

# 基于.NET平台的软件构件开发方法研究

胡顺扬, 瞿有甜, 周波, 刘芳

(浙江师范大学数理信息学院, 浙江金华 321004)

**摘要:**随着网络技术的进一步发展,分布式构件技术已经成为建立基于网络的服务应用框架和实现软件复用的核心技术。在分析现有构件开发方法和当前主流的软件构件技术的基础上,提出了一种基于.NET平台的分布式构件开发方法,并结合实例加以具体说明。应用实践表明,利用该方法开发分布式软件,具有开发周期短、成本费用低、维护方便、所开发的系统具有较好的适应动态演化的能力等优点。

**关键词:**.NET;分布式构件;接口

**中图分类号:**TP311.52

**文献标识码:**A

**文章编号:**1673-629X(2008)11-0058-04

## Study of Development Approach of Software Components Based on .NET Platform

HU Shun-yang, QU You-tian, ZHOU Bo, LIU Fang

(College of Mathematics, Physics and Information Engineering, Zhejiang Normal University, Jinhua 321004, China)

**Abstract:** With the development of networks, distributed component technology has become the core technologies of establishing service application framework and software reuse. Analyses the existing components of the development approach and the current mainstream software technology, then gives a development methods based on .NET platform with a reality example. Application showed that use the technology above to develop the distributed software can have a short development cycle, low-cost, easy maintenance, and the whole system can have a better ability to adapt to the dynamic evolution.

**Key words:** .NET; distributed component; interface

### 0 引言

传统的软件开发方法有很多种,按开发基本思想来说分为结构化法、生命周期法、原型化法等,然而这些方法在软件的开发过程中暴露出开发周期长、重复劳动多、维护成本高、系统升级复杂和生产效率低等严重的缺陷,这些缺陷制约了软件工业化的发展。在这种背景下,软件构件技术应运而生并迅速发展。通过构件技术,可以解决面向对象技术无法使大量的结构相似的应用程序得到重用的矛盾,全面应用对象技术与概念,成为开发出高效、低成本应用程序的重要实现途径<sup>[1]</sup>。构件技术对软件复用和大规模的软件开发具有非常重大的意义,已成为软件界的研究热点。

随着 Internet 的飞速发展和企业(事业)信息化程度的不断提高,网络环境下的分布式系统已成为目前

计算机应用系统研究和开发的主流,但由于其固有的分布性、异构性和自治性,使得分布式系统开发的难度大大增加。而.NET作为一种分布式对象技术能有效地解决分布式对象和异构系统之间的访问,将.NET分布式对象技术与传统的软件构件开发方法相结合,能有效地提高软件生产效率和产品质量,有利于软件的维护和更新。

### 1 软件构件

构件是一个功能单元,符合构件模型(或具有规范接口),允许不同构件开发商开发的构件进行组装<sup>[2]</sup>。它主要是由接口构成的,通过接口,把内部的实现细节与外界交互的实现分离开,有利于降低对构件理解的难度,同时避免了由于对内部所作的修改而可能带来的负面影响,构件接口也是构件使用者理解构件和构件组装的桥梁。

由此可见,接口是构件的核心,是构件间信息通信的机制。只要定义了统一的接口标准,各种语言开发的软件就可互操作了。

收稿日期:2008-02-29

基金项目:浙江省自然科学基金资助项目(Y106469)

作者简介:胡顺扬(1983-),男,硕士研究生,研究方向为软件工程、构件技术;瞿有甜,教授,研究方向为构件技术、面向 Agent 的软件工程、数据库技术。

### 1.1 基于构件的软件开发过程

传统的软件开发过程在重用元素、开发方法上都有很大的不同。虽然面向对象技术在一定程度上实现了软件重用,但是,只实现了类和类继承的重用。基于构件的软件开发方法能最大可能地重用已有的构件模块,这些构件模块可能是在不同的时间、由不同的人员开发的,并有各种不同的用途。基于构件的软件开发使得使用这些构件对象模型的构件开发者可以像搭积木一样快速构造应用程序。基于构件的软件开发方法一般包括需求分析、总体设计、构件部署、具体化构件、连接、生成代码、产生最终目标代码<sup>[3]</sup>。

