面向普适计算的自适应软件模型及平台实现

史殿习 丁 博 张 威 王怀民

(国防科学技术大学计算机学院 长沙 410073)

摘 要 适应性是普适计算软件的主要特征。现有研究工作在模型和体系结构层面上缺乏对软件自适应的内在抽象,因而很难为自适应软件的建模、开发和运行提供全面支撑。针对现有研究工作存在的不足,首先以适应性为核心,提出了自主单元的概念及其构件化参考实现模型,用统一的、具有适应能力的自主单元来对普适计算实体进行抽象和描述;进而提出基于自主单元的自适应软件体系结构,从概念、开发和运行3个层面对其进行了系统化阐述,并且实现了支撑该体系结构、基于微内核架构的普适计算软件平台UbiStar;最后以智能博物馆应用为典型案例,对自主单元模型和UbiStar平台的有效性进行了验证。

关键词 普适计算,自适应,构件,软件体系结构

中图法分类号 TP311 **文献标识码** A

Self-adaptive Software Model and Platform Implementation for Ubiquitous Computing

SHI Dian-xi DING Bo ZHANG Wei WANG Huai-min

(School of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract Adaptability is the main feature of ubiquitous computing software. Existing research lacks the inherent abstraction of software adaptability in the model and architecture level, which makes it difficult to provide comprehensive support for the modeling, development and running of self-adaptive software. Aiming at this deficiency in existing research, we firstly proposed the concept of Autonomous Units, which is an unified abstraction of pervasive computing entities that centers on adaptability, as well as a reference implementation model of this concept based on the component technology. And then, we proposed an architecture for self-adaptive software based on Autonomous Units, which has been described systematically in its concept, developmented and runtime level. We also implemented a micro-kernel based ubiquitous computing software platform, UbiStar, to reify and support this architecture. Finally, we used the Smart Museum application to verify the effectiveness of our work.

Keywords Ubiquitous computing, Adaptability, Component, Software architecture

1 引言

普适计算是一种新型计算模式,其目标是实现物理空间与信息空间无缝融合,且在融合后的空间中,用户可以随时随地透明地获取所期望的计算服务[1,2]。普适计算软件突破了静态、封闭的桌面系统和网络的限制,运行在复杂多样、动态多变、物理空间和计算空间高度整合的环境中。在这种环境下,要为用户持续地提供其所期望的服务,软件就必须具备自适应能力,也即软件要能够感知环境变化和用户意图,据此主动地调整自身的结构或功能。

自适应是普适计算软件的主要特征[3-6],也是普适计算实现"人本计算"的根本途径。针对这一目标,研究人员已经展开了大量研究工作,具有代表性的研究成果包括Aura,iROS,Gaia,PCOM,MADAM以及欧盟正在进行的PerAda等项目。其中,Aura[7]提供了用户意图捕获、上下文敏感的服务等来支持软件自适应;iRos[8]能够根据运行环境的变化为不同的设备自动生成适合不同设备的人机界面;Gaia[9]通过对传统MVC应用程序框架的扩展来实现软件的自适应;PCOM[10,11]通过监测机制以及内建于系统中的回调机制实现对构件的在线选择和替换;MADAM[12]面向移动应用,通过体系结构动态变化来实现构件实例的版本切换;PerAda[13]项目关注大规模普适计算系统中的个体构件和系统自适应行为。

上述项目在如何开发自适应软件、如何为其提供运行时支撑等方面做出了有益探索。然而,它们在模型和体系结构层面上缺乏对软件自适应的内在抽象,因此相关支撑机制往往集中于某一个维度,无法为自适应软件的建模、开发和运行提供全面支撑。针对这一不足,本文首先以自适应为核心,从

到稿日期:2010-05-26 返修日期:2010-08-23 本文受国家核高基重大专项课题(2009ZX01043-001),国家863课题(2007AA010301,2006AA01Z198)资助。

史殿习(1966—),男,博士,副研究员,主要研究方向为分布计算、中间件、普适计算等,E-mail:dxshi@nudt.edu.cn;丁 博(1978—),男,博士生,主要研究方向为分布计算、普适计算等;张 威(1987—),男,硕士生,主要研究方向为分布计算;王怀民(1964—),男,博士,教授,主要研究方向为分布计算,网络安全。

