维普资讯 http://www.cqvip.com

计算机科学1999Vol. 26№. 9

一种可重用构件模型——类属构件

A Model of Reusable Component——Generic Component

李景峰 刘西洋 陈 平

(西安电子科技大学软件工程研究所 西安710071)

Abstract Component-based software reuse is an important reuse approach. An important means to improve a component's reusability is to make it more abstract. In order to make a component more reusable in this way, the paper first presented a new model of reusable component——generic component, then discussed its concepts, description and parameterization mechanism. Finally, a runtime instance of generic component——metadata-based component is given. At present the idea of generic component has already been applied successfully in the development of a cross-platform component collaboration framework.

Keywords Component-based reuse, Generic component, Metadata-driven component, Unified modeling language

软件重用是在开发一个新的应用系统时,重复使用以前在别的应用系统开发活动中所建立和使用过的软件资源[1]。这些软件资源不仅包括源代码,还包括软件开发方法、需求规格说明、设计结构、开发工具与支撑环境、测试分析数据和维护信息等。软件重用技术有助于提高软件开发的生产率,提高软件系统的可靠性,降低软件开发的成本,因而它已成为软件工程的重要研究课题。

作为软件重用的一种重要方式,可重用构件(以下简称构件)得到了广泛的应用。它是将应用系统中相对稳定的成分抽取出来,形成可以重复使用的软件单元——构件,以便于以后系统开发时使用。软件开发者可以通过组装已有的构件来开发新的应用系统。然而在实际系统开发时,我们发现了这样的现象:一些构件基本功能相同,只是部分功能存在差异,但它们却是不同的构件,这导致了构件的可重用程度不高。为了增加构件的可重用性,我们需要提高构件抽象层次,使其适应性更广。基于此,我们提出了类属构件的可重用构件模型。

1 类属构件的概念

1.1 类属构件的定义

在定义类属构件之前,我们先明确构件和构件域的概念。

构件是封装了它的设计和实现而仅向外部提供接口的相对独立的可重用软件单元。所谓接口是一套用于说明构件提供服务的操作集合,它着重于一个给定服务的行为而不是服务的实现。服务的实现由构件内部的实体具体完成。接口给出了一组操作的名字,这些操作协调工作完成一个系统或子系统的逻辑行为。

定义1 构件C=(CN,I,S):

(1)CN 表示构件名,构件采取统一的命名规则,并且能够体现出领域特色,这样可以方便地存储和管理构件;

(2)I 表示接口,即操作集,I={Operation|Operation = (OperationName, Description)} OperationName 是操作名,Description 是操作的描述;

(3)S 是对构件功能的详细描述,它采用非形式化、自然语言的形式对构件的特性作出更为详细、准确的描述,主要记录构件版本说明、构件开发环境、构件运行环境、构件的使用环境、构件功能的自然语言描述、构件的可重用信息。这些形成了构件文档。

构件域是一组接口相同、功能相近的构件集合。所以一个构件域有一个标准接口,这个域中的每个构件都必须满足这个接口。但是同一构件域中的各构件可以有不同的操作对象或实现。

定义2 构件域V=(I,S):

维普资讯 http://www.cqvip.com

(1)I 表示构件域的接口,I 的含义与构件定义中的相同;

(2)S 是对构件域中的各构件功能共性的描述。

类属构件是对构件域的抽象,它抽取出构件域中各构件的共性部分,把不同的部分用参数的形式表示。类属构件可以看成是构件的模板,是一种参数化的构件机制。类属构件只有经过实例化后,才能生成相应的具体构件。

定义3 类属构件 GC=(GN,I,P,S):

(1)GN 是类属构件名,采取统一的命名规则;

(2)I 表示类属构件的接口,I 的含义与构件定义中的相同;

(3)P 是类属构件的参数列表,$P=(p_1,p_2,\cdots,p_n)$,$n\geq 1$,其中 $p_i(1\leq i\leq n)$ 属于一个确定有限的值集;

(4)S 是对类属构件功能的描述。

1.2 类属构件的类属性

类属构件表示的是一个构件集合的共性。它描述的不是一个构件,而是一类构件或一个构件域。类属构件从两个方面表达其类属性:(1)类属构件的参数:把同一构件域中各构件的共性抽取出形成类属构件,而把各构件的差异用类属构件参数的形式表示;(2)类属构件的语义描述:在类属构件的语义描述中,类属构件参数与相应构件的对应关系必然要出现在该语义描述中。

1.3 类属构件的实例化

类属构件的实例化是类属构件各参数值集笛卡尔积的子集 R 到构件域 V 的一个映射,即 $G:R\rightarrow V$,$R\in R_1\times R_2\times\cdots\times R_n$,V 是一个构件域。其实例化的过程是类属构件的形参被实参代替的过程。

1.4 类属构件的作用

引入类属构件有以下作用:(1)进一步提高了应用系统中实体的抽象层次。类属构件是对构件域的抽象,它描述了问题的一个空间,其本质是提高了构件的抽象层次。(2)使构件机制具有很大的柔性。因为类属构件表示了构件域中各构件的共性部分,而它们的差异通过类属构件参数的形式表示,这增加了构件机制的适应性。(3)对构件库的构造提供了强有力的支持。

2 类属构件的描述

对类属构件的描述,我们给出两种方法:语法描述和扩充的 UML 描述。语法描述是基于构件形式化开发的基础,也便于构件库中构件的管理。用扩充的 UML 描述类属构件便于我们在基于构件的非形式化开发中,表示系统结构和构件的特性。