### 1.2 现有主流构件技术概述

目前,业界使用比较广泛的构件技术,有以下几种:公共对象模型(Common Object Model, COM), JavaBean, 公共对象请求代理体系结构(The Common Object Request Broker Architecture, CORBA), 和 .NET。

COM 是由 Microsoft 公司推出的构件接口标准,目前已有大量的基于 COM 的构件可复用。COM 是一种二进制标准的构件技术,实现简单、实用,但其与运行环境有关。

CORBA 是对象管理组织(OMG)1990 年为了解决分布式异构构件和硬件环境下对象之间的互操作问题而首次提出的,是一种以 IDL 为桥梁,基于 ORB 中间件的构件技术。CORBA 技术是以 IDL 为标准的,与实现构件接口的语言、软件平台和硬件平台无关,但它无法支持 Internet 上大量的移动计算的需求。一个构件的实现是与硬件平台相关的,即一个构件的实现是不能在网上移动的。

JavaBean 是一种能提供在网上移动的技术,它通过 ByteCode 技术,提供在“任何地方运行,任何地方重用”的功能,但它却是与语言相关的。

而 .NET 就是借鉴 Java,采用 XML,并远远超出了 Java 新的体系。它是建立在公共语言运行库(Common Language Runtime, CLR)的跨语言属性和 .NET 框架的丰富类库的基础上,对于分布式或嵌入式系统的开发任务,还提供了远程和跨平台调试。

## 2 基于 .NET 平台的软件构件开发

.NET 是一个在 J2EE 之后出现的平台,它大量地吸收了 Java 平台,甚至是 J2EE 平台的优点,具备了强大的跨平台能力,支持多种开发语言,还允许创建各种

各样的应用程序,是一个优秀的应用开发和部署平台。

### 2.1 .NET 的技术架构

构件化软件的开发是面向需求的,即设计者集中精力于业务逻辑本身,而不必为分布式应用中的通信、效率、互操作、可靠性等大量与业务无直接关系但又非常重要的问题,而耗费大量的精力,理想的构件开发平台在这些方面应为构件软件提供良好的运行环境。

公共语言运行时(CLR)作为 .NET 构件模型的基础设施为 .NET 构件提供了一个理想的运行环境。CLR 使用应用程序域来隔离应用程序,每个应用程序域可加载多个应用程序集。在分布式构件开发中,不同的应用程序域的程序集的交互可通过远程处理机制来完成。NET 的远程处理为进程间通信提供了一种抽象的方法,它将可远程处理的对象与特定客户端或服务端应用程序域以及特定的通信机制隔离开来。因此很灵活且很容易自定义。可以用一种通信协议替换另一种通信协议,或者用一种序列化格式替换另一种序列化格式,而不必重新编译客户端或服务器。图 1 说明了 .NET 的分布式构件的主要组成结构。