普适计算空间建模入手，用统一的、具备自适应能力的自主单元概念来对各种设备/资源进行抽象和封装，在屏蔽异构性的同时，使得这些实体具备内在的自适应能力。在此基础上，我们将抽象的自主单元概念映射到成熟的构件技术，提出了构件化的自主单元参考实现模型，进而从概念、开发和运行 3 个维度对基于自主单元的软件体系结构进行了系统性阐述，设计了支撑这一软件体系结构的、基于微内核架构[14]的普适计算平台 UbiStar。上述工作从建模到开发以及运行各个阶段为自适应软件构建提供了支持机制和手段，并已经在基于 UbiStar 平台所开发的智能博物馆应用中得到了验证。

本文第 2 节描述了自主单元概念及其构件化参考实现模型；第 3 节从概念、运行以及开发 3 个维度对基于自主单元的自适应软件体系结构进行系统阐述；第 4 节描述了支撑自主单元运行、面向普适计算的自适应软件平台 UbiStar；第 5 节描述了基于 UbiStar 平台开发的智能博物馆应用；最后，对全文进行了概括和总结，指出了下一步的研究工作。

2 以适应性为核心的自主单元模型

普适计算空间通常由各种各样的实体（如设备、软件、资源以及网络）等构成。我们以适应性为核心，提出了自主单元概念[15]，用自主单元来对普适计算空间中的实体进行抽象。这一抽象为普适计算空间实体添加了内在的自适应能力。自主单元是应用系统中能进行上下文感知的、具备适应能力的基本成分，自主单元之间通过消息交互，构成普适计算应用程序空间，并通过动态协同支持用户完成具体任务。

自主单元适应能力集中体现在按需聚合和动态适应两个方面：按需聚合是指自主单元在运行时可按需加入或退出应用系统，遵守不同的行为准则，与其它自主单元实现动态聚合和协同；动态适应是指自主单元在提供服务过程中能够根据当前上下文进行主动决策，实施适应性动作或修改自身行为方式，从而对物理空间或计算空间变化做出适当响应。

自主单元仅是一个抽象的概念。为了具体化这一概念，我们基于成熟的构件技术[16]，提出了自主单元的构件化参考实现模型，如图 1 所示。构件化自主单元由元层容器和基层构件组成。构件在容器中运行，分为行为构件与感知构件两种类型。前者负责实现业务逻辑；后者是物理传感器或其它上下文提供者的抽象，负责感知物理空间和信息空间的上下文，并将之推送给行为驱动引擎。运行在元层的自适应构件容器是构件的运行环境和驱动部件，它负责监视和管理基层构件实体，从感知构件获取外界上下文信息，根据上下文和所配置的策略通过行为驱动引擎驱动行为构件的执行，调整和管理整构件之间的交互，完成构件的动态配置和部署，改变自主单元对外的功能表现，从而体现出其对环境变化的适应性。

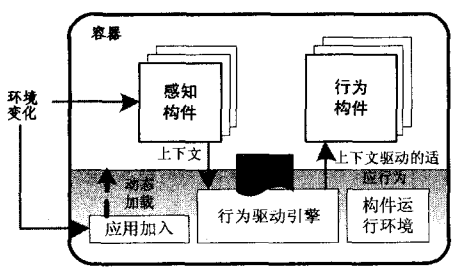


图 1 自主单元的构件化参考实现模型

使用自主单元对普适计算空间中的实体进行抽象，使得无论是用户还是开发者在普适计算空间中看到的是统一的实体，而不再是一个个具有不同特性的设备，从而有效地屏蔽了设备的异性。构件化自主单元参考实现模型将应用逻辑与其运行支撑环境相分离，使得应用逻辑不与具体的运行环境绑定。容器是自主单元的应用逻辑的运行支撑环境，后文将其称为自主单元的“壳”，自主单元的应用逻辑则被称为自主单元的“核”；自主单元的核是可以根据设备当时所处的环境和用户需求动态地从应用发起者或服务器中下载所需的业务逻辑，进而与自主单元壳形成一个完整的自主单元，实现动态绑定；这种特点尤其适合移动和资源受限设备，因为对移动和资源受限设备来说，事先不可能或者也没有必要将所有的可能的业务逻辑都预先安装到设备中，而是根据环境和执行状态的变化动态地下载、更新、替换现有的应用逻辑，从而体现其灵活性和良好的适应性。上述机制使得自主单元可动态的、按需加入到应用中，进而与应用中的其他自主单元协同完成用户任务。

3 基于自主单元的软件体系结构

基于自主单元的普适计算空间的建模，使得普适计算空间中的设备/资源均被抽象为自主单元，因而普适计算空间可被视为若干自主单元所组成的集合，这些自主单元根据用户需求动态聚合成被称为应用系统的群体，并通过动态协同支持用户完成具体任务。基于自主单元的应用系统作为一个整体如何开发、如何启动以及如何运行需要一个完整的体系结构模型来提供支持。为此，本节分别从概念、运行和开发 3 个角度来阐述基于自主单元的件体系结构，目的是从建模到开发以及运行各个方面和阶段为基于自主单元的自适应软件的构建提供一套完整的支持机制和手段。

