基于信息物理系统的面向方面中间件特征模型

吴 帅,张立臣
(广东工业大学 计算机学院,广东 广州 510006)

摘 要:信息物理系统(Cyber-physical Systems,CPS)是集计算、通信和控制能力于一体的多维复杂的分布式系统,这就要求系统必须通过专门的定制。实时中间件可以有效地解决分布式系统的异构性和互操作性等问题。信息物理系统中间件存在着分布性、容错性、性能分析、实时性、日志记录等非功能的横切关注点,基于面向方面的中间件构建技术,能够把这些横切关注从中间件中分离出来,形成独立于中间件的核心功能的方面。文中提出一个面向方面的中间件,能够处理分布式信息物理系统的动态问题,在文中,将重点放在特征模型定义,并且概述了其映射到一个动态的面向方面的中间件平台。

关键词:面向方面;信息物理系统;中间件;特征模型;软件产品线
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2012)02-0076-04

A Feature Model of Aspect-Oriented Middleware for Cyber-Physical Systems

WU Shuai,ZHANG Li-chen
(College of Computer, Guangdong University of Technology, Guangzhou 510006,China)

Abstract:CPS(Cyber-physical Systems) is one multi-dimensional complex distributed system setting of computing,communications and control capabilities,which requires the system must pass a special customization. Real-time middleware can effectively solve the heterogeneity and interoperability issues of distributed systems. Aspect oriented middleware can separate non-functional concerns,such as distribution,fault-tolerance,performance analysis,real-time,logging,etc,from the core functions of middleware,implement them as independent of core functions. It will propose a family of aspect-oriented middleware platforms,is able to deal with the high dynamic issue of CPS. It will focus on the feature model definition and also outline its mapping to a dynamic aspect-oriented middleware platform.

Key words:aspect-oriented;CPS;middleware;feature models;software product line

0 引 言

信息物理系统CPS是一个综合计算、网络和物理环境的多维复杂系统,通过3C(Computation、Communication、Control)技术的有机融合与深度协作[1],实现大型工程系统的实时感知、动态控制和信息服务。CPS实现计算、通信与物理系统的一体化设计,可使系统更加可靠、高效、实时协同,具有重要而广泛的应用前景。近年来,CPS不仅已成为国内外学术界和科技界研究开发的重要方向,预计也将成为企业界优先发展的产业领域[2~5]。

收稿日期:2011-07-21;**修回日期**:2011-10-27
基金项目:国家自然科学基金重大研究计划(90818008);国家自然科学基金项目(60774095,60474072Z);广东省自然科学基金项目(07001774,04009465)
作者简介:吴 帅(1986-),男,硕士研究生,研究方向为软件设计与实时系统;张立臣,博士,教授,研究方向为并行处理、分布式处理、实时系统。

中间件的体系结构被设计为支持许多的应用程序域,中间件需要处理大量的分布式底层应用,因此中间件本身就变成极其复杂。而大部分时间,只有一小部分功能真正得到了使用,出现了许多的冗余功能。中间件应该建立一种方式,它只适合其应用领域。为了实现这一目标,将分散中间件功能以及模块化成可配置的单位。运用面向方面的编程技术可实现中间件横切关注和核心功能关注的并行设计与开发,并且实现良好的模块化,可配置性,部署时或运行时的可剪裁性,从而分布式实时系统中间件的通用性和专用性寻找一个平衡点。

1 面向方面编程技术(AOP)

面向方面编程技术AOP(Aspect-Oriented Programming)可以把非功能需求形成的横切关注点从功能需求形成的核心关注点中分离出来,将非功能需求与功能需求模块化,分别进行设计与编码,最后利用面

向方面的编织技术将两者的代码编织在一起形成最终的系统。面向方面的开发步骤分为三步[6~8]:

(1)方面分解:按照系统的需求,分离出系统核心模块级关注点和横切关注点。

(2)关注点实现:并行开发实现不同的关注点。

(3)方面的重新组合:在这一步里,方面集成器通过创建一个模块单元(方面)来指定重组的规则,重组过程,即织入(weaving)或结合(integrating),则使用这些信息来建构最终的系统。

