轻量级构件部署与配置工具的研究与实现

史殿习[1] 丁 博[1] 崔 巍[2] 张一明[1] 王怀民[1]
(国防科学技术大学计算机学院 长沙 410073)[1] (江南计算技术研究所 无锡 214000)[2]

摘 要 普适计算应用运行在高度异构、复杂多变的环境中,如何便捷、高效地部署与配置此类系统,是目前普适计算领域面临的挑战之一。为有效支持构件化普适计算应用的部署与配置,基于模型驱动的思想,遵循OMG的轻量级构件规范和构件部署与配置规范,设计实现了一个轻量级构件部署与配置工具StarDCTool。StarDCTool能够通过目标运行平台建模对计算环境的异构性进行支持,通过部署计划建模支持构件化应用的部署和配置过程的重用,进而自动化地生成与部署和配置相关的元数据。通过具体的构件化导航应用案例,验证了StarDCTool的功能和特点。

关键词 普适计算,构件开发,模型驱动,部署与配置
中图法分类号 TP311 **文献标识码** A

Research and Implementation of Deployment and Configuration Tool for Light-weight Components

SHI Dian-xi[1] DING Bo[1] CUI Wei[2] ZHANG Yi-ming[1] WANG Huai-min[1]
(School of Computer, National University of Defense Technology, Changsha 410073, China)[1]
(Jiangnan Institute of Computing Technology, Wuxi 214000, China)[2]

Abstract Pervasive computing applications run in highly heterogeneous and dynamic changing environments. Therefore, it is a great challenge for pervasive computing to deploy and configure this kind of applications conveniently and efficiently. In order to support the deployment and configuration (D&C) of component-based pervasive computing applications, this paper proposed the design and implementations of a D&C tool for light-weight components, StarDCTool, which is based on the model-driven paradigm and complied with OMG's light-weight component D&C specification. StarDCTool deals with the heterogeneity by modeling the target platforms and enables the reuse of the D&C process of component-based applications by modeling the deployment plan. Based on those modeling results, StarDCTool can generate the meta-data related to the D&C process automatically. We validated the functions and features of StarDCTool by a component-based navigational application.

Keywords Pervasive computing, Component deployment, Model-driven, Deployment and configuration

1 引言

普适计算追求"无时无处不在又不可见"的计算[1],具备泛在性和便捷性特点,其环境由大量异构设备(如服务器、PC、PDA、智能手机、各种传感器等)和各种网络(Internet、Wifi、红外等)等构成,且在运行时可能发生动态变化。如何在这种环境下部署和配置分布式应用系统,使其能够为用户提供便捷的服务,是目前普适计算领域面临的挑战之一[2]。

现有面向普适计算的软件工程实践,由于具有提高软件重用、可动态配置部署等特点,广泛采用构件技术[3]特别是针对资源受限设备的轻量级构件模型。例如,国际对象管理组织OMG在对CCM(CORBA Component Model)规范进行裁剪的基础上发布了轻量级CORBA构件模型(Lightweight CCM)规范[4],以支持分布式实时嵌入式应用的开发;作为轻量级、面向服务的构件框架,OSGi技术的应用已经突破了传统家庭网关的范畴[5];Gaia[6],PCOM[7],One. world[8],SATIN[9]等项目都内置了某种形式的轻量级构件模型。然而,现有研究较少考虑如何针对普适计算环境异构、复杂多变的特点来实现构件化系统的高效部署和配置。

模型驱动开发[10]是一种以模型为中心的新型软件开发模式,其核心是在模型层面实现软件重用以及实现代码的自动生成。在模型驱动开发中,应用的部署与配置过程可以通过模型来描述,与具体软硬件平台解耦,因而可有效应对平台异构性;另一方面,模型驱动开发可以支持应用部署与配置过程的重用,环境发生变化时只需对模型进行部分修改。因此,针对现有研究所面临的挑战,我们将构件技术与模型驱动开发技术有机结合,以模型集成计算[11](MIC)为核心思想,设计实现了一个轻量级构件部署与配置工具StarDCTool。StarDCTool能够通过对目标运行平台建模来支持计算环境的异构性,通过部署计划建模来支持构件化应用部署和配置过程的重用,进而自动化生成与部署和配置相关的元数据,有效支持普适计算应用的快速部署与配置。