3.1 体系结构中的主要概念

基于自主单元的软件体系结构在概念上可以分为三层逻辑实体：构件、自主单元和应用程序。

3.1.1 构件

构件是自包含的二进制可重用软件模块。构件分为行为构件与感知构件，前者封装基层业务逻辑，后者封装上下文处理逻辑。所谓上下文，是指相对于某一构件而言的外部环境状态，如网络消息、温湿度等。从功能的角度而言，行为构件是业务功能的实现者，感知构件是上下文的提供者；从反射的角度而言，行为构件是基层计算实体，感知构件是元数据的提供者。

(1)运行阶段的构件概念

运行阶段构件表现为一个个提供接口的“黑盒”。行为构件对外提供的接口被称为业务接口，由用户自行定义；感知构件对外提供的接口被称为上下文接口，所有感知构件的上下文接口都是相同的，包括一套上下文注册/发布方法及获得上下文元信息（如其名称和数据类型）的方法，如图 2 所示。

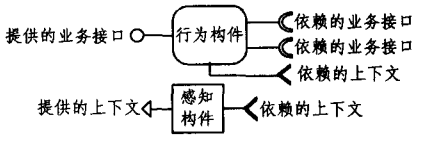


图 2 行为构件与感知构件模型

(2)开发阶段的构件概念

开发阶段的构件由实现体和构件自描述信息组成，即：

构件＝〈实现体，自描述信息〉

构件实现体在模型层面不与具体的开发语言绑定，但不同的语言的具体实现需要不同版本的软件平台的支撑。自描述信息使用 XML 表示，包括构件唯一 ID、构件版本号、构件提供的接口名称、构件依赖的接口名称、构件语言种类及用户指定信息等。自描述信息从 UCDL 构件定义、编译器及用户指定信息中生成：构件定义表述构件的外部特征，使用语言无关的 UCDL（Ubiquitous Component Definition Language）语言（基于 IDL 设计的一种扩展语言）描述。

3.1.2 自主单元

自主单元是采用构件技术实现的、具有自主行为能力的软件模块，也是设备/资源在普适计算空间内的软件抽象。

（1）运行阶段的自主单元概念

自主单元是设备/资源的软件抽象，随设备的启动而启动，自主地加入不同的应用并下载应用相关的部份，从而表现出不同的功能。自主单元有两种状态：休眠和活跃，即通过加入和退出具体应用实现状态之间的切换。自主单元生命周期状态变迁关系及触发条件如图 3 所示。

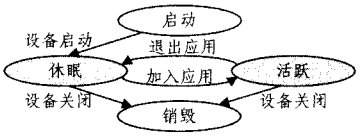


图 3 自主单元的生命周期

活跃状态的自主单元由行为构件、感知构件、策略和行为驱动引擎组成。策略定义了一组“条件-动作”的规则，其条件部份是若干上下文条件的组合，其动作部份则是行为构件的方法（及对行为构件生命周期的管理、构件的动态部署和组装等）。策略是自主能力的表达形式，由应用程序员以 script 或 XML 方式描述。行为驱动引擎在策略的控制下，根据感知构件所提供的当前上下文视图驱动行为构件。休眠状态的自主单元仅包括行为驱动引擎，处于等待下载行为构件、感知构件和策略的状态。行为驱动引擎包括了加入（及下载）和退出应用程序的逻辑。从上下文驱动模型的角度而言，感知构件是上下文提供者，策略是规则制定者，行为驱动引擎是策略的具体执行者，行为构件则是被驱动的功能实现者。从反射的角度而言，感知构件是元数据提供者，策略即为元协议，行为构件是基层计算实体，行为驱动引擎是元层计算实体。每个自主单元有独立于具体应用的描述信息，如该自主单元的唯一 ID、设备标识信息、所能提供的计算资源、可否复用、复用上限、单元目标、发起者地址等。该信息用于运行时自主单元核到自主单元的动态绑定。

（2）开发阶段的自主单元概念

应用程序开发阶段的自主单元表现为自主单元核，它们是自主单元中与具体应用相关的部份。自主单元加入一个应用程序即下载其对应的自主单元核。自主单元核同样是自包含的，其结构如下：