面向方面编程允许用不同的维度分解软件系统,软件开发者可以使用垂直分解过程建立中间件的基本分解模型,它最优化的提供透明的网络通信便利。然后,使用面向方面技术在不改变现存结构的情况下,将正交设计需求的实现添加到基本模型上。

文中认为在面向方面中间件体系结构设计层面上,至少需要以下的概念才能把面向方面的机制描述清楚,并且不至于过早的把与实现有关的细节描述出来。这些概念包括[9,10]:

连接点(Join Point):是程序运行流程中的一点,包括方法的调用、异常处理的执行等。

切入点(Pointcut):是由切入点正则表达式描述的一系列连接点的组成,匹配连接点的断连。

通知(Advice):一个通知是一个逻辑,这个逻辑由特定的事件触发;它是行为,这个行为能够被插入在调用者和被调用者之间,在一个方法调用者和实际的方法之间,通知是 AOP 真正的关键。

方面(Aspect):是用于实现某个非功能核心关注点的模块单元,它扩展了类的概念,通过了模块化方式实现横切关注点。

织入(Weaving):是把切面连接到其他的应用程序类型或者对象上,并创建一个被通知的对象。

2　中间件平台的特征模型

中间件将遵循一个微内核和服务结构。微内核术语描述的是操作系统设计形式必须在特权模式下执行的代码量保持一个绝对最小值。因此,其他的服务被当做一个单独的模块来建立,并由内核来插入和执行。这样,将获得一个更加模块化和可重用的系统。此外,根据应用的需求将区分强制性基础服务和其他可选服务。

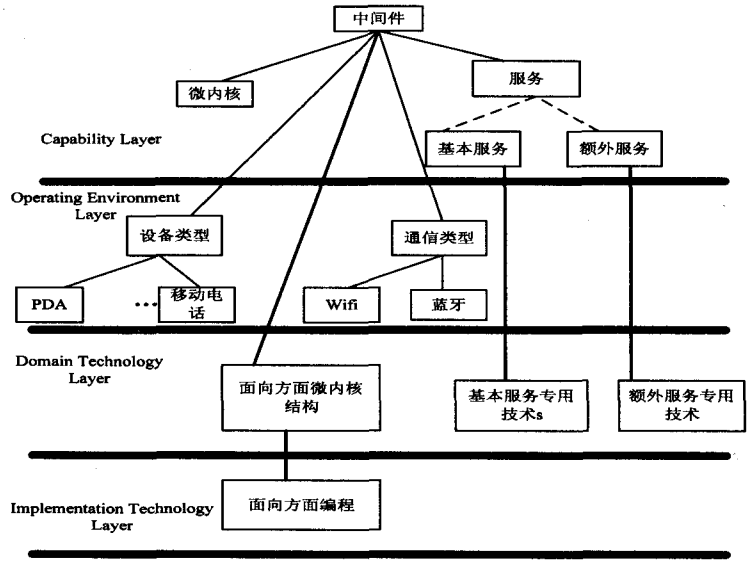

图1　中间件特征模型

如图1所示,中间的功能是由两个强制性(mandatory)特征组成,微内核和服务。而服务包括基本服务和额外服务。基本服务也是一个强制性特征,但额外服务就是一个可供选择的特征。所有的这些特征都在 Capability Layer,图2对这些特征进行更详细地描述。这一层包含所有用户可见的特征。