到稿日期:2010-07-01 返修日期:2010-11-15 本文受国家核高基重大专项课题(2009ZX01043-001),863国家重点课题(2007AA010301)资助。
史殿习(1966-),男,博士,副研究员,CCF会员,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,主要研究方向为分布计算、普适计算等;崔 巍(1985-),男,硕士,主要研究方向为分布计算等;张一明(1982-),男,硕士,主要研究方向为分布计算等;王怀民(1964-),男,博士,教授,CCF会员,主要研究方向为分布计算、网络安全等。

本文第 2 节阐述背景和相关研究；第 3 节阐述 StarDCTool 系统结构；第 4、5 节分别描述 StarDCTool 的元模型以及模型映射机制；第 6 节以一个具体的应用案例，对工具进行应用验证；最后对全文进行了总结。

2 背景和相关研究

本文工作以模型驱动开发技术为基础，以支持轻量级 CORBA 构件模型在异构、动态环境下的部署和配置为目标。

2.1 模型驱动开发和模型集成计算

模型驱动开发技术的核心是在模型层面上实现重用以及代码的自动生成，从而实现系统的快速开发和部署，其典型代表是模型驱动结构 MDA(Model Driven Architecture)[12]和模型集成计算 MIC(Model Integrated Computing)[11]等。其中，模型集成计算强调模型的领域相关性，通过领域相关建模语言 DSML(Domain Specific Modeling Language)[11]来描述特定领域的元模型及模型转换，领域的元模型描述了特定领域的语法、语义及语法、语义的映射规则，基于领域的元模型可以进一步建立应用的模型，进而通过模型转换规则和转换机制将领域模型转换为应用系统的部署与配置描述文件、系统可执行的代码等等。

基于模型集成计算技术开发的核心问题是设计领域相关的建模语言 DSML。因此，如何建立一个支持面向普适计算环境的构件化应用部署和配置的 DSML，是本文重点研究和解决的问题之一。本文后续工作将在支持领域相关建模语言 DSML 开发、集成与配置的环境 GME[13]上实现。

2.2 轻量级 CORBA 构件模型

为了适应分布式实时嵌入式系统资源受限的特点，OMG 在对 CCM 规范裁剪以后发布了轻量级 CORBA 构件模型(Lightweight CCM)规范[4]。在对 CCM 规范进行裁剪的过程中，OMG 主要遵循以下一些原则：去除冗余；去除嵌入式环境下较少使用的机制，包括持久化、事务、安全和部分反射接口；在部署、用户源码和互操作上与 CCM 兼容。

将轻量级 CORBA 构件模型作为普适计算应用软件的模块组织方式的优点包括：(1)可应用于资源受限环境；(2)接口定义的对称性和松耦合性。轻量级 CCM 构件在接口一级不仅支持对所提供服务的定义，也支持对所需要服务的严格定义，这种对称性提供了更多的构件元信息，从而可方便地支持普适计算环境下所需的动态配置、容错等。(3)对分布式应用部署的支持。轻量级 CCM 构件良好的自包含性带来了丰富的自描述信息，从而使应用程序被映射到特定普适计算空间时适应计算环境。

2.3 构件部署与配置

为了规范基于构件的软件开发过程，OMG 把基于构件的软件开发方法划分为构件开发、运行平台定制管理、构件应用部署 3 个阶段：在构件开发阶段，由构件开发者开发构件，构件组装者通过组装已有构件形成复合构件，构件打包者把多个同种功能构件的不同实现打包在构件包里；在运行平台定制管理阶段，域管理员制定系统运行的目标平台，如计算结点、网络资源和共享资源等；在构件应用部署阶段，构件部署与配置人员根据构件库中的构件包和运行平台的信息制定部署计划，把构件库中的构件部署到运行平台上，启动应用程序。