自主单元核＝〈自描述信息，应用相关部件〉

应用相关部件＝〈行为构件＊，感知构件＊，策略＊〉

自主单元核中应用相关部件的概念在前文中均已阐述。自主单元核的自描述信息包括自主单元唯一 ID、包含的构件、关注的上下文以及运行时信息，前三者可以自动生成（从构件自描述及策略中），而运行时信息包括所对应的自主单元版本、可否多次下载、是否只允许启动时下载、可否加入多个应用、启动模型、启动顺序、所需计算资源乃至于只能在哪台指定设备上运行等。

3.1.3 应用程序

应用程序是由一组自主单元耦合而成的完成某一具体任务的执行体。运行阶段的应用程序由一组自主单元及其交互组成。开发阶段应用程序是一组自主单元核的集合，即：

应用程序＝〈自描述信息，自主单元核＋〉

自描述信息主要包括应用程序名称、版本和目标域等。

3.2 体系结构运行机制

运行视图对应用程序启动、运行、终止以及应用程序之间交互机制进行描述。在运行视图中牵涉到实体状态管理与发现服务，它负责管理和查找限定范围的普适计算空间内所有自主单元和应用程序的必需的状态信息，维护一个限定范围的普适计算空间内的所有实体的一个全局视图。该服务可以通过 P2P 形式实现并以容器基础设施的方式提供，也可以通过以集中式的应用程序方式提供。

3.2.1 应用程序的启动

应用程序包括分别运行在多个设备之上的自主单元核，但其启动仅从某个通常具有较少的资源约束的自主单元（如 PC 和 PDA，下称发起者）上启动。应用程序的启动有 3 种模型：主动加入模型（又称“拉”模型）、邀请加入模型（又称“推”模型）和混合模型。

主动加入模型是在每个自主单元上通过静态配置或其它机制获取发起者的地址，应用程序放在发起者上。自主单元启动时加入到发起者所对应的应用，到发起者上下载本单元所对应的核并驱动其运行，而哪个单元下载哪个核在开发阶段即已被指定。

3.2.2 应用程序的运行

同一应用的自主单元在下载完毕自主单元核后，从休眠到活跃状态的变迁可能需要遵循一定的顺序，例如客户端只能在服务端活跃以后才能激活。这种关系是在自主单元核的元信息描述中定义的，并通过各个自主单元监视实体状态管理与发现服务相关自主单元的状态实现。在应用程序启动以后，其运行分为自主单元内部和自主单元之间两部份：一是各个自主单元内部由上下文驱动运行；二是各个自主单元之间的交互从抽象的层次可以视作网络上下文，从实现的层次可以分为两层：基层业务逻辑构件之间的交互，表现为接口调用；元层容器之间的交互，主要是为了维护上下文视图而进行的一些上下文交流、实体发现协议的交互等。从反射的角度而言，所有行为构件的交互构成了完整的应用程序基层业务逻辑，各个自主单元内部则根据感知构件所提供的上下文对这种基层业务逻辑进行自省和调整。

3.3 应用开发过程

开发视图对应用程序完整开发流程进行描述。应用程序的开发视图包括构件开发和应用程序组装两个阶段。构件开发流程如图 4 所示。程序员首先使用 UCDL 语言定义构件的外部特征。UCDL 编译器将 UCDL 描述编译成为构件框架和构件的 XML 描述。程序员基于构件框架类编写实现类，编译后即为构件运行代码；程序员还可能会指定一些构件描述信息（如运行时线程模型等），它们与 UCDL 生成的构件

描述一起合成为最终的构件描述。

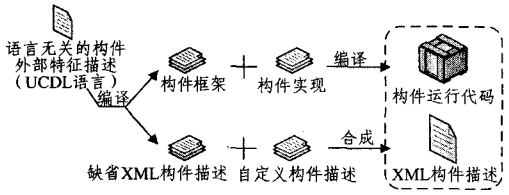


图 4　构件开发流程

构件开发是面向构件类型的开发，应用程序组装是基于构件实例的组装。前一阶段和后一阶段的构件概念是不同的，二者的关系相当于类与对象的关系。如图 5 所示，应用程序组装可以划分为 3 个阶段：构件选择和策略指定阶段、构件组装阶段以及自主单元划分及单元策略指定阶段。

