ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, Supplement, December 2009, pp.113−122

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种面向普适计算的适应性软件体系结构风格*

丁 博+, 王怀民, 史殿习

(国防科学技术大学 计算机学院,湖南 长沙 410073)

An Adaptive Software Architecture Style for Pervasive Computing

DING Bo+, WANG Huai-Min, SHI Dian-Xi

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

+ Corresponding author: dingbo@nudt.edu.cn

Ding B, Wang HM, Shi DX. An adaptive software architecture style for pervasive computing. *Journal of Software*, 2009,20(Suppl.):113−122. http://www.jos.org.cn/1000-9825/09014.htm

Abstract: Pervasive computing software has to adapt itself to the dynamically changing execution environments and user requirements. This feature complicates software implementation significantly, which makes it necessary to adopt software reuse means on the design level, such as software architecture style, in its development. Based on an adaptive abstract model of pervasive computing space, this paper proposes a software architecture style for pervasive computing, UbiArch, and details it in its concept view, runtime view and development view. UbiArch supports a novel behavior pattern of software entities, i.e., dynamically joining applications according to user requirements and actively adapting itself to the execution environment. As a result, architectural-level can be achieved reuse for software adaptabilities. Besides, this architecture style is based on mature software techniques, such as component technology, which ensure its practicability. A software platform to support this architecture as well as several UbiArch-based applications has been developed to validate the effectiveness and generality of UbiArch.

Key words: pervasive computing; adaptability; software architecture

摘 要: 普适计算软件需要适应用户需求和运行环境的动态变化.这一特点使得软件复杂度空前增加,迫切需要以软件体系结构为代表的架构/设计层面重用手段来支持其高效开发.在以适应性为中心的普适计算空间抽象模型基础上,提出了一种面向普适计算的软件体系结构风格 UbiArch,并从概念视图、运行视图和开发视图这3个维度对该软件体系结构风格进行了阐述.UbiArch 支持软件实体按需加入应用、主动适应环境的行为模式,实现了软件适应能力的高层次重用,同时与构件等成熟软件技术的紧密结合也保证了其可实践性.支撑该体系结构风格的软件平台原型系统及其上的应用验证了 UbiArch 的有效性和通用性.

关键词: 普适计算;适应性;软件体系结构

普适计算具备泛在性、便捷性、适应性的特点[1].泛在性在物理维度上表现为各种各样的计算设备、广泛存在的环境感知和无处不在的网络接入;便捷性在应用维度上表现为用户以最自然甚至不觉察方式享受丰富

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2006AA01Z198 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2005CB321800 (国家重点基础研究发展计划(973)); the National Science Fund for Outstanding Youths of China under Grant No.60625203 (国家杰出青年科学基金)

Received 2008-09-20; Accepted 2009-04-09

的计算服务;适应性在系统维度上表现为软件可感知上下文(情境)并适应其变化.适应性是普适计算软件实现技术的核心,使得软件可以对运行环境及用户需求的变化做出适当响应,从而在泛在运行环境之上表现出便捷应用模式.适应性也同时极大地增加了软件的复杂度,迫切需要有效的软件重用手段来支持其高效开发.然而,传统的软件工程方法大多针对封闭环境下、具备静态结构的软件系统,往往只使用诸如异常、容错协议等语言和算法层面的硬编码机制来实现软件的适应能力[2].软件架构/设计层面适应能力重用机制的缺乏使得普适计算软件难以开发和演化.

软件体系结构是指以构件、构件之间的关系、构件与环境之间的关系为内容的某一系统的基本组织结构,以及指导上述内容设计与演化的原则[3].软件体系结构的研究可以被划分为多个层次,包括软件体系结构风格、软件体系结构模式、平台体系结构、领域体系结构和应用体系结构等[4].其中,由一类应用所共享的软件体系结构风格/惯用模式[5]是软件设计思想的重要重用手段,同时基于体系结构的方法也被认为是实现软件适应的有效途径[6].因此,有必要开展面向普适计算的适应性软件体系结构研究.

普适计算软件体系结构研究已经存在一些初步成果.Garlan等人早期即指出普适计算将在资源占用、灵活性、用户移动等方面为软件体系结构研究带来新的挑战[7];Cheng等人将普适计算系统分为运行层、模型层和任务层,通过基于体系结构的方法实现软件适应[8];ISAM体系结构面向分布式移动应用,通过多层协同来实现适应过程[9];Gaia拓展了传统软件的MVC模型,将应用分成 4 个部分:模型、控制器、表示和元层的协调者[10];PCOM[11],AMUN[12],One.world[13]等普适计算软件平台原型系统也分别提出了所支持的上层应用软件架构,它们均基于对现有的构件技术、Agent技术或者面向服务的体系结构(SOA)的扩展.然而,现有研究大多均处于探索阶段,仍然缺乏统一、完备的软件体系结构[14],尤其缺乏对适应性进行内在抽象和全面支持的工作.