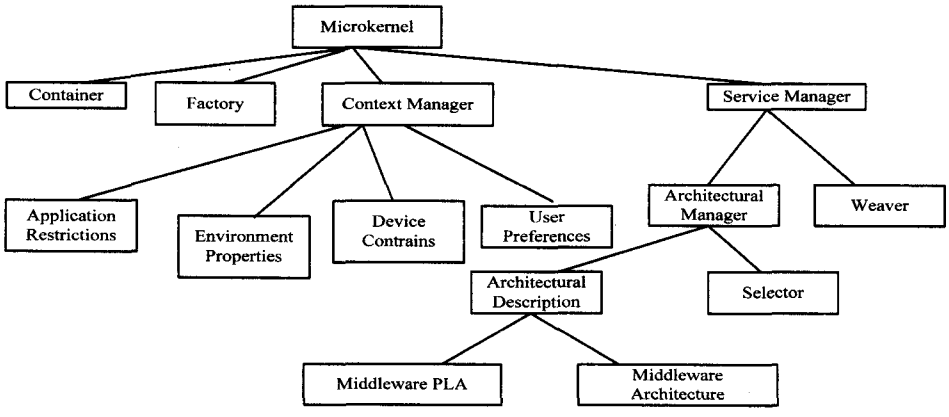

图2　微内核特征模型

中间件可以使用多种设备(如 PDA、移动电话),并允许使用多种通信类型(例如 WiFi、蓝牙)。操作环境层表示使用应用程序的环境,分为两个:设备类型和通信类型。设备类型的特点是:通过不同的设备可以在子功能上作出很多的选择。而通信类型则是通过不同的通信设备可以选择不同的通信方式。在 Domain Technology Layer 的目的是实现中间件作为一种面向方面的微内核结构。另一方面,对每一项服务,使用自己的域技术,但为简单起见,暂时得出两个通用的特

征,基本服务和额外服务。最后,中间件是通过面向方面编程技术来实现的。

图2显示了微内核的详细特征模型。微内核是由容器、工厂、上下文管理器和服务管理器组成。容器是负责搜索所需的服务。工厂元素是对服务进行创建和实例化。上下文管理器是管理上下文的变化。已经发现了四个实体引起上下文的变化。

(1)应用程序可能会发生变化造成需要不同的服务。例如,一个应用程序只在某些时刻是需要安全机制,那么要保证这些时刻可供选择的安全服务是处于有效状态。

(2)环境的变化和一些服务可能不以正常方式执行。例如一个设备的丢失。

(3)系统意识到设备的限制。例如,当电池的电量小于最小允许值时,应用程序中的图像必须做出改变。

(4)用户的喜好也会引起上下文的变化。

微内核的最后一个子功能是服务管理器来管理中间件的体系结构和应用程序的运行。它由两个子功能,结构管理和织入器。最后是一个解释程序,组成所有的中间件和应用程序中的元素。就像面向方面中间件方法,称这个解析程序为织入器。架构说明由两个子特征所组成,中间件产品线架构和这个产品线的一个实例,用来说明中间件产品线的架构说明。而选择器根据上下文属性(比如:设备约束,用户喜好等)从一个产品线生产特殊产品的工具。

基本服务特征模型。所有这些服务都是强制性特征。查找服务(Lookup)搜索哪里的中间件正在运行。探索服务(Discovery Service)搜索其他哪些设备在使用中间件。通讯(Communication)实现所有的中间件元素之间的沟通。设备管理服务控制部署应用程序的设备。最后是容错特征,当中间件元素失去连接时会使用到容错等特征。

图3显示的是可选的额外服务。这些服务都是中间件给予应用程序的设施。安全服务给应用程序提供一个安全的环境,此功能是由许多的子功能所组成的,例如加密或认证。持久性是一种传统服务,允许数据存储在几个不同的地方,以避免丢失数据。错误处理服务提供几种协议,应用程序可以使用它来处理应用错误。上下文感知服务是感知上下文的一切变化并作出回应。最后是定位服务,它用来处理不同的定位技术和数据表示。

3 特征模型映射到面向方面中间件

3.1 特征模型映射到产品线结构

在文献[11]中显示了一种特征模型和面向方面架构描述的映射,特别是xADL方面的扩展[12]。这个扩展显示了一个对称的分解模型,其中一个组件被认为是一个方面,当它在参与了方面的交互。这些交互是放置在连接器上的。当组件参与方面交互时调用"方面化(aspectual)",并且当着连接器表达横切关系时称之为"方面化"。"方面化"组件不仅具有横切特征,同时也解决特征之间的依赖关系。

在这架构中,所有的中间件服务都是横切特征,并且将被映射成方面化组件。在中间件中也只有组件才能成为微内核,并且应用程序也可以被视为其他的常规组件。

作为映射的例子,文中考虑上下文感知服务"方面化"的组成部分作为模型。然后,当组件被指定执行时,它将成为"方面化"的连接器。上下文感知服务将会做出一些调整,例如当一台设备的电池小于某一临界值。在这种情况下,这项服务调用上下文管理微内核组件,为了处理电池的这种约束,这项服务将调用选择器来创建一个新的中间件架构。再比如,当电池发送它当前值和状态给监视器时,上下文感知服务将会检查这些值和状态来重新作出决策。

