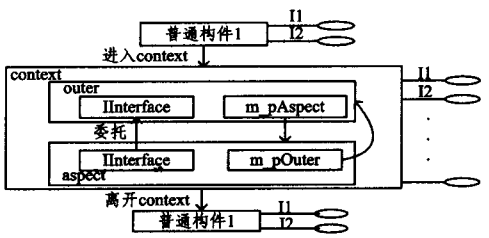

图 3　动态合成过程

在 CAR 文件里，构件属性分类要生成的构件，默认为普通构件，Applet 属性表明要创建 Applet 构件，aspect 定义方面构件，也作为一个属性用来修饰一个语境，说明此语境具有某些 aspect 方面特性。aspect 对象的特征可以被其它构件对象聚合，该构件类实现 IAspect 接口。另一方面，aspect 属性用于修饰 context 构件类，aspect 对象作为语境的属性来表示语境的特征，文法形式为：[aspect(aspect 对象 1，aspect 对象 2，…)]。Aggregate 属性用于修饰类，将方面 aspect1，aspect2 与所要定义的类聚合，文法形式为：[aggregate(aspect1，aspect2…)]。

4.3　情境元模型

情境感知系统基于当前的环境状态为用户提供相应的信息和服务。与之相关的研究包括情境建模、知识共享和用户隐私保护等。在情境建模方面，Ranganathan 以谓词表示情境[10]，谓词的名字描述情境的类型（例如位置、温度以及时间等）。一些谓词有主语和宾语参数（ContextType〈Subject〉，〈Object〉）或主谓宾参数（Context Type(〈Subject〉，〈Verb〉，〈Object〉）。情境元模型在构件对象内以面向对象建模情境信息，软件实体间以基于事件通知异步的方式交互。

定义　事件是描述主体对象的行为、状态或状态的改变过程。形式上描述为：Objclass. Descriptioner. Operator_ Se-

mantic(Data_Type Descriptioner,…)每个事件都属于某个特定的构件对象,称之为主体对象,Objclass 指定事件对象所属的类。Descriptioner 是对象产生的一类事件的抽象描述,为分类对象上操作语义相似的事件。Operator_ Semantic 指定事件的操作语义。Dey 介绍了情境信息的 4 个本质特征[11]:身份、位置、状态和时间。被应用使用的身份标识在命名空间内必须是唯一的,Objclass. Descriptioner. Operator_ Semantic 表明情境信息的身份。自描述的数据类型表达情境的位置、状态和时间等信息。

物理环境中的实体建模为面向对象的软件中的类。当感知到情境信息发生变化时,对象触发相应的事件。事件在服务构件内定义,由客户构件订购。服务构件获取来自传感器或外设的原始数据。以事件建模情境,分类应用客户感兴趣的事件,维护情境数据,当监视到某一事件发生时,立即通知事件注册者。

事件驱动的体系结构在设计时面临高并发性和向软件开发者提供简单透明编程接口的挑战。计算实体或物理设备间交换数据有同步和异步两种模式。就文件系统来说[12],同步保证了提交到磁盘的文件在操作系统崩溃或电源失效时不会丢失数据。但是,在数据提交过程中其它应用不能有效地利用磁盘设备。相反,异步方式能有效地利用磁盘设备,但存在数据丢失隐患。因此,文件系统的设计需要权衡性能和响应时间的问题。构件间交互模式的设计面临更为复杂的系统问题,系统的消息/事件处理机制的线程模型是复杂系统软件系统设计中一个非常核心的部分,对于系统的性能、稳定性和开发成本有决定性的作用。在构件模型方面,引入面向应用领域的 Applet 构件,Applet 构件的回调上下文对象维护事件队列,接收服务构件对象的事件通知,同步调度回调处理函数。在运行的初始阶段,Applet 构件部署满足用户需求的 UI 以及创建服务构件,在服务对象上注册感兴趣的事件,循环处理事件队列中由服务构件投递的事件。图 4 为 Applet 构件和服务构件的交互模式。