针对普适计算的需求及现有研究不足,本文在对计算空间进行适应性抽象建模的基础上,提出了一种面向普适计算的软件体系结构风格 UbiArch.UbiArch 支持软件实体的按需加入应用、主动适应环境的行为模式,实现了软件适应能力的高层重用.同时,与构件等现有成熟软件技术的紧密结合保证了 UbiArch 的可实践性.本文首先以适应性为中心建立普适计算空间抽象模型,然后将该抽象模型映射到构件等成熟软件技术之上,通过对已有技术的扩展来实现集中体现适应性的 Join/Adapt 操作语义.在从概念视图、运行视图和开发视图 3 个维度对 UbiArch 软件体系结构风格进行详细阐述后,本文构建了支撑该体系结构风格的软件平台原型系统,并通过其上的应用实例验证了该体系结构风格的有效性和通用性.

1 以适应性为中心的普适计算空间抽象建模

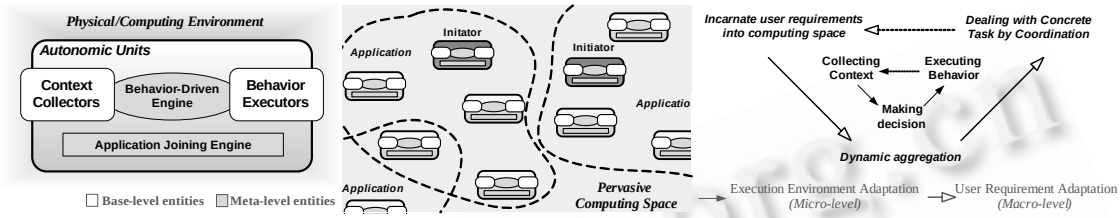
抽象模型可以为普适计算系统的设计和实现提供系统化的方法论指导.已有的一些知名工作包括:美国国家标准与技术研究院参考网络OSI七层模型提出了普适计算概念模型LPC(layered pervasive computing),增加了用户模型、环境和意图层[15];Banavar等人提出了以“设备是门户、软件即任务、物理设施即计算环境”为基本理念的普适计算应用模型[16];Debashis等人给出了由设备、网络、中间件和应用组成的普适计算模型[17];Gaia的活动空间(active space)模型由一个具有良定义边界的物理空间及其上软件基础设施组成[10].区别于上述工作,本节将适应性放在首位,基于所引入的自主单元概念对普适计算空间进行建模,从而为适应性软件体系结构风格UbiArch奠定基础.

1.1 基于自主单元的普适计算空间抽象模型

诸如桌面应用、SOA 服务等传统软件实体均采用手动静态部署、被动提供服务的行为模式,这一模式基于如下假定:软件运行过程中环境和需求不会发生变化,仅仅需要对用户或其他软件实体的调用请求做出响应.然而,普适计算环境下的软件需要能够主动适应用户需求和运行环境的变化,上述行为模式将无法再被沿用.因此,我们在普适计算空间中引入了自主单元的概念:自主单元是普适计算设备/资源的具有适应能力的软件抽象,能够按需聚合为不同应用服务,并在提供服务过程中对环境的变化做出适当响应.

自主单元的适应能力集中体现在其所具备的 Join/Adapt 两类操作语义上:Join 操作语义是指自主单元在运行时可以按需加入或退出应用系统,遵守不同的行为准则,与其他自主单元实现动态聚合和协同;Adapt 操作语

义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的变化做出适当响应.自主单元的适应能力来源于其基于反射的内部结构(如图1(a)所示).它由上下文感知部件、行为执行部件、行为驱动引擎和应用加入部件组成,其中上下文感知部件和行为执行部件是基层计算实体,分别负责收集感兴趣的上下文和实现具体的业务逻辑;应用加入部件和行为驱动引擎是元层计算实体,为 Join 和 Adapt 操作语义提供基础设施:前者负责动态获取应用相关的基层计算实体和元层行为规则,后者依据上下文感知部件所收集的上下文进行决策并进而驱动行为执行部件.



(a) Internal structure of AU (b) Pervasive computing space (c) Adaptation process in abstract model

(a) 自主单元内部结构 (b) 普适计算空间 (c) 抽象模型中的适应过程

Fig.1 Pervasive computing space model based on autonomic units

图 1 基于自主单元的普适计算空间模型

普适计算设备/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(如图1(b)所示),在群体内部通过协同完成指定的任务.聚合过程包括需求捕获、应用映射和动态加入 3 个阶段,由被称为发起者的一类特殊自主单元所引导:发起者有能力捕获用户需求,且可以根据普适计算空间当前状态决定满足用户需求的应用系统需要哪些自主单元参与,然后这些自主单元借助其应用加入部件从发起者处动态获取应用相关的部件,包括基层计算实体和元层行为规则,从而为它们协同完成指定任务奠定基础.