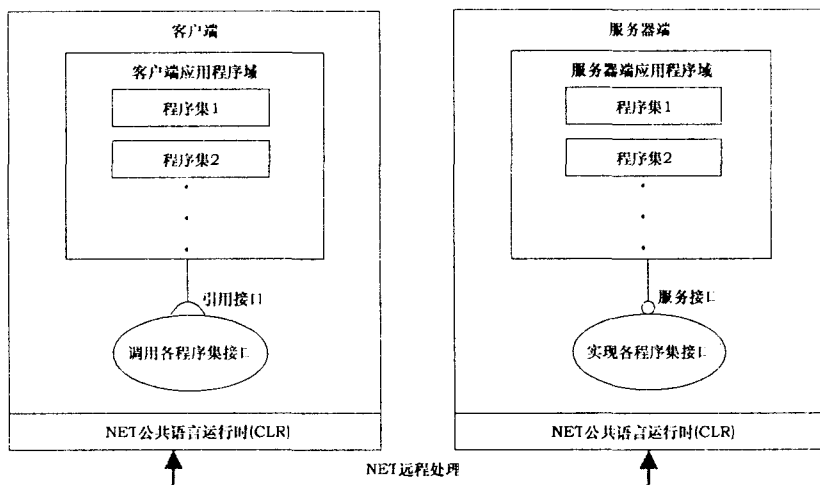


图 1 .NET 的分布式构件的主要组成结构

### 2.2 程序集(Assembly)

程序集(assembly)是构成一个逻辑单元的一个或多个文件的集合,是 .NET 框架应用程序的主要构造块<sup>[4]</sup>。程序集构成了基于 .NET 的应用程序的部署、版本控制、重用、激活范围和安全权限的单元。程序集是 .NET 中基本的封装单元,其汇集了多个物理文件到一个单一的逻辑单元中。一个程序集可以是一个类库(DLL)或者一个独立的应用程序(EXE),可以包含多个实体模块(可以是一个实体的 DLL 文件),每个模块可包含多个构件。如图 2 程序集 A 包含一个单一的模块,而程序集 B 包含两个。

程序集中的元数据是一个全面、标准、强制、完全

的、描述程序集内所包含的内容的方式。元数据描述在程序集中有何种可用的类型(如类、接口、枚举、结构等),以及包含它们的命名空间、每个类型的名称、可见性、它的基类、方法、支持的接口、每个方法的参数等。面向 .NET Framework 的编译器向所有的模块和程序集嵌入元数据,使 .NET 构件成为自描述构件,使构件交互更加简单,这种方式有助于构件的无缝集成。

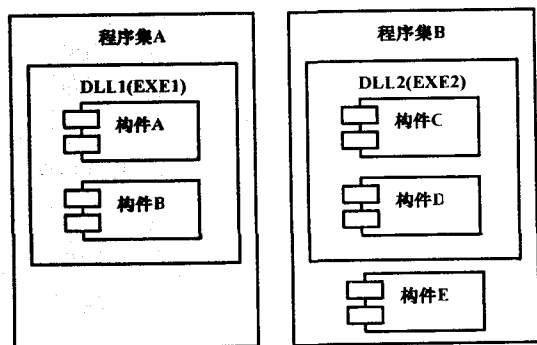


图2 程序集当作逻辑包装单元

### 2.3 公共语言运行时(Common Language Runtime, CLR)

.NET CLR 提供一个公共上下文来执行所有的 .NET 构件,而不考虑具体的编写语言。CLR 管理代码运行时的方方面面,包括提供内存管理、安全的运行环境,以及访问底层操作系统服务。CLR 提供了深层的语言综合,能够使用不同语言开发的构件在运行时高度交互。其结合了多种不同平台和不同操作系统的技术和功能,具有较强的跨平台性。

### 2.4 .NET 的远程处理(.NET Remoting)

.NET Remoting 提供了一种允许对象通过应用程序域与另一对象进行交互的框架。它使得处于不同应用域、不同过程和不同机器上的对象可以实现无缝通信。.NET Remoting 提供的编程模型和运行时支持功能强大且易于使用,能够实现透明的交互操作。.NET Remoting 的体系结构如图3所示。

在图3中.NET Remoting 中通过格式化通道来实现两个应用程序域之间的对象通信。首先,客户端通

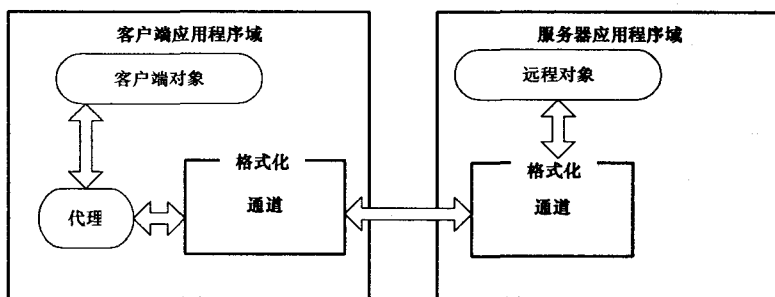


图3 .NET Remoting 体系结构

过格式化通道访问服务器端对象,以获得服务器端对象的代理。服务器端也即通常所说的远程对象,使用时是通过跨应用程序域边界传递对象引用获得该远程对象的代理。对于客户程序来说,代理提供了与远程对象完全一样的方法和属性。不过,要实现指定方法的代理,通常需要将调用传给通道对象。通道对象代表了到远程应用程序的连接<sup>[5]</sup>。