相应地，OMG 制定了构件部署与配置规范[14](以下简称 D&C 规范)，使得开发人员可以通过组装已经存在的构件来构造复杂的大规模及超大规模的应用系统，可以抽象出应用程序在部署和配置过程中所涉及的公共元素，在组装与部署基于构件的应用时，重用这些元素，进而动态地、自动化地把构件部署到目标平台上。

3 StarDCTool 体系架构

StarDCTool 将模型集成计算的思想引入构件部署与配置过程中，以 GME 为开发环境，遵循 OMG 的 D&C 规范建立了构件部署与配置工具。StarDCTool 的体系架构如图 1 所示，自底向上分别是基础设施层、部署与配置元模型层、应用模型层以及模型解释器层。

(1)基础设施层即 GME 支撑平台，提供其他各层所需的基础设施，包括可视化建模环境、元建模语言(如 UML 与 OCL 等)、约束检验部件、模型访问接口以及模型数据存储等。

(2)元模型层基于元建模基础设施构建符合 D&C 规范的部署与配置元模型，该元模型以 UML 类图和 OCL 语言组合实现，具有精确的结构语义及模型约束语义。

(3)应用模型层是在部署与配置元模型约束下建立的用户系统模型。

(4)解释器层即部署与配置模型解释器，基于模型访问接口，完成从系统模型到代码数据的自动映射。

元模型层、系统模型层以及约束管理部件实现对静态语义约束即应用模型结构的正确性和有效性的支持，而应用模型、解释器及模型访问接口部件实现对动态语义映射即从应用模型到可运行代码到自动生成的支持。构件部署与配置相关的元模型、应用模型以及模型间解释器即构成了 StarDCTool 工具的可视化部署与配置建模语言(Visual Deployment and Configuration Modeling Language)。

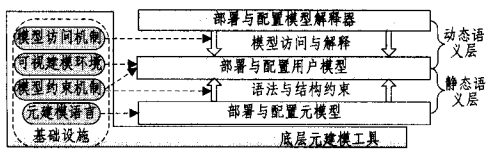


图 1 StarDCTool 体系结构

4 构件部署和配置元模型

元模型是对模型在更高层次上的抽象和描述。若要在 GME 基础设施之上为遵循轻量级 CCM 规范的构件化应用的部署与配置提供有效支持，首先应遵循 OMG 的 D&C 规范和轻量级 CCM 规范建立其元模型。为此，我们遵循 OMG 的 D&C 规范以及轻量级 CCM 规范，基于 GME 开发环境中的元建模语言，建立了完整的面向轻量级 CCM 构件模型的部署与配置元模型体系，进而通过基于对象约束语言 OCL[15]在元模型上施加语义约束。

4.1 StarDCTool 元模型体系

StarDCTool 元模型体系包括轻量级 CCM 构件模型元模型、目标平台元模型和部署计划元模型 3 类元模型。

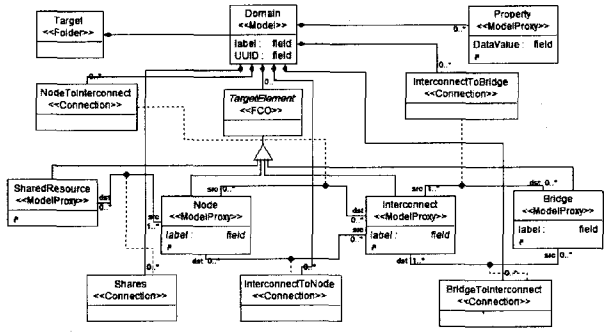
(1)轻量级 CCM 构件元模型

基于 GME 环境，我们建立了完整的轻量级 CCM 构件模型的元模型，包括轻量构件定义元模型、构件实现元模型以及构件组装元模型等。轻量级 CCM 构件拥有 5 种类型的端口：刻面、接插口、事件源、事件槽和属性。在运用 GME 建模