1.2 适应性在普适计算空间抽象模型中的体现

系统科学中的复杂适应系统(CAS)理论[18]指出,自然界中具备适应性的复杂系统由大量具备个体适应能力的节点组成,这些节点进一步通过动态建立关联关系展现出集体层面的适应性,典型的示例包括人类社会中的人与组织、生物界中的蚂蚁与蚁群等.基于自主单元的普适计算模型借鉴了这一理论的思想:微观层面上,每个软件节点(自主单元)通过Adapt语义对其周围的运行环境(如网络带宽、物理位置等)进行适应;宏观层面上,节点间在发起者引导下,通过Join语义实现动态聚合和协同,以适应用户需求的变化.图 1(c)中的两个回路保证了基于自主单元的普适计算空间模型内在的适应性.

2 UbiArch 软件体系结构风格概述

本文后续内容关注如何以现有软件技术为基石,通过完整的软件体系结构风格的设计为前述抽象模型提供参考实现,尤其是如何实现集中体现适应性的 Join/Adapt 操作语义.

构件技术已经被公认为是提高软件开发效率、质量和灵活性的有效途径[19].但在现有构件模型中,容器仅仅被视为构件的运行环境[20,21],无法根据上下文的变化主动驱动和管理构件.从反射技术的角度而言,容器与构件分别运行在元层与基层,这一关系使得通过内部反馈控制回路实现具备主动行为能力的自主单元成为可能.

基于上述思路,UbiArch 对传统构件技术进行了扩展,采用“基层构件+元层容器”架构来实现自主单元(如图 2 所示).我们首先对构件进行细分,提出了行为构件与感知构件模型.其次,我们对传统的构件容器进行扩充,通过增加策略、行为驱动引擎、应用加入等部件来增强其元层功能.其中,策略是一组行为规则,描述了“何时干什么”,是开发者和管理人员指定和约束自主单元适应能力的方式.行为驱动引擎解释和执行策略,依据从感知构件处获取的上下文信息驱动行为构件运转、调整和管理构件间交互、完成构件生命周期管理等,实现 Adapt 语

义;应用加入部件动态获取当前所加入应用的元层策略和基层构件,为 Join 语义的实现奠定基础.

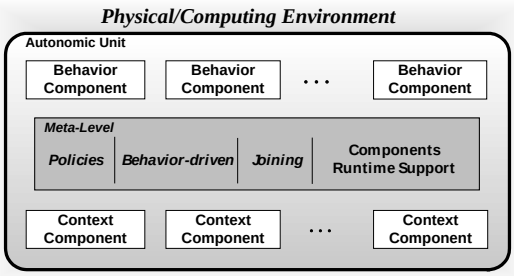


Fig.2 Component-Based autonomic unit

图 2 自主单元的构件化实现

对于第 1 节中所提及的聚合过程的需求捕获、应用映射和动态加入 3 个阶段,UbiArch均提供了相应的实现.对于后两个阶段,UbiArch设计了基于资源描述的应用映射机制和 3 种不同的自主单元加入模型:前者通过资源发现和匹配将应用动态映射到普适计算空间中,从而决定需要哪些自主单元来完成指定的任务;后者包括主动加入模型、邀请加入模型和混合模型,它们定义了自主单元在加入过程中与发起者之间的交互协议.需求捕获阶段目前则仅通过用户选择所要启动的应用这一原始方式实现,但UbiArch的设计并未排除使用诸如意图推导[22]等其他方式的可能性.

图 3 以 UML 标记法给出了 UbiArch 软件体系结构风格中重要的概念及它们之间的关系,它们将会在第 3 节中予以详细阐述.