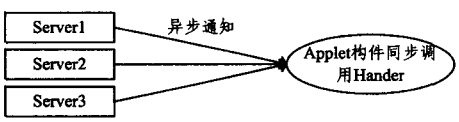

图 4 构件间的交互模式

事件在 CAR 文件里以回调接口定义,由自动代码框架工具产生数据存储、事件合并、事件通知等相应的软件设施,并为客户端产生订阅和取消订阅事件接口。

结束语 由于普适计算环境的复杂性和挑战性,迫切需要一种具有感知和自适应能力的中间件。与其它中间件技术比较,提出的面向普适计算的中间件特点在于以下几个方面:接口元模型以同步和异步接口的方式获取构件信息并提供内省设施和对外的服务;情境信息在构件对象内被建模,构件间以基于事件异步通知的方式交互,获得了时间、空间和同步上的解耦;框架元模型允许应用在开始运行时装载一些核心功能,随着运行环境和需求的变化,动态聚合或拆卸一个或多个方面构件,从而改变自身的结构和行为。异质普适设备的计算能力往往是有限的,在运行阶段,软件启动运行时加载一些核心功能,随着运行环境的变化,系统不间断地添加、删除和替换构件成分。最终,应用能够感知和适应动态变化的环境,满足普适计算环境下的动态自适应需求。

参 考 文 献

[1] Weiser M. The computer for the twenty-first century [J]. Scientific American, 1991, 265(3): 94-104

[2] 徐光祐,史元春,谢伟凯普适计算[J]. 计算机学报,2003,26(9):1042-1052

[3] Blair G, Coulson G, Andersen A, et al. The design and implementation of Open ORB 2[J]. IEEE Distrib. Syst., 2001(6)

[4] Monson-Haefel R. Enterprise JavaBeans 3.0, fifth ed [M]. O'Reilly & Associates, 2006

[5] Barnett M, Schulte W. Runtime verification of .Net contracts [J]. Systems and Software, 2003, 65: 199-208

[6] Object Management Group. CORBA Component Model Specification [EB/OL]. OMG Available Specification Version 4.0, 2006. OMG Document formal/06-04-01

[7] Blair G, Coulson G, Grace P. Research Directions in Reflective Middleware: the Lancaster Experience[C]// Proc. 3rd Workshop on Reflective and Adaptive Middleware (RM2004). Canada, 2004: 262-267

[8] 刘家红,吴泉源. 一个基于事件驱动的面向服务计算平台[J]. 计算机学报,2008,31(4):587-599

[9] Szyperski C. Component Software: Beyond Object-Oriented Programming [M]. Addison-Wesley, 2002

[10] Ranganathan A, Campbell R H. A Middleware for Context-Aware Agents in Ubiquitous Computing Environments [C]// ACM/IFIP/ USENIX Int'l Middleware Conf. LNCS 2672. Springer-Verlag, 2003

[11] Dey A, Abowd G, Salber D. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications[J]. Journal on Human Computer Interaction, Lawrence Erlbaum publ., 2001, 16(2-4): 97-166

[12] Edmund B N, Kaushik V, Peter M C, et al. Rethink the Sync [C]//OSDI'06: 7th USENIX Symposium on Operating Systems Design and Implementation. Seattle, WA, October 2006: 1-14

(上接第 70 页)

[2] Clausen T, Jacquet P. Optimized Link State Routing Protocol [J]. Internet Draft draft-ietf-manet-olsr-11, July 2003

[3] Perkins C E, Bhagwat P. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for mobile computers[C]// SIGCOMM. 1994: 234-244

[4] Chiang C C. Routing in Clustered Multi-hop, Mobile Wireless Networks with Fading Channel[C]// IEEE SICON'97. 1997: 197-211

[5] Perkins C, Royer E, Das S. Ad Hoc On-demand Distance Vector Routings[S]. RFC 3561, 2003-07

[6] Johnson D, Maltz D. Dynamic source routing in Ad hoc wireless networks [C]// Imielinski T, Korth H, eds. Mobile Computing, Kluwer, 1996: 153-181

[7] Park V D, Corson M S. A highly adaptive routing algorithm for mobile wireless networks [A]// Proceedings—IEEE INFOCOM [C]. 1997: 1405-1413

[8] Chai-Keong T. Associativity-based routing for Ad-Hoc mobile networks[J]. Wireless Personal Communications, 1997, 4: 103-139

[9] 冯美玉,程胜,张勖,等. Ad hoc 网络中自愈路由协议研究[J]. 北京邮电大学学报,2005,28(2):46-49