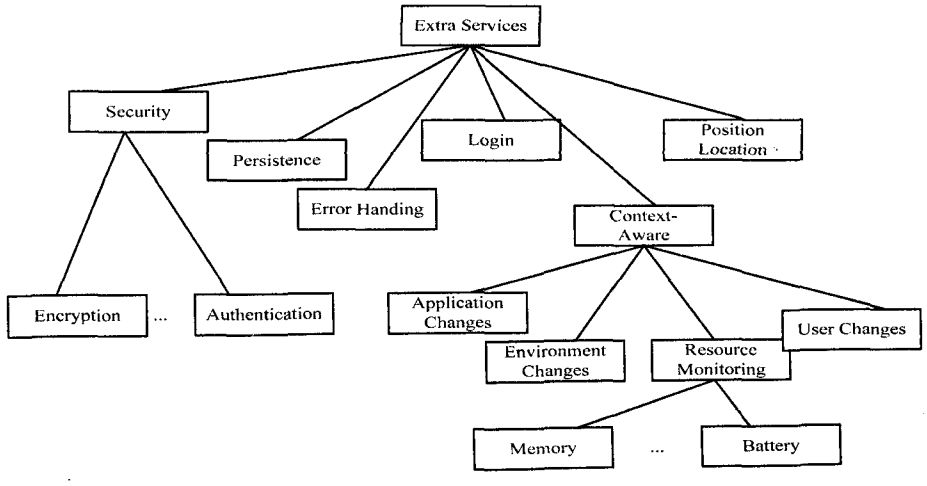

图3 额外服务特征模型

```
< types: connectorType
types:id = " AspConnType" >
    <sendMessage>
    <sourceComp>
    <roles>Battery</roles>
    </sourceComp>
    <targetComp>
    <roles>Monitoring</roles
>
    </targetComp>
    <targetMessages>
    <message name = " value-
Battery" />
```

```
</targetMessages>
<AFTER_SEND>
<concurrent>
<aspect id="context-aware"/>
</concurrent>
<AFTER_SEND>
</sendMessage>
</types:connectorType>
```

上面使用一个已经定义了的 DSL。由于中间件由架构来驱动,为了组合这些中间件元素,DSL 将被作为输入。这个组件在运行时不断改变体系结构说明和中间件,并允许中间件的动态重配置。

3.2　面向方面中间件结构

正如上一节显示的中间件结构,由两个组件(应用程序和微内核)和一些方面化组件(所有的服务)组成。同样,大部分传统中间件平台的架构遵循这样一个层次结构,如图 4 所示。中间件的第一个子层是应用服务器,如安全、错误处理等应用服务。这些服务的下一层是基本服务,如通信、查找等。再下一层是微内核,就像是特征模型描述所示,是由容器、工厂、上下文管理器和服务管理器组成。