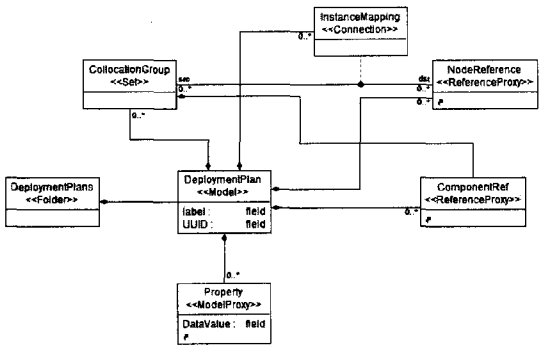
过程中，将轻量级 CCM 构件定义为 Model 类型，刻面、接插口、事件源和事件槽定义为 Reference 类型，同时为了方便描述轻量级 CCM 构件的配置信息，将轻量级 CCM 构件的属性定义为 Model 类型。

(2)目标平台元模型

D&C 规范中对运行平台进行了高度抽象，将构件化应用程序的运行环境抽象为域，把实际运行的 PC，PDA 等具体设备抽象为结点，结点之间的连接抽象为网线，无线网络抽象为连接器；连接器之间的连接，诸如路由器、集线器、网桥等抽象为桥接器。结点、连接器和桥接器中都拥有资源，如结点的资源可能是 CPU 的速度、内存的大小、操作系统的类型等，而连接器的资源则是线路中带宽的大小等，桥接器的资源是它拥有多少个端口、每个端口的带宽等等，共享资源表示诸如打印机这样的共享设备。图 2(a)为遵循 D&C 规范的目标平台的元模型片段。该元模型是对域、结点、连接器、桥接器及共享资源等进行抽象和描述，通过对运行目标平台的抽象建模，可以有效解决运行平台多样异构问题。



a) 目标平台元模型片断



b) 部署计划元模型片断

图 2 StarDCTool 元模型片断

(3)部署计划元模型

部署计划是为了描述构件如何运行在目标平台上，它具体包括所需实现的接口信息、每个结点上的构件实例信息、构件之间的交互信息、结点上的具体实现制品描述信息、非功能的配置信息及一些本地约束等信息。根据这些信息部署引擎，可以实现自动化的应用部署。部署计划元模型定义了构件实例与运行节点的具体映射规则，因此元模型中包括构件引用(ComponetRef)和节点引用(NodeReference)，如图 2(b)所示。构件实例与节点实例之间使用 InstanceMapping 类型连接，表示构件实例的绑定关系。

4.2 模型约束与检查机制

构件部署与配置元模型建立过程中另一个需重点研究和解决的问题是如何提供相应的检验机制。按照元模型中所表达的语义约束，对用户所创建的模型进行合法性检验，以确保部署与配置模型的有效性。GME 元建模语言基于 UML，并且支持兼容 OCL1.4 标准的不变量类型约束的表达和检查。为有效支持模型构建时有效性的即时检查，我们设计实现了一个模型构建即时纠错机制，如图 3 所示。在元模型建模过程中，不同模型以及模型之间关联关系的约束条件以 OCL 表达式的方式进行描述，并将其附加在相应的元模型之上。同时将对模型的约束检验与特定建模事件(如创建模型或连接模型)进行绑定，当发生某种建模事件时，由 OCL 管理器自动化地对模型进行约束检验。如果不满足约束条件(约束检查结果返回为 false)，则会向用户提示模型的错误信息。

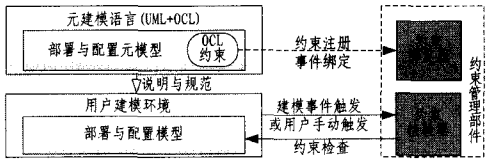


图 3 StarDCTool 元模型上的 OCL 约束及其检查机制

5 模型转换与解析

基于上一节建立的构件部署与配置元模型，应用开发者可以在 GME 环境中构建具体的应用模型，包括构件接口模型、构件实现模型、构件的组装模型、构件的打包模型、配置模型、运行平台模型以及部署模型等。如何将用户定义的这些合法有效的、可视化的模型自动化转换为与运行平台相关的可执行的代码及元数据，是构件部署与配置建模工具需要解决的另外一个关键问题。我们设计实现了相应的模型转换与解释机制，如图 4 所示。