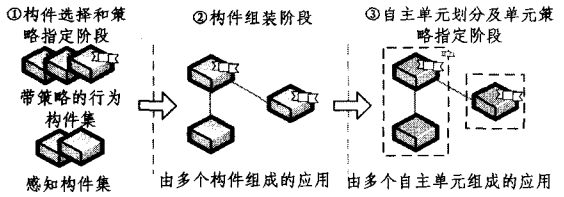


图 5　应用程序组装过程

(1)构件选择和策略指定阶段

在这一阶段，应用组装人员根据具体应用需要选择行为构件和感知构件，并为行为构件根据具体场景指定策略(见图 6)。值得注意的是，策略可能增加构件依赖的上下文，亦即带策略的行为构件可能比原始行为构件多出上下文依赖。在这一阶段可供选择的构件还有一种类型：其它应用程序中的构件。这种构件是以自主单元核为粒度进行选择的，即如果希望与其它应用程序中的构件交互，则将该构件所在的自主单元核的元信息(即自主单元核中除运行代码之外的所有信息)拿过来，本应用程序中的构件就可以访问其它应用程序中的该自主单元核。这些构件在后续阶段必须特殊对待，例如不必理会其依赖的接口，不能划分到其它自主单元中。

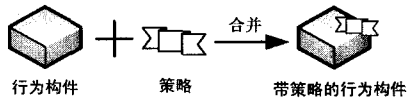


图 6　构件策略指定

(2)构件组装阶段

在这一阶段，应用组装人员将构件根据其依赖和提供的接口类型连接起来。

(3)自主单元划分及单元策略指定阶段

在这一阶段，构件按其未来拟在什么设备上运行划分，同一设备上的构件被划分在一起，自然地形成一个自主单元。如果有必要，则在自主单元上加上严格受限的、跨多个构件的单元级策略(如容错策略)。

4　面向自主单元的平台设计与实现

软件体系结构往往与特定的中间件或应用程序框架绑定，后者可以为前者的开发和运行提供必要的支撑[17]。为有效支持基于自主单元的软件体系结构，我们设计了面向普适计算的构件化自适应软件平台 UbiStar[18](见图 7)。UbiStar 采用微内核架构，由自适应通信层、自主单元框架层、可复用构件层、通用服务自主单元、应用层和开发与管理工具组成。

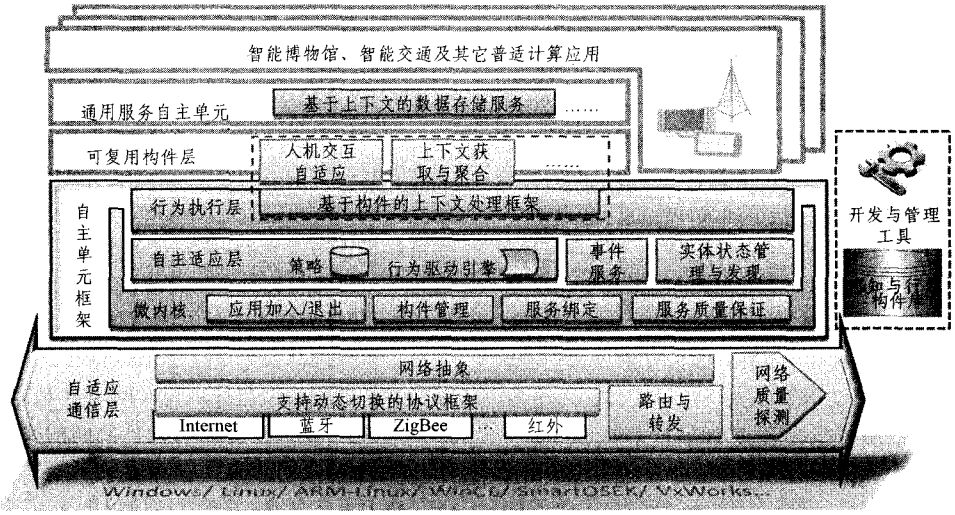
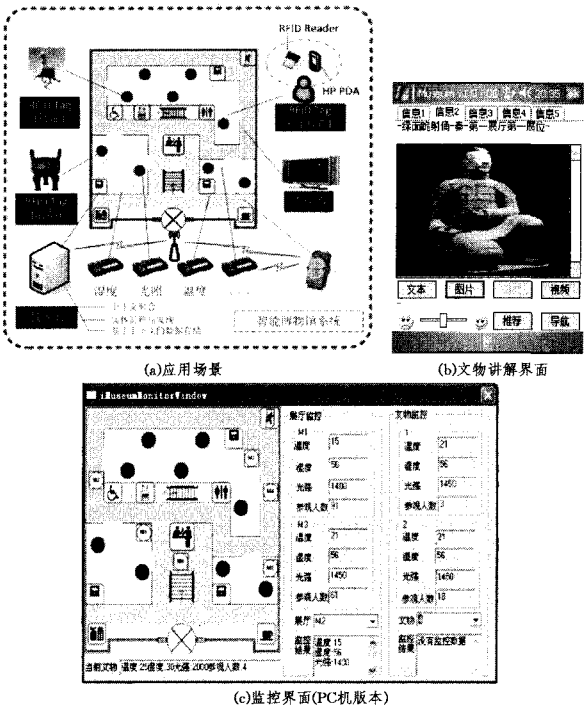