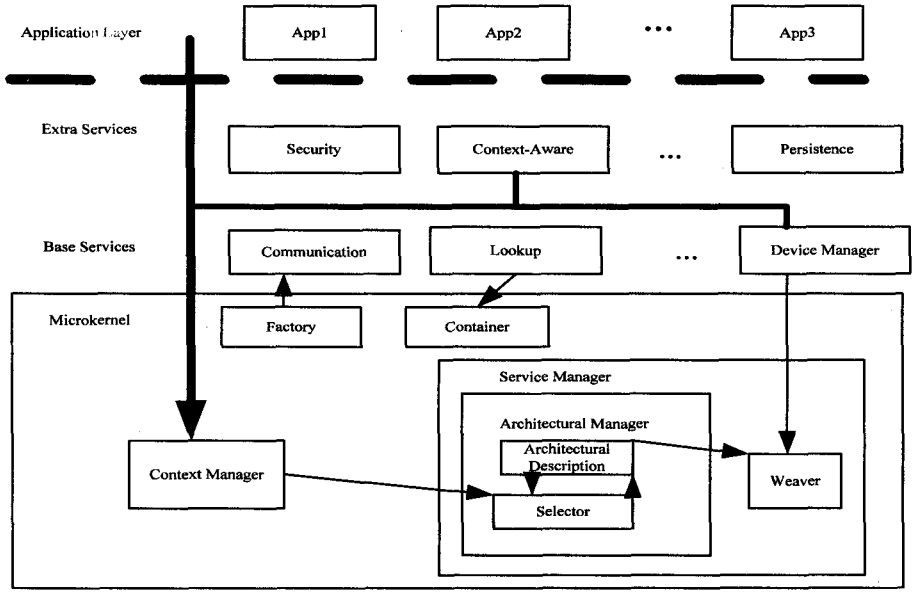

图 4　中间件架构

中间件的产品线结构说明是处于服务管理内部。微内核,通过上下文管理器可以知道上下文属性。有了这些信息和应用程序的体系结构说明,服务管理器自动选择器设备运行,以实例化一个特定的产品线结构的中间件架构。这是一个静态配置中间件。

利用上下文属性,工厂将实例具体选择的服务。为了运行应用程序,服务管理器使用织入器(weaver)来对方面化组件,相关的服务和组件(微内核和应用程序)之间进行组合。如果上下文有所变化时(用户属性,设备约束等),微内核将重新使用选择器,并实例一个新的产品线中间件。负责执行上下文感知访问上下文管理器是额外服务。这是中间件的动态重配置。

4　结束语

在文中展示了如何使用一个中间件方法来解决对分布式信息物理系统中的一些问题。为了使信息物理系统中间件具有更强的可重配置和适应性,中间件必须遵循具有一个微内核和一些服务的层结构。此外,我们采用一个面向方面的中间件,它允许动态重配置,因为重配置可以处理上下文感知等问题。重点是中间件产品(middleware product)的特征模型设计,之后将用于生产这中间件架构产品线。然后,也概况了特征模型和体系结构之间的映射,表明中间件架构遵循层方法。只是信息物理系统中间件发展的初始阶段,今后的工作中,必须继续设计面向方面的 DSL,将特征模型映射到 DSL 架构中。

参考文献:

[1] 何积丰. Cyber-physical Systems[J]. 中国计算机学会通讯,2010,6(1):25-29.

[2] Lee E A. Cyber Physical Systems: Design Challenges [R]. California: University of California, Berkeley, 2008.

[3] Wikipedia. Cyber-Physical Systems[EB/OL]. [2010-06-02]. http://en.wikipedia.org/wiki/Cyber-physical-system.

[4] Wing J M. Cyber-physical systems research challenges [C]//Keynote Speech: Cyber Physical Systems Week (CPSWEEK). [s.l.]:[s.n.],2010.

[5] CPS Steering Group. Cyber-Physical System: Executive Summary[R/OL]. [2008-10-25]. http://varma.ece.clnu.edu/summit/index.html.

[6] 熊峻锋. 基于面向方面技术的软件安全框架的研究[D]. 重庆:重庆大学,2004.

[7] 李　行,张立臣. 面向方面的实时系统中间件[J]. 计算机技术与发展,2008,18(7):8-10.

[8] 刘东星,张立臣,高　娜. 面向方面在实时系统中间件中的应用[J]. 计算机技术与发展,2010,20(7):40-43.

[9] 康　蕊,张立臣. 基于 AOP 的 QoS 中间件自适应机制研究

(下转第 84 页)