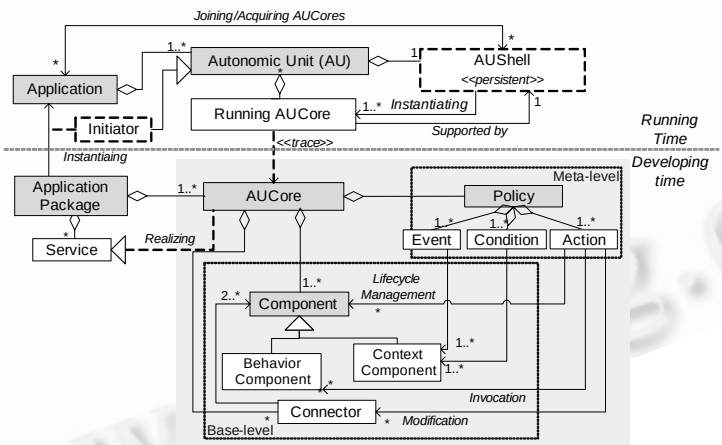


Fig.3 Meta-Model of UbiArch software architecture style

图 3 UbiArch 软件体系结构风格元模型

3 UbiArch 软件体系结构多视图模型

多视图模型被广泛用来描述软件体系结构[23],例如Kruchten等人所提出的“4+1”视图模型[24]、RM-ODP的 5 类视图框架[25]等.但现有的多视图模型描述对象往往是具体应用体系结构,而非一类应用所共有的体系结构风格.因此在本节中我们并未沿用已有工作,而是从概念、运行和开发 3 个视图对UbiArch软件体系结构风格进行阐述:概念视图定义UbiArch体系结构风格下自主单元和应用系统的静态结构,运行视图说明自主单元是如何动态聚合的,开发视图则给出UbiArch体系结构风格下的软件开发方法.

3.1 概念视图:自主单元的构件化实现

在 UbiArch 中,自主单元通过扩展传统“构件+容器”架构实现,包括对基层构件模型的扩展和对元层容器功能的扩展两个方面.

(1) 对基层构件模型的扩展

在介绍 UbiArch 对构件模型所做扩展之前,我们需要对上下文的相关概念进行定义:

定义 1(上下文和上下文事件). 上下文 c 可定义为二元组 $\langle name,type\rangle$,其中 $name$ 是所提供的上下文名称,$type$ 是该上下文的类型;上下文 c 的值被记为 v;上下文事件 ec 被定义为三元组 $\langle c,v_1,v_2\rangle$,代表上下文 c 值从 v_1 变化到 v_2,其中 $type(v_1)=type(v_2)=c.type$.

在UbiArch构件模型中,构件被细分为感知构件和行为构件.区分感知构件与行为构件的原因在于二者在构件语义[26]上的差异:前者封装上下文获取、聚合等的处理过程,并以上下文事件方式将之提供给元层容器;后者则沿续一般意义上封装业务逻辑的构件的语义.

定义 2(感知构件和行为构件). 感知构件 $CComp$ 是上下文的提供者,可定义为三元组 $\langle S_{CPp},S_{CPr},F\rangle$.其中 S_{CPp} 是所提供的上下文接口集合,它以上下文事件的方式输出上下文,S_{CPr} 则是所依赖的上下文接口集合;$F:S_{CPr}\cup E(CComp)\rightarrow S_{CPp}$ 定义了感知构件的功能,其中 $E(CComp)$ 是 $CComp$ 的运行环境;行为构件 $BComp$ 是业务逻辑的实现者,可定义为四元组 $\langle S_{Ip},S_{Ir},S_{CPr},f\rangle$.其中 S_{Ip} 是所提供的业务接口集合,S_{Ir} 是所依赖业务接口集合,S_{CPr} 是直接依赖的上下文的集合;$f:S_{Ir}\cup S_{CPr}\rightarrow S_{Ip}\cup E(BComp)$ 定义了行为构件功能.

上述定义中的业务接口是指传统意义上提供方法和属性的普通接口,区别于以上下文事件方式输出的上下文接口.连接子在 UbiArch 体系结构风格中是一阶实体,其定义如下:

定义 3(连接子). 设 $Comps$ 为感知与行为构件集合,连接子 $Con_{Comps}\in m_{CP}(Comps)\cup m_I(Comps)$,其中 $m_{CP}:\{CP_r\mid\exists C\in Comps \text{ and } CP_r\in C.S_{CPr}\}\rightarrow\{CP_p\mid\exists C\in Comps \text{ and } CP_p\in C.S_{CPp}\}$ 定义了上下文接口依赖关系,$m_I:\{I_r\mid\exists C\in Comps \text{ and } I_r\in C.S_{Ir}\}\rightarrow\{I_p\mid\exists C\in Comps \text{ and } I_p\in C.S_{Ip}\}$ 定义了业务接口依赖关系.

(2) 对元层容器功能的扩展

UbiArch 将扩展了元层功能的构件容器称为自主单元壳(shell).自主单元壳与传统构件容器的区别在两个方面:可通过感知构件获取上下文,动态解释和执行“何时做什么”的策略;能随设备/资源启动而启动,动态加入应用系统,获取应用系统相关的自主单元核(core).

定义 4(策略). 策略 P_{Comps} 是在构件集合 $Comps$ 上使用ECA模式[11]定义的行为规则,表现形式是三元组 $\langle events,conditions,actions\rangle$,其中 $events=\{ec\}$ 是所要匹配的上下文事件集合;$conditions$ 是上下文值所要符合的条件集合;$actions$ 是在事件发生且条件满足前提下要实施的动作集合,其元素可以是 $Comps$ 中构件业务接口的调用、构件生命周期管理或某一个 Con_{Comps} 中的连接子的修改动作.