### 3 基于.NET平台的软件构件化开发过程

.NET Remoting 是在 DCOM 等基础上发展起来的一种技术,其屏蔽了底层的细节,它能实现跨平台、跨语言、穿透防火墙等特点,使得在 .NET 平台上开发分布式构件的过程得到了较大的简化。其重点已主要是放在程序集接口的规范、定义、实现和调用上。这里的接口可以有静态成员、嵌套类型、抽象、虚拟成员、属性和事件。基本的开发过程如图4所示。

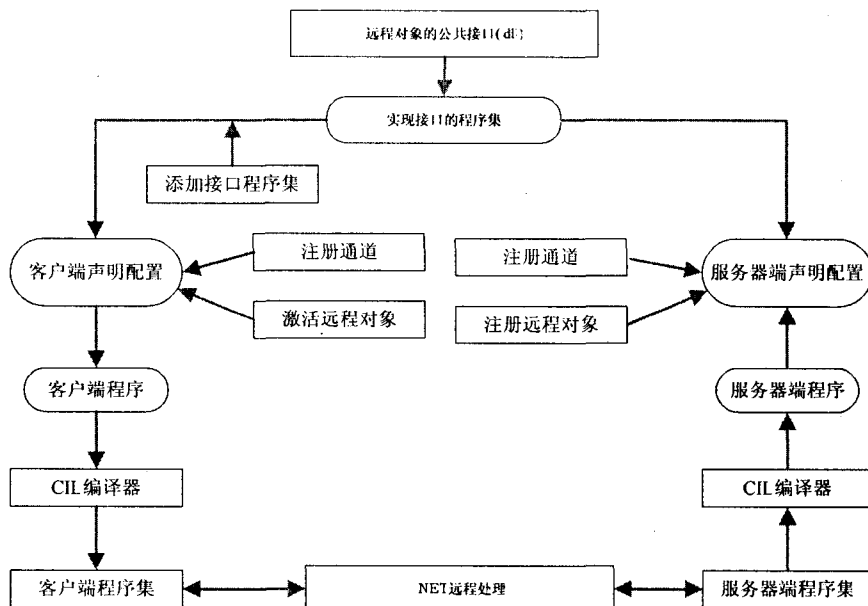


图4 .NET 平台的分布式软件构件开发过程

在图4中,首先定义一个远程对象的公共接口,经过编译,生成了一个后缀名为 DLL 的程序集,这个接口程序集可以同时被客户端和服务端引用。由于

.NET 平台的 CLR 可以运行不同语言开发出来的代码,因而各客户端和服务端只要遵循公共接口规范,就可以用任何编程语言实现客户端程序和服务器端程序。

#### 4 应用实例

政府行业的信息化管理是属于典型的分布式组织结构体系。政府的下属部门往往都要定期地将相关的统计信息传送给上级部门,以便上级部门备案和查询。然而,长期以来,由于多数业务系统都是针对某个具体的基层单位需求开发的,产品总体也一直缺乏系统级的综合规划与设计。随着 Internet 的普及和信息化提高,对于跨区域、跨系统的资源整合需求也使得情况日益恶化,由于缺乏统一的数据结构标准和业务规范,往往必须采取点对点的资源整合方式,针对每个可能的需求进行接口的定制开发,其工作量大、工作效率也是相当低的,且一旦业务流程调整,都会引起系统整体的连锁反应。这些问题,最终导致了政府各级部门办公效率不高、信息共享低、系统运行效率低、可扩充性差等问题。结合软件构件开发和 .NET 平台在开发分布式构件中的优势,可以有效地解决这些问题。而软件构件,最重要的还是在接口实现上。作为实例,现只给出某市人民法院的执行案件中组合查询构件的一个简单接口实现,来说明 .NET 平台的分布式软件构件开发方法。图 5 给出了客户端与服务端接口的实现。