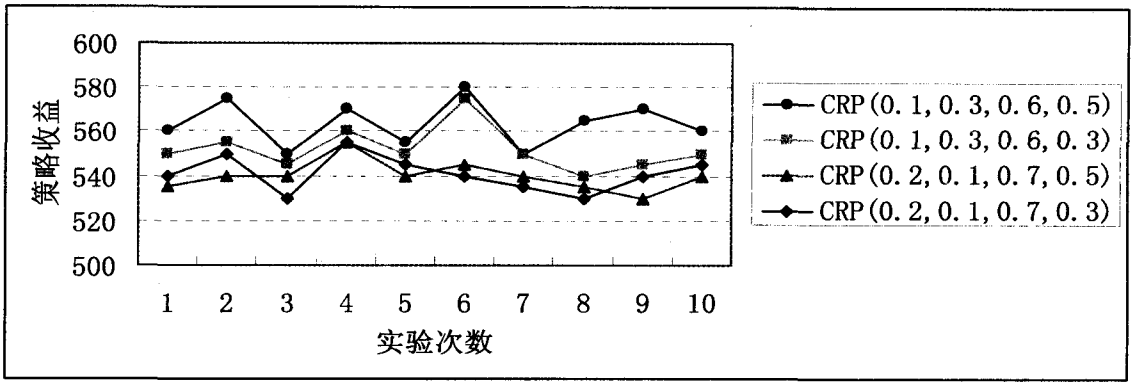

图3 调整CRP算法参数所获得的策略收益值

同。其中，$(\alpha,\beta,\gamma,\lambda)=(0.1,0.3,0.6,0.5)$ 时，CRP算法能获得更好的策略收益值。由此可进一步说明，基于CRP算法的资源提供策略能够在满足用户任务请求的前提下降低运营成本，获得更大收益值，也证明了这一策略的有效性。

4 结束语

在云计算环境下，资源提供策略中资源的合理选择是在满足用户请求的同时又能有效降低云服务提供者的运营成本的一项重要环节。通过实验也可以看出，文中提出的基于CRP算法的资源提供策略根据云计算中资源的特点，充分考虑资源的计算能力、可靠性和计算成本三点因素，能够得到更好的策略收益，同时也说明了这一策略的有效性。

参考文献：

[1] 王庆波，金 淬，何 乐，等. 虚拟化与云计算[M]. 北京：电子工业出版社，2009：110-180.

[2] 刘 鹏. 云计算[M]. 第2版. 北京：电子工业出版社，2011：1-15.

[3] 陈 全，邓倩妮. 云计算及其关键技术[J]. 计算机应用，2009(9)：2562-2566.

[4] Armbrust M, Fox A, Griffith R, et al. Above the Clouds: A Berkeley View of Cloud Computing[J]. Communications of the ACM, 2010, 53(4): 50-58.

[5] Chen Yang, Wo Tianyu, Li Jianxin. An Efficient Resource Management System for Online Virtual Cluster Provision [C]//2009 IEEE International Conference on Cloud Computing. Washington, DC, USA: IEEE Computer Society, 2009: 72-79.

[6] 田冠华，孟 丹，詹剑锋. 云计算环境下基于失效规则的资源动态提供策略[J]. 计算机学报，2010，33(10)：1859-1872.

[7] Fu Song. Failure-aware resource management for high-availability computing clusters with distributed virtual machines [J]. Journal of Parallel and Distributed Computing, 2010, 70(4): 384-393.

[8] Rosenblum M, Garfinkel T. Virtual machine monitors: current technology and future trends[J]. IEEE Computer Society, 2005, 38(5): 39-47.

[9] Li Yunfa, Li Wanqing, Jiang Congfeng. A Survey of Virtual Machine System: Current Technology and Future Trents[C]// 2010 Third International Symposium on Electronic Commerce and Security (ISECS). Guangzhou, China: IEEE Computer Society, 2010: 332-336.

[10] Schroeder B, Gibson G A. A Large-scale Study of Failures in High-performance Computing Systems[J]. IEEE Transactions on Dependable and Secure Computing, 2010, 7(4): 337-350.

[11] Fu Song, Xu Chengzhong. Exploring Event Correlation for Failure Prediction in Coalitions of Clusters[C]//SC'07 Proceedings of the 2007 ACM/IEEE Conference on Supercomputing. New York, USA: ACM, 2007: 1-12.

[12] Clark C, Fraser K, Hand S, et al. Live migration of virtual machines[C]//NSDI'05 Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation. Berkeley, CA, USA: USENIX Association, 2005: 273-286.

(上接第79页)

[J]. 计算机科学，2008，35(8)：287-289.

[10] 刘敬勇，张立臣，钟 勇. 面向方面的中间件[J]. 计算机技术与发展，2008，18(8)：68-71.

[11] Fuentes L, Gamez N. Adding aspects to xadl 2.0 for software product line architectures [C]//VaMoS. [s. l.]: [s. n.], 2007.

[12] Dashofy E, van der Hoek A, Taylor R. A comprehensive approach for the development of modular software architecture description languages[J]. ACM Transactions on Software Engineering and Methodology, 2005, 14(2): 199-245.