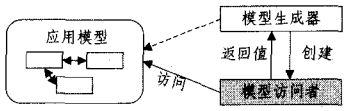


图 4 模型解析机制

此机制的核心是模型解析器，其核心功能是创建相应的访问者(Visitor)，对用户构建的模型目录树进行访问。访问者能够根据预置的数据框架生成相应的 XML 元数据或代码返回给解释器。解释器将这些访问者返回的数据加以综合处理，最终生成可以直接部署在运行平台上的元数据和二进制代码。不同类型模型的访问方法由不同的 Vistor 类分别实现，解释器的业务逻辑只需关注模型的结构信息。

模型解释器遵循 OMG 的 XMI 规范[16]以及 OMG 的 UML Profile for CORBA and CCM 规范[17]，通过 GME 提供的 API 接口与模型实例进行交互，生成以 IDL 和 XML 形式描述的模型实例的描述文件，具体包括：构件接口描述，描述一个构件的接口、端口、属性等等；构件实现描述，描述一个构件接口的特定实现，同时描述构件端口之间的连接关系等；构件包描述，描述一个构件的多个实现，如在不同操作系统下的实现之类的信息等。构件包配置描述，描述一个构件包的特定配置、需要什么元数据等信息。构件运行平台的描述，描述应用系统的运行平台、描述平台包括的可计算结点、网络信息、共享资源等。部署计划描述，描述应用系统各个子构件具体运行在平台哪个结点上、每个结点上有什么构件实例等信息。

6 应用案例

本节通过部署与配置一个构件化导航应用 NavAppl 来展现部署与配置工具的功能和特点，验证其有效性。该应用

基于我们自主研发的普适计算中间件平台 UbiStar[18] 开发，由定时器、GPS 定位、中心处理、导航显示 4 个构件组成(见图 5)。定时器 Timer 构件周期性地产生 Timeout 事件；GPS 构件接收定时器的周期触发事件，从物理设备获取当前位置信息，并产生一个更新事件 data_avail；中心处理构件 AirFrame 接收 GPS 构件的数据，更新事件的触发，使用 get_data 方法从 GPS 构件获取当前最新原始位置数据，并转换成可被显示设备使用的数据；导航显示构件 NavDisp 将数据显示在导航终端设备之上。

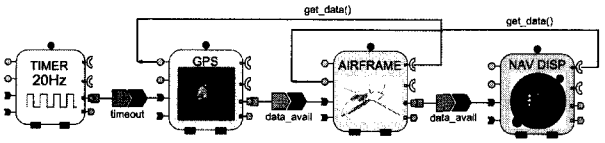
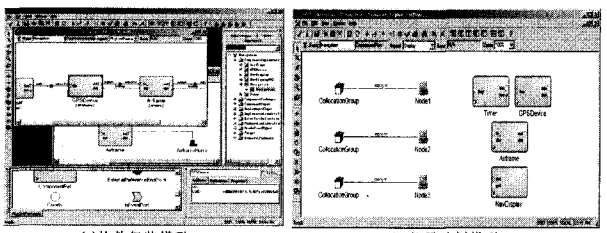


图 5　导航应用中构件之间的关系

(1)建立应用模型

如前所述，NavAppl 的部署与配置过程包括构件定义、构件组装、目标平台建模以及制定部署计划等。图 6(a)给出了使用 StarDCTool 建立的 NavAppl 应用的构件组装模型，图 6(b)给出了使用 StarDCTool 建立的 NavAppl 部署计划模型。