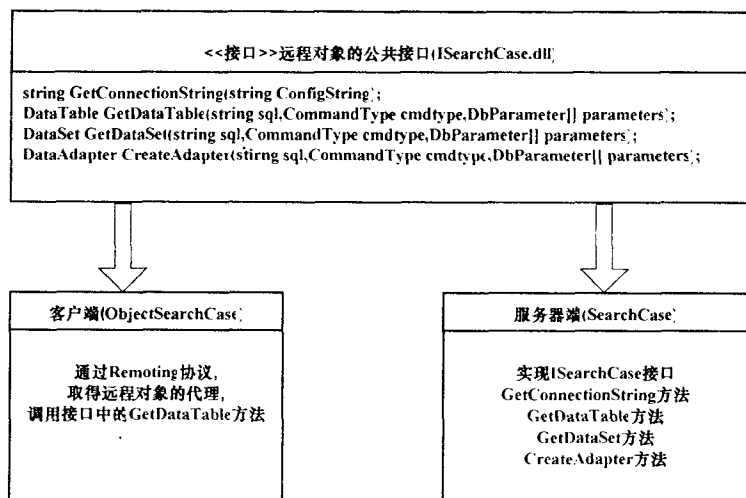


图 5 客户端和服务端接口的实现

图 5 中,首先定义一个远程对象的公共接口类,客户端的对象激活和服务端端的接口实现,都遵循着定义好的 ISearchCase 的接口规范。当客户端创建远程对象的实例 ObjSearchCase 调用 GetDataTable 方法时,.NET 框架在客户端的应用程序域里创建了一个

代理,通过注册一个通道与远程的服务器端连接,当服务器端接到从客户端发来的通道对象信息时,服务器端通过格式化通道,把对象的 GetDataTable 返回给代理,再通过代理解析为客户端对象。具体的开发过程如下:

1) 定义远程对象的公共接口:利用统一建模语言 UML(Unified Modeling Language)对系统进行分析建模,得到对象的构件图(Component Diagram)。在 UML 的开发环境 Rose 中将设计好的构件图生成类库文件,再把类库文件编译成 .DLL 文件。

2) 接口的实现:新建一个名为 RemoteWrapper 的类库项目,添加对接口程序集 ISearchCase.dll 的引用,将类文件的文件名改为 SearchCase.cs,在该文件中定义一个 SearchCase 类,用于实现接口 ISearchCase,同时该类须继承自 MarshalByRefObject。

3) 声明配置文件:下面是一个服务器端通过 SingleCall 模式激活远程对象的配置文件(命名为 DbRemoteServer.exe.config):

```

<configuration>
<!--实现数据库连接的相关配置信息-->
<system.runtime.remoting>
  <application name="RemoteHostService">
    <service>
      <wellknown type="RemoteWrapper, SearchCase, SearchCase"
        objectUri="SearchCase" mode="SingleCall"/>
    </service>

    <channels>
      <channel ref="tcp" port="1234"/>
    </channels>
  </application>
</system.runtime.remoting>
</configuration>
  
```

如果是客户端激活模式,则把 well-known 改为 activated,远程类改为接口类,同时删除 mode 属性。然后,只要需要使用下面一行代码就可以发布远程对象:

```
RemotingConfiguration.Configure("DbRemoteServer.exe.config")
```

同样,客户端也可以使用声明配置文件来获得对远程对象的引用,在使用上述一行代码后,只需要简单使用 new 运算符(还可以采用其他方法)即可以像操作本地对象一样来操作远程对象。

4) 编写实现服务器端的功能代码,注册通道和远

(下转第 65 页)

表 1 DV-Hop 算法和 ROCRSSI 算法性能评价表

性能评价指标	DV-Hop 算法	ROCRSSI 算法
定位方式	分布式	分布式
测距方式	非测距	非测距
计算开销	$(18K - 17 + 2^3/3)F$	$2C(C + 4)F + \Delta$
通信开销	$2AN$	$A(1 + C)$
定位精度	适中(约 33%)	依赖锚节点密度和环境
容错性	不受测距误差影响,较高	较差
网络成本	较低	一般
覆盖速度	与网络规模有关	较慢
覆盖率	较高	与锚节点密度有关

注:  $N$  为网络节点数,  $A$  为网络锚节点数,  $F$  为单个 flop 的能量消耗,  $C$  为网络平均连通度,  $K$  为参与一次多边测量定位的锚节点个数。