定义 5(自主单元核和自主单元壳). 自主单元核 $AUCore$ 是自主单元中应用相关的部分,可被定义为五元组 $\langle R,BComps,CComps,Con_{BComps\cup CComps}{}^{*},P_{BComps\cup CComps}{}^{*}\rangle$,其中 R 是 $AUCore$ 运行所需的资源描述,$BComps$ 是行为构件集合,$CComps$ 是感知构件集合;自主单元壳 $AUShell$ 是自主单元中应用无关的部分,可被定义为一个四元组 $\langle R,D,J,E\rangle$,R 是 $AUShell$ 所封装的资源,D 是负责解释和执行策略的行为驱动引擎,J 是动态获取自主单元核的应用加入部件,E 是传统意义上的构件运行环境.

定义 6(自主单元). 自主单元 AU 是一个二元组 $\langle AUCore^{*},AUShell\rangle$.自主单元所提供的服务 $Service_{AU}=\{I_p\mid\exists Comp\in AU.AUCore \text{ and } I_p\in Comp.S_{Ip}\}$.

在定义 6 中,自主单元由一个壳和零到多个核组成,也即自主单元可以处于无核的等待加入应用的状态,此时对外不提供任何服务;自主单元壳也可以被多个核所复用;自主单元对外所能提供的服务也随其当前核不同而不同.

定义 7(应用系统). 应用系统 App 是一组为完成指定任务而在运行时动态聚合而成的自主单元集合,该集合中所有依赖的上下文接口和业务接口均已找到相应的提供者.

3.2 运行视图:从自主单元到普适计算空间

尽管我们已给出了应用系统的定义,但未涉及自主单元是如何动态聚合的.运行视图将详细阐述这一过程.如前所述,聚合过程的需求捕获阶段在 UbiArch 中是通过直接让用户选择所需启动的应用方式实现的,而其他阶段的实现则涉及如下两个方面:

(1) 基于资源描述的应用映射机制

如前所述,自主单元随设备/资源的启动而启动,在加入应用前处于无核的等待状态,因此我们可以将普适计算空间定义如下:

定义 8(普适计算空间). 具有确定边界的普适计算空间 *PS*={*AUShell*|*AUShell.R* 位于该空间内}.

尽管边界是确定的,定义 8 所给集合的中元素仍随时间变化而变化,可能不断有设备因为能源、物理移动、故障等因素加入或退出这个集合.

定义 9(应用系统包). 开发完毕的基于 UbiArch 体系结构风格的普适计算应用被称为应用系统包.应用系统包 $AppPackage=\{AUCores, ExtraRequiredServices\}$,其中 *AUCores* 是一组自主单元核的集合,*ExtraRequiredService* 则是所依赖但未被本应用系统包中自主单元所实现的服务集合.

定义 10(映射). 应用系统包的资源需求 $R_{AppPackage}=\{AUCore.R \mid AUCore \in AppPackage\}$,普适计算空间的资源状态 $R_{PS}=\{AUShell.R \mid AUShell \in PS\}$.应用系统包 *AppPackage* 到普适计算空间的映射是一个函数 $mapping: R_{AppPackage} \to R_{PS}$,对于每一个 $mapping(R_1)=R_2$,皆有 R_2 可满足 R_1 所描述的需求.

映射过程将应用系统包动态映射到普适计算空间,一方面使得应用系统包可以与具体普适计算空间解耦,另一方面使得自主单元动态聚合成为可能.映射过程由发起者完成,是Join语义的重要实现环节,其基本算法如下所示,其中第 2 行和第 9 行可基于现有的资源发现技术实现[27].

算法 1. 应用系统包到普适计算空间的映射.

```
1.  //自主单元核到自主单元壳的映射
2.  find a resource matching function mapping for application package APP in current pervasive
    computing space;
3.  for each AUCore in APP.AUCores
4.      notify AUShell that encapsulates mapping(AUCore.R) to join P;
5.      wait for AUShell to download AUCore;
6.  end for
7.  //额外需要的服务到已运行的自主单元的映射
8.  for each ExtraRequiredServcie in P.ExtraRequiredServices
9.      find an existing Adaptive Unit which implement ExtraRequiredService;
10. end for
```

(2) 自主单元加入模型

自主单元加入模型定义了自主单元与发起者之间的交互协议.针对不同的应用场景,UbiArch 支持 3 种不同的自主单元加入模型:1) 主动加入模型,即自主单元拥有发起者的地址,主动向发起者发出加入请求,下载元层策略和基层构件,从而加入到指定应用之中.它不涉及到映射本身,优点是简单、可预期性好,适用于专用设备且发起者地址固定的场合;2) 邀请加入模型,即发起者根据应用系统包的资源需求,查找符合需求的自主单元,并发出加入邀请.自主单元在收到邀请加入的请求后,到发起者处下载对应的元层策略和基层构件,加入到指定应用之中.该模型具有较佳的灵活性,但可能会出现无法成功映射等问题;3) 混合模型,即在同一个应用系统映射过程中部分自主单元核采用主动加入模型,部分采用邀请加入模型.