图 7　UbiStar 软件平台架构

UbiStar 对自适应软件开发和运行的支持集中体现在如下两个方面：(1)UbiStar 是基于自主单元的软件体系结构的直接支撑者。UbiStar 通过运行和开发两个层面对自主单元以及自适应语义提供支持：首先，USbitar 是自主单元运行时的基础设施，实现了自主单元构件化实现模型中的应用无关部份，它随设备/资源的启动而启动，可以加入不同应用、下载不同的行为构件、感知构件和策略，是这些业务逻辑相关部件的运行环境。其次，UbiStar 提供一系列面向自主单元的开发工具，包括感知和行为构件接口定义编译器[19]、图形化应用组装配置工具、负责引导自主单元加入过程的应用发起者等。(2)USbitar 内部采用了微内核架构[14]，以适应多维资源受限的环境。微内核由应用加入/退出、构件管理、服务绑定和服务质量保证等支撑自主单元运行的最小功能模块集合构成，其它模块均实现为微内核之上可插拔的系统级行为/感知构件。微内核设计使得 Ubistar 平台可根据运行环境和应用需求静态或动态裁剪，从而适应内存、网络、计算能力等多维资源受限的环境。UbiStar 平台目前可支持 Windows、Linux、WinCE、VXWorks、面向传感器的 ARM-Linux 等多种操作系统，在 WinCE 平台上的最小映象为 97.5kB。

5 应用案例

为有效验证自主单元模型以及 UbiStar 平台的有效性，我们与第三方合作开发了智能博物馆应用。该应用场景描述如图 8(a)所示，所有实体如 PDA、各类传感器等被抽象为具有一定适应能力的自主单元，各个设备(如传感器、PDA、智能手机和 PC 机以及服务器等)都安装了 UbiStar 平台。在参观者进入博物馆展厅之前租用一个可提供多媒体服务的 PDA，进入文物展区后，PDA 上的自主单元壳通过主动加入协议自动加入到博物馆应用中，主动去加载博物馆环境的应用构件，当参观者走到某一个展品旁边，PDA(通过 RFID 读卡器)会主动发现这个展品，然后利用本体上下文查询技术，从远程服务器上下载有关该文物的多媒体信息(如文本、图片、音频、视频等)并在 PDA 进行显示，如图 8(b)所示，PDA 上的文物界面和语音讲解随着参观者的当前文物展柜的位置自动更新，动态变化，由于展品信息保存在远程的服务器上，管理员可以随时更新。



(a)应用场景　(b)文物讲解界面

(c)监控界面(PC机版本)

图 8　智能博物馆演示验证应用场景

在博物馆中，我们还利用 UbiStar 平台建立了一套博物馆监控系统，在博物馆中每个重点的地方放置了 iMote 2[20] 传感器，在 iMote 2 传感器上运行的是 ARM-Linux 操作系统，每个上面都运行了 UbiStar 自主单元，以实时地将传感数据汇总到监控平台当中。管理人员可以通过移动终端 PDA 及 PC 进行展厅温度、湿度、光强及参观人数的监控。当某个 iMote 2 传感器的参数不正常时，监控平台马上进行实时报警(见图 8(c))。监控人员在赶去这个故障地点的过程中，他还可以通过 PDA 实时地监控其它展品的安全情况。

上述场景涉及多种软件自适应行为：设备(如 PDA 上的自主单元)根据环境信息自动加入到应用中，与其他自主单元动态聚合为一个博物馆应用；根据运行环境动态对 UbiStar 平台进行裁剪，如 iMote 2 传感器根据其运行环境的限制加载最小化的自主单元壳；不同设备在不同时刻加载不同的感知和行为构件，例如各个传感器根据承担的不同任务动态下载不同的温度获取构件、光照获取构件等。上述行为均基于 USbitar 平台实现。

结束语　本文提出了以适应性为中心的自主单元模型，使用该模型对普适计算空间实体进行抽象和描述，从而在屏蔽异构性基础上，使得这些实体具备了内在的适应性。在此基础上，提出了构件化自主单元参考实现模型和基于自主单元的软件体系结构，设计了支撑自主单元模型的 UbiStar 软件平台。通过基于 UbiStar 平台所开发的智能博物馆应用，本文工作的有效性得到了充分验证。我们下一步工作是对自主单元模型和 UbiStar 平台进行进一步完善，对 UbiStar 平台的性能进行测试，将其推向实用化。