(a)构件组装模型　　(b)部署计划模型

图 6　NavAppl 应用模型片断

(2)模型检验

根据元模型中的 OCL 语义约束，能够组装起来的功能端口(如 Facets 和 Receptacles)或事件端口(如 Event Sink 和 Event Source)必须具有相同的类型。在本例中，Timer 构件的事件输出端口只能与 GPSDevice 构件的事件输入端口连接，因为二者具有相同的事件类型 timeout。用户在尝试连接 Timer 构件的输出事件端口(Event Sink)timeout 与 NaviDisp 构件的事件输入端口的 data_refreshed 时，StarDCTool 工具会出现错误提示，并给出修改建议，如图 7 所示。

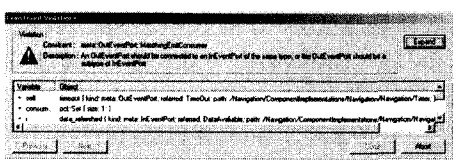


图 7　模型检验的错误输出

(3)模型映射

StarDCTool 可以根据应用模型自动地生成以 IDL 和 XML 形式描述的模型的描述文件，如构件接口描述文件、构件实现描述文件、构件运行平台描述文件以及部署计划描述文件等。我们将 4 个构件分为 3 个组：Timer 构件实例和 GPS 构件实例运行在 Node1 上，AirFrame 构件实例运行 Node2 上，NavDispl 构件实例用性在 Node3 上。将所生成的描述文件输入到构件运行平台上，启动应用程序。

结束语　轻量级构件部署与配置工具 StarDCTool 的核心功能是将构件部署和配置过程以图形的形式表现，以模型的形式描述构件模型、目标运行平台模型以及应用的部署计划模型等，根据这些模型自动地生成部署与配置相关的元数据，进而基于这些元数据将应用部署到系统中的各个节点之上。相较于 K2-CCM[19]，MicoCCM[20] 等工具，我们所开发的 StarDCTool 工具的最大优势在于通过模型驱动的思想屏蔽了大量的底层细节；部署与配置过程与平台无关；通过对模型添加 OCL 实现静态语义，实现系统模型的早期验证；通过对层次性组装的支持，可以应对普适计算系统的大规模复杂特性，能自动化地生成构件描述文件、部署和配置文件等。其提高了开发效率，因而具有更好通用性、扩展性和更强的可演化性。

参 考 文 献

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

[2] Rigole P, Clerckx T, Berbers Y, et al. Task-driven automated component deployment for ambient intelligence environments [J]. Pervasive and Mobile Computing, 2007, 3(3): 276-299

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

[4] Object Management Group. CORBA Component Model Specification[S]. v4. 0, 2006

[5] OSGi Alliance, OSGi Service Platform Release 4. 2[S]. 2009

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

[7] Becker C, Handte M, Schiele G, et al. PCOM-A Component System for Pervasive Computing. IEEE Computer Society, 2004

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

[9] Zachariadis S, Mascolo C, Emmerich W. The SATIN component system-a metamodel for engineering adaptable mobile systems [J]. IEEE Transactions on Software Engineering, 2006, 32(11): 910-927

[10] Mellor S J, Clark A N, Futagami T. Model-driven development [J]. IEEE software, 2003, 20(5): 14-18

[11] Sztipanovits J, Karsai G. Model-integrated computing[J]. Computer, 1997, 30(4): 110-111

[12] Object Management Group. Model Driven Architecture[S]. v1. 0, 2003

[13] Ledeczi A, Maroti M, Bakay A, et al. The Generic Modeling Environment[C] // Workshop on Intelligent Signal Processing. 2001

[14] Object Management Group. Deployment and Configuration of Component-based Distributed Applications Specification[S]. v4. 0, 2006

[15] Object Management Group. Object Constraint Language[S]. v2. 0, 2006

[16] Object Management Group. The XML Metadata Interchange [S]. v2. 1, 2005

[17] Object Management Group. UML Profile for CORBA and CORBA Components Specification[S]. v1. 0, 2008

[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] The MICO CORBA Component Model Project[EB/OL]. http: // www. fpx. de/MicoCCM

[20] The K2-CCM Project[EB/OL]. http: // www. icmg. nu/corp/ccm/ ccm. overview. asp