在上述机制和模型的基础上,UbiArch 体系结构风格下自主单元动态聚合过程如图 4 所示.图中同时给出了这一过程与抽象模型中聚合过程 3 个阶段的对应关系,以及发起者与一般自主单元在这一过程中的分工.

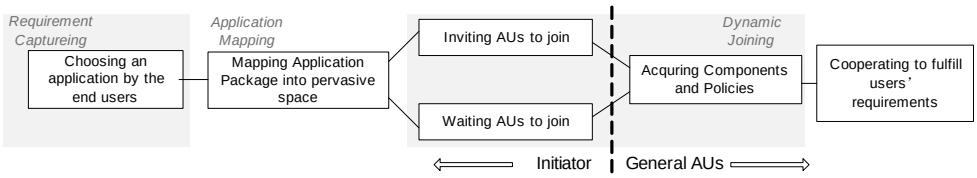


Fig.4 Dynamic aggregation process of autonomic units

图 4 自主单元动态聚合过程

3.3 开发视图:继承成熟的构件开发方法

UbiArch 最大程度继承了成熟的构件化软件开发方法,以保证其可实践性和应用开发者较低的学习成本.UbiArch 开发视图可以分为构件开发和应用系统开发两个部份:前者的目的是获得用于搭建自主单元的行为构件和感知构件,其具体流程与传统构件开发方法无异;后者目的是获得应用系统包,包括组装、可选的策略指定及打包 3 个阶段.

4 UbiArch 支撑环境与应用验证

软件体系结构往往需要运行时框架的支撑[28].为支持基于UbiArch软件体系结构风格的应用程序,我们开发了自适应软件平台UbiStar,进而在其上构建了一系列应用实例来验证UbiArch的有效性和通用性.

4.1 支持UbiArch的软件平台原型

软件平台是普适计算环境下的基础软件之一,知名项目包括以展露上下文变化、鼓励自主组合、缺省数据共享为设计理念的One.world[13];基于活动空间模型,由核心和应用程序框架组成的Gaia[10];在设计理念上强调软硬件各层的主动性与自调整的Aura[29];面向普适计算的构件中间件PCOM[11]等.这些软件平台往往支持某种特定的上层应用体系结构,为该体系结构提供可重用的软件基础设施,从而使具备普适计算特点的应用得以快捷高效的开发和运行[1].

UbiStar 以支持适应性软件体系结构风格 UbiArch 为目标,其设计如图 5 所示.

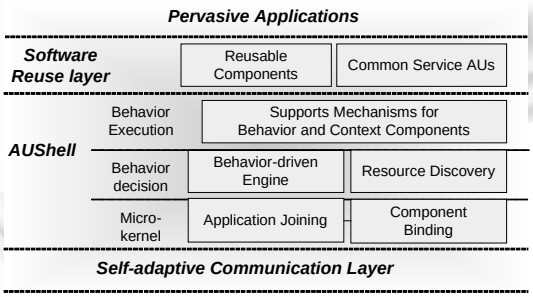


Fig.5 Architecture of UbiStar Platform

图 5 UbiStar 平台基本架构

各层功能依次如下:

(1) 自适应通信层:提供网络抽象,实现在不同网络间的互操作及网络间自适应漫游和切换;

(2) 自主单元壳:直接为UbiArch软件体系结构风格提供运行时支撑.该层可进一步细分为微内核层、行为决策层和行为执行层,分别负责核心功能、自主决策和支撑动作实施.微内核设计模式[30]的采用使得UbiStar软件平台自身能够适应资源受限的计算设备;