参 考 文 献

[1] Weiser M. Computers for the 21st Century[J]. Scientific American, 1991, 265(3): 94-105

[2] Saha D, Mukherjee A. Pervasive Computing: A Paradigm for the 21st Century[J]. Computer, 2003, 36(3): 25-31

[3] Niemel E, Latvakoski J. Survey of Requirements and Solutions for Ubiquitous Software[C]// Proc. Mobile Ubiquitous Computing Conf. ACM Press, 2004: 71-78

[4] Grimm R, Davis J, Lemar E, et al. System Support for Pervasive Applications[J]. ACM Transactions on Computer Systems, 2004, 22(4): 421-486

[5] Costa C A, Yamin A C, Geyer C F R. Toward a General Software Infrastructure for Ubiquitous Computing[J]. Pervasive Computing, 2008: 64-73

[6] Garlan D, Schmerl B, Cheng S W. Software Architecture-Based Self-Adaptation[M]. Autonomic Computing and Networking, 2009: 31-55

[7] Garlan D, Siewiorek D P, Smailagic A, et al. Project Aura: Toward Distraction-free Pervasive Computing[J]. Pervasive Computing, 2002, 1(2): 22-31

[8] Ponnekanti S R, Johanson B, Kiciman E, et al. Portability, Extensibility and Robustness in iROS[C]// Proceedings of IEEE International Conference on Pervasive Computing and Communications. 2003: 11-19

[9] Roman M, Hess C, Cerqueira R, et al. Gaia: A Middleware Infrastructure for Active Spaces[J]. Pervasive Computing: 2002, 1(4): 74-83

[10] Becker C, Handte M, Schiele G, et al. PCOM-A Component System for Pervasive Computing[C]// Proceedings of International Conference on Pervasive Computing and Communications. 2004: 67-76

[11] Handte M, Herrmann K, Schiele G, et al. Automatic Reactive Adaptation of Pervasive Applications[C]// Proceedings of IEEE International Conference on Pervasive Services. 2007: 214-222

[12] Floch J, Hallsteinsen S, Stav E, et al. Using Architecture Models for Runtime Adaptability[J]. IEEE software, 2006, 23(2): 62-70

[13] PerAda projec[EB/OL]. http://www.perada.eu/, 2010

[14] Buschmann F, Meunier R, Rohnert H, et al. Pattern-Oriented Software Architecture, A System of Patterns, Volume 1[M]. New York: John Wiley & Sons, 1996

[15] 丁博，王怀民，史殿习. 一种面向普适计算的适应性软件体系结构风格[J]. 软件学报，2009，20(增刊)：113-122

(下转第 169 页)

而有所改变，项目中的风险始终存在。在 PERT/CPM 方法中，项目中的风险事件隐藏在项目活动的工期估计中。受到风险事件的影响，项目活动工期的波动范围很大，使得活动工期的估算高，进而导致项目总工期的估计偏于保守。再加上帕金森法则和学生综合症的影响，屡屡发生项目延期现象。在关键链方法中，Goldratt 看到了活动工期估算的保守以及帕金森法则和学生综合症的后果，提出将单个活动中的安全时间取出来放到整个项目的最后，从而大大缩短了项目工期。关键链方法确实解决了 PERT/CPM 中的一些问题，但是却没有探到问题的源头。在事件链方法中，风险事件从项目活动中分离开来单独建模，使得对项目活动工期的估算更为准确。同时，通过分析风险事件对项目活动的影响，能够更好、更准确地分析风险事件所带来的进度风险。

在进度风险度量方面，三者有相似之处。由于事件链方法中特有的对事件的建模，使得其在风险度量上有着得天独厚的优势，通过基于事件链方法来管理进度风险更为直接和方便。然而作为一种新的方法，事件链方法依然需要更多的研究和实践。

结束语 PERT/CPM 方法最早应用于项目管理，直到现在仍然是主要方法之一；关键链方法也取得了巨大成功，同样应用广泛；而事件链方法方兴未艾，有了初步的实际应用。事件链方法显示出了它相对于前两种方法在风险管理和精确管理等方面的极大优势。然而与前两者相比，事件链方法却比较新，存在较大的探索空间。在 Intaver 公司提出的事件链方法中，一些问题尚待进一步研究，比如事件作用于项目活动的影响模型的建立，比如应用贝叶斯方法来利用历史数据进行模型的学习，比如使用不同于灵敏度分析的其他方法来评估进度风险，比如能否运用模糊数学等数学工具来计算超期风险等等。这些都有待于研究者对事件链方法进行深入的研究和完善。