## 4 结束语

DV-Hop 算法利用节点间大量的冗余信息来定位,是以通信代价换取的锚节点数量减少,要进一步优化锚节点的部署来提高对网络拓扑结构变化的适应能力和获得更高的覆盖率。该算法仅适合应用在各向同性网络中。在各向异性网络中,可以采用与 Exposure 模型<sup>[5]</sup>、分布式自散步算法<sup>[6]</sup>、基于虚拟力的算法<sup>[7]</sup>等相结合的方法来调整网络拓扑结构,提高网络覆盖率。ROCRSSI 定位算法在锚节点部署、定位精度和通信开销等方面仍有待改进的地方,基于 RSSI 与 DV-Hop 算法相结合来对节点实施定位也是一种较好的定位思路。

通过对上述 DV-Hop 算法和 ROCRSSI 算法两种非测距定位算法性能的评价,可以看到目前并没有一种算法是最优的,每个算法都有自己的应用条件和适用

范围,因此在实际应用中还应采取“均衡策略”选取合适的满足特定需求的定位方案。

## 参考文献:

- [1] 孙利民,李建中,陈渝,等. 无线传感器网络[M]. 北京:清华大学出版社,2005.
- [2] Bulusu N, Heidemann J, Estrin D. GPS-less low-cost outdoor localization for very small devices[J]. IEEE Personal Communications, 2000(10): 28-34.
- [3] Niculescu, Nath B. Ad-hoc positioning systems[C]//Proc. of the 2001 IEEE Global Telecommunications Conf. San Antonio: IEEE Communications Society, 2001: 2926-2931.
- [4] Liu Chong, Wu Kui, He Tian. Sensor Localization with Ring Overlapping Based on Comparison of Received Signal Strength Indicator[C]// Proceedings of Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference. Fort Lauderdale, Florida, USA: IEEE Press, 2004: 516-518.
- [5] Seapahn M, Farinaz K, Qu Gang, et al. Exposure in wireless ad-hoc sensor networks[C]// Proceedings of the 7th annual international conference on mobile computing and networking. Rome, Italy: ACM Press, 2001.
- [6] Heo N, Varshney P K. A distributed self spreading algorithm for mobile wireless sensor network[J]. Wireless Communications and Networking, 2003(3): 1597-1602.
- [7] Iyengar K C S S, Qi Hairong, Cho Eungchun. Grid coverage for Surveillance and target location in distributed sensor networks[J]. IEEE Transactions on Computers, 2002, 51(12): 1448-1453.

(上接第 61 页)

程对象,编译,链接,产生服务器端应用程序。

5) 在编写客户端应用程序时,添加对公共接口的引用,设计相关的界面和编写调用服务的代码,编译,链接,产生客户端应用程序。

.NET 的基本构件单元是以程序集(Assembly)的形式存在,因而通过上述的操作所得到的构件,其在部署、版本的控制和更新以及重用上,都提供了极大的方便。客户端通过获取服务器端的对象,来获得服务端的方法,这样保证了客户端与服务器端的相关对象的松散耦合,同时也优化了通信性能。

## 5 结束语

构件技术是软件复用的核心,随着 Internet 网络的广泛应用,对于分布式构件的要求也越来越高。而.NET 技术集合了过去相关开发平台的优点,其使得开发多语言、跨平台、跨区域、可复用的分布式构件成为

可能。文中所提出的基于 .NET 平台的分布式软件开发方法,能够很好地适应今后系统的演化,在开发同一领域的应用软件时,会大大缩短开发的生命周期,降低开发上的费用,减轻维护工作,从而很好地满足了系统的开发要求,对于用构件化开发方法开发分布式系统,具有一定的指导意义。

## 参考文献:

- [1] 李延春,晏敏. 软件构件技术的现状和未来[J]. 计算机工程与应用, 2003, 39(31): 86-96.
- [2] 王志坚,费玉奎,姜渊清. 软件构件技术及其应用[M]. 北京:科学出版社,2005.
- [3] 刘宪凯,张维石,罗武军. 基于 CORBA 的软件构件开发方法研究[J]. 计算机工程与应用, 2001, 37(19): 151-153.
- [4] Lowy J. Programming .NET Components[M]. Gravenstein Highway North Sebastopol: O'Reilly Media, 2005.
- [5] Barnaby T. .NET 分布式编程—C# 篇[M]. 黎媛,王少锋译. 北京:清华大学出版社,2004.