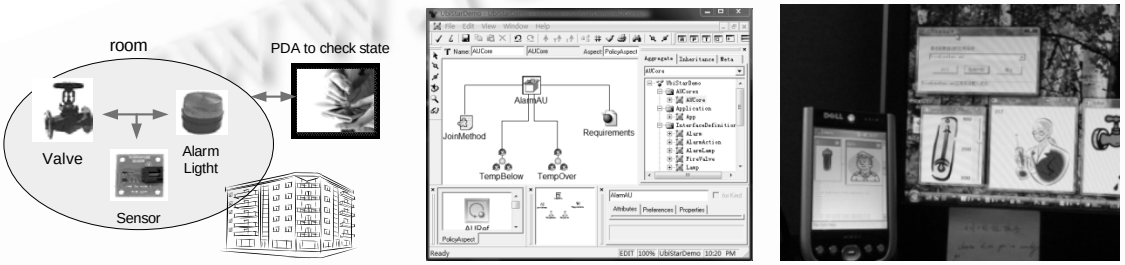
(3) 软件复用层:以可复用构件和通用服务自主单元两种形式为基于 UbiArch 体系结构风格的应用提供各种公共服务,如上下文聚合、上下文敏感的数据管理、资源发现等;

(4) 应用层:由自主单元构成的、基于 UbiArch 软件体系结构的各种普适计算应用系统.

此外,为支持 UbiArch 软件体系结构风格下应用的开发及运行时管理,UbiStar 还提供了一组开发与管理工具,包括图形化建模工具、构件规约到具体编程语言映射的编译器、策略动态更新工具等.

4.2 基于UbiArch的应用实例

我们在 UbiStar 基础上构建了火警响应、远程实验监控等一系列基于 UbiArch 软件体系结构风格的应用实例(场景).其中,火警响应应用由 imote2 传感器、PDA 和 PC 组成,实现模拟的火警响应过程和远程状态查询功能(如图 6 所示).该应用在开发时并未绑定到具体的设备,而仅仅描述了各个自主单元核的资源需求.UbiStar 软件平台运行于参与该应用的所有设备上,将它们抽象为自主单元.当应用系统包被用户选择后,发起者可以将之实时映射到由这些自主单元所组成的普适计算空间中.各个自主单元通过邀请加入模型加入应用系统,在策略驱动下对传感器收集到的上下文进行响应.策略可以通过 UbiStar 所提供策略动态更新工具进行在线升级,从而动态改变各个自主单元的适应能力,如火警响应的阈值和火警发生时所要执行的动作等.远程实验监控应用则着重验证 Join 语义.它面向通过摄像头监视实验室中长时间化学反应的场景:当监控者离开下在显示实验影像的 PC 机时,其所携带的智能手机可以动态加入到该应用中,下载与智能手机对应的自主单元核,从而实现从 PC 机到智能手机的透明切换,保证监控者可以随时随地查看正在进行的实验影像.



(a) Simulated scenario (b) Modeling in Ubistar's development tool (c) Running snapshot

(a) 所模拟的场景 (b) 在 UbiStar 图形化开发工具中建模 (c) 运行时截图

Fig.6 Fire alarm application

图 6 火警响应应用

5 结束语

UbiArch 软件体系结构风格在构件等成熟软件技术基础上,通过引入感知和行为构件模型、增加行为驱动引擎和应用加入部件扩展元层容器功能,使得软件实体可以主动适应环境;通过建立应用系统到普适计算空间的映射机制、定义适用于不同场景的自主单元加入模型,使得软件实体可以按需加入应用.通过上述设计,UbiArch 支持软件对运行环境和用户需求的适应,实现了适应能力在架构/设计层面的重用,同时对构件技术的向前兼容保证了其可实践性.UbiStar 软件平台及其上应用验证了我们已有工作的有效性和通用性.我们未来的工作包括基于体系结构的软件动态适应方法、UbiArch 体系结构风格下的软件演化模型以及自主单元间交互协议的进一步研究等.

References:

[1] Ding B, Shi D, Wang H. Pervasive middleware technology. Journal of Frontiers of Computer Science & Technology, 2007,1(3): 241−254 (in Chinese with English abstract).

[2] Cheng SW, Huang AC, Garlan D, *et al*. Rainbow: Architecture-Based self adaptation with reusable infrastructure. IEEE Computer, 2004,37(10):46−54.

[3] Hillard R. IEEE-std-1471-2000: Recommended practice for architectural description of software-intensive systems. IEEE, 2000.

[4] Frakes WB, Kang K. Software reuse research: Status and future. IEEE Trans. on Software Engineering, 2005,529−536.

[5] Le Metayer D, Irisa R. Describing software architecture styles using graph grammars. IEEE Trans. on Software Engineering, 1998, 24(7):521−533.

[6] Oreizy P, Gorlick MM, Taylor RN, *et al*. An architecture-based approach to self-adaptive software. IEEE Intelligent Systems, 1999, 54−62.

[7] Garlan D. Software architecture: A roadmap. In: Proc. of the Conf. on The Future of Software Engineering. 2000. 91−101.

[8] Cheng SW, Garlan D, Schmerl BR, *et al*. Software architecture-based adaptation for pervasive systems. In: Schmeck H, Ungerer T, Wolf L, eds. Proc. of the Int'l Conf. on Architecture of Computing Systems: Trends in Network and Pervasive Computing. 2002. 67−82.