参 考 文 献

[1] Schwalbe K. IT 项目管理[M]. 邢春晓，张勇，黄梦醒，等译. 北京：机械工业出版社，2008：107-109

[2] Standish Group International. The Chaos Report [EB/OL]. 1994. http://www.standishgroup.com,1994

[3] Standish Group International. The Chaos Report [EB/OL]. 2009. http://www.standishgroup.com,2009

[4] van Genuchten M. Why is Software Late? An Empirical Study of Reasons for Delay in Software Development[J]. IEEE Transactions on Software Engineering,1991,17(6):582-590

[5] 王天青，潘金贵. 基于 CMMI 的软件风险管理[J]. 计算机科学，2005,32(2):140-141

[6] 丁剑洁，郝克刚，侯红，等. 基于粗糙集的软件项目风险管理研究[J]. 计算机科学，2010,37(4):117-119

[7] 项目管理协会. 项目管理知识体系指南[M]. 王勇，张斌，译. 北京：电子工业出版社，2009：228-234

[8] Zuo-Wei Z,Bao X,Qiu-Guo H,et al. Research on Risk Analysis for Construction Project Schedule Based on Flexible Network Simulation[C]// Management Science and Engineering. 2006: 2098-2103

[9] Huang Jian-wen,Wang Xing-xia . Risk Analysis of Construction Schedule Based on PERT and MC Simulation[C]// International Conference on Information Management, Innovation Management and Industrial Engineering. 2009,2:150-153

[10] Fargier H,Galvagnon V,Dubois D. Fuzzy PERT in series-parallel graphs[C]// Fuzzy Systems. 2000

[11] Wang Jin-Hsien,Hao Jongyun. Fuzzy linguistic PERT[J]. IEEE Transactions on Fuzzy Systems,2007,15(2):133-144

[12] Ma Lihua. Research on Risk Degree of Project Duration in U-PERT Network[C]// International Conference on Information Management, Innovation Management and Industrial Engineering. 2009:6-9

[13] 毕鲁雁，焦宗夏，范圣韬. 基于云模型的网络计划建模方法[J]. 计算机工程与设计，2007,28(5):1108-1110

[14] Williams T. Why Monte-Carlo simulations of project networks can mislead[J]. Project Management Journal,2004,35(3):53-61

[15] Goldratt E. 关键链[M]. 罗嘉颖，译. 北京：电子工业出版社，2006

[16] Leach L P. Critical chain project management [M]. Artech House Boston,2005

[17] 雷晓凌，汪小金. 关键链技术中缓冲确定方法的比较研究[J]. 项目管理技术，2007(5):38-40

[18] Herroelen W,Leus R. On the merits and pitfalls of critical chain scheduling[J]. Journal of Operations Management,2001,19(5): 559-577

[19] Virine L,Trumper M. Project decisions:the art and science[M]. Management Concepts Inc,2007

[20] IntaverInstituteInc. ScheduleNetworkAnalysisUsingEventChain Methodology[EB/OL]. http://www.intaver.com

[21] Intaver Institute Inc. Project Management Using Event Chain Methodology[EB/OL]. http://www.intaver.com

[22] Intaver Institute Inc. Software Project Scheduling Under Uncertainties[R]. http://www.riskyproject.com

[23] Intaver Institute Inc. Adaptive Project Management[EB/OL]. http://www.intaver.com

[24] Intaver Institute Inc. MindManager and Risk Analysis [EB/OL]. http://www.intaver.com

[25] Conover W J. 实用非参数统计[M]. 北京：人民邮电出版社，2006：225-239

(上接第 163 页)

[16] Syperski C. Component Software:Beyond Object-Oriented Programming[M]. Boston,MA:Addison-Wesley,2002

[17] Medvidovic N,Mehta N R,Mikic-Rakic M. A Family of Software Architecture Implementation Frameworks[C]// IEEE/IFIP Conference on Software Architecture:System Design,Development and Maintenance. 2002

[18] 史殿习，丁博，李骁，等. 面向普适计算的自适应软件平台：概念与架构[C]//第四届和谐人机环境联合学术会议. 2008

[19] 刘惠，丁博，史殿习，等. 面向普适计算应用的构件定义语言 UCDL[J]. 国防科技大学学报，2009

[20] Nachman L,Huang J,Shahabdeen J,et al. Imote2:Serious Computation at the Edge[C]//Proceedings of International Wireless Communications and Mobile Computing Conference. 2008