[9] Augustin I, Yamin AC, Barbosa J, *et al*. ISAM, a software architecture for adaptive and distributed mobile applications. In: Proc. of the 7th Int'l Symp. on Computers and Communications (ISCC 2002). Washington: IEEE Computer Society, 2002.

[10] Roman M, Hess CK, Cerqueira R, *et al*. Gaia: A middleware infrastructure to enable active spaces. IEEE Pervasive Computing, 2002,1(4):74−83.

[11] Becker C, Handte M, Schiele G, *et al*. PCOM-A component system for pervasive computing. In: Proc. of the IEEE Conf. on Pervasive Computing and Communications. 2004. 67−76.

[12] Trumler W, Bagci F, Petzold J, *et al*. AMUN—Autonomic middleware for ubiquitous environments applied to the smart doorplate project. Advanced Engineering Informatics, 2005,19(3):243−252.

[13] Grimm R, Davis J, Lemar E, *et al*. System support for pervasive applications. ACM Trans. on Computer Systems (TOCS), 2004, 22(4):421−486.

[14] Jiang LF, Lu GZ, Xin YW. Research on software architecture under pervasive computing environment. Computer Science, 2007,34(9):249−252 (in Chinese with English abstract).

[15] Ciarletta L, Dima A. A conceptual model for pervasive computing. In: Proc. of the Int'l Workshops on Parallel Processing. 2000. 9−15.

[16] Banavar G, Beck J, Gluzberg E, *et al*. Challenges: An application model for pervasive computing. In: Proc. of the 6th Annual Int'l Conf. on Mobile Computing and Networking. 2000. 266−274.

[17] Saha D, Mukherjee A. Pervasive computing: A paradigm for the 21st century. Computer, 2003,25−31.

[18] Holland JH. Hidden Order: How Adaptation Builds Complexity. Boston: Addison Wesley Publishing Company, 1996.

[19] Heineman GT, Councill WT. Component-Based Software Engineering: Putting The Pieces Together. Boston: Addison-Wesley Longman Publishing, 2001.

[20] Gobel S, Aigner R, Pohl C, *et al*. The COMQUAD component container architecture. In: Proc. of the 4th Working IEEE/IFIP Conf. on Software Architecture. 2004. 315−318.

[21] Kobryn C. Modeling components and frameworks with UML. Communications of the ACM, 2000,43(10):31−38.

[22] Satyanarayanan M. Pervasive computing: Vision and challenges. IEEE Personal Communications, 2001,8(4):10−17.

[23] Clements P. Documenting Software Architectures: Views and Beyond. Boston: Addison-Wesley Professional, 2003.

[24] Kruchten PB. The 4+1 view model of architecture. IEEE Software, 1995,12(6):42−50.

[25] Steen M, Derrick J. ODP enterprise viewpoint specification. Computer Standards & Interfaces, 2000,22(3):165−189.

[26] Lau KK, Wang Z. Software component models. IEEE Trans. on Software Engineering, 2007,33(10):709.

[27] Liu X. Research and implementation of entity discovery system in pervasive computing environment [MS. Thesis]. Changsha: National University of Defense Technology, 2007 (in Chinese with English abstract).

[28] Medvidovic N, Mehta NR, Mikic-Rakic M. A family of software architecture implementation frameworks. In: Proc. of the 3rd IEEE/IFIP Conf. on Software Architecture: System Design, Development and Maintenance. 2002. 221−235.

[29] Garlan D, Siewiorek D, Smailagic A, *et al*. Project aura: Toward distraction-free pervasive computing. IEEE Pervasive Computing, 2002,1(2):22−31.

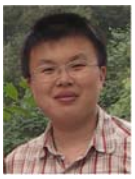
[30] Buschmann F, Meunier R, Rohnert H, *et al*. Pattern-Oriented Software Architecture. Vol.1. A System of Patterns. New York: John Wiley & Sons, 1996.

附中文参考文献:

[1] 丁博,王怀民,史殿习.普适计算中间件技术.计算机科学与探索,2007,1(3):241−254.

[14] 姜丽芬,卢桂章,辛运帏.普适计算软件体系结构研究.计算机科学,2007,34(9):249−252.

[27] 刘璇.普适计算环境下实体发现机制的研究与实现[硕士学位论文].长沙:国防科学技术大学,2007.



丁博(1978−),男,湖南攸县人,博士生,主要研究领域为分布计算,普适计算.

史殿习(1966−),男,博士,副教授,主要研究领域为分布计算,普适计算.

王怀民(1962−),男,博士,教授,博士生导师,主要研究领域为分布计算,网络安全.