2.1 类属构件的语法描述

下面的语法描述中,符号"*"是闭包的含义,"[ ]"中的内容可以为空,类属构件的语法如下:

⟨GComp_def⟩ ::= GComp ⟨GComp_spec⟩ [⟨GComp_body⟩]END⟨GComp⟩

一个类属构件由类属构件说明和类属构件体两部分组成。其中类属构件说明部分的语法为:

⟨GComp_spec⟩::=
Spec ⟨GComp_name⟩类属构件名
⟨GComp_para⟩类属构件参数表
⟨GComp_semantic⟩类属构件语义描述
END⟨GComp_spec⟩

这里类属构件说明由类属构件名、类属构件参数表和类属构件语义描述三部分组成,类属构件名的语法:

⟨GComp_name⟩::=f(⟨Identifier⟩)

⟨Identifier⟩::=Letter{Letter|Digital|'_'}*

类属构件名是自变量为标识符标识的一个函数,该函数把标识符按统一的命名规则命名,并且能够体现领域的特色。标识符是以字母打头的字母数字或下划线串。

类属构件参数表的语法:

⟨GComp_para⟩::=Para(⟨para⟩{,⟨para⟩}*)

⟨para⟩::=⟨Identifier⟩

类属构件语义描述的语法:

⟨GComp_semantic⟩::=
⟨GComp_function⟩
⟨GComp_paratofunc⟩
⟨GComp_env⟩
⟨Doc_name⟩

类属构件语义描述由类属构件功能描述、参数与功能的对应关系、类属构件环境和类属构件的相关文档名四部分组成。

类属构件环境的语法:

⟨GComp_env⟩::=⟨GComp_version⟩⟨Develop_env⟩

类属构件环境包括版本说明、构件开发环境。

类属构件体的语法说明:

⟨GComp_body⟩::=Body ⟨GComp_obj⟩ END ⟨GComp_body⟩

类属构件体是类属构件操作的具体实现。

2.2 用扩充的 UML 描述类属构件

UML(统一建模语言)是由 Rational 公司的著名 OO 专家 Gray Booch、Jim Rumbaugh 和 Ivar Ja-

维普资讯 http://www.cqvip.com

cobson 三人主导，由众多 OO 专家与计算机厂商共同开发的第三代面向对象建模语言[2,3]。它采纳和扩充了 Booch、OMT 和 OOSE 方法，并包含了其他学者和软件厂商的建议，现已被 OMG 组织认定为面向对象建模语言的标准。UML 是一种适应于广谱建模语言，支持各种典型的 OO CASE 工具。

在 UML 中没有类属构件的概念，所以无法表示它。这里我们对 UML 进行扩充，引入类属构件的表示法，类属构件可以用图1表示，它表示带有参数表 $P(p_1,p_2)$ 的类属构件 A，其中 p_1,p_2 为形式参数。

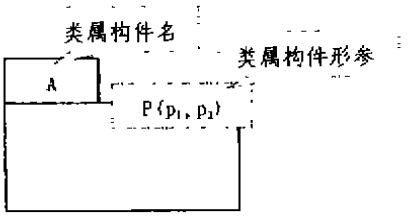

图1 类属构件

类属构件的唯一的行为就是实例化。只有类属构件中的形参被实参替换实例化成构件后，才可以被使用。图2表示了类属构件的实例化。

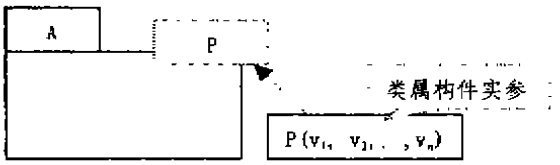

图2 类属构件的实例化

类属构件经过实例化后已经成为构件，所以可以用在任何能引用构件的地方。

3 类属构件的参数

3.1 类属构件参数简述

类属构件的参数反映了类属构件的柔性，它是构件域中各构件差异的表现形式。通过改变参数的值可以控制类属构件的行为，反映出这种差异。

类属构件的参数是一个参数列表，即 $P=(p_1,p_2,\cdots,p_n)$，其中 $p_i(1\leqslant i\leqslant n)$ 表示形式参数。对于每个 p_i，它都有一个确定的、有限的值集，即 $p_i\in R_i$，R_i 是一个有限集。类属构件参数的取值是 $R_1\times R_2\times\cdots\times R_n$ 的一个子集 R。

类属构件参数按时态可分为编译时的参数和运行时的参数。编译时的参数主要是数据类型，它可以是基本数据类型和抽象数据类型；运行时的参数主要是类型的值。编译时的参数在编译时决定类属构件的类型，而运行时的参数在构件实例化时改变其结构和行为。它们与类属构件、构件及构件实例之间的关系如图3。

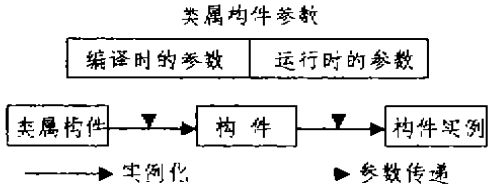

图3

3.2 类属构件编译时的参数

类属构件编译时的参数可以是基本数据类型和抽象数据类型，下面我们就分两种情况讨论：

·基本数据类型作为类属构件编译时的参数。因为基本数据类型本身不具有对自身操作的能力，所以它作为类属构件参数的情况是比较简单的。通过给类属构件的参数赋予不同的基本数据类型，我们可以使类属构件实例化成基本功能相近，但又有差异的不同构件，即同一构件域中的构件。这种情况可以用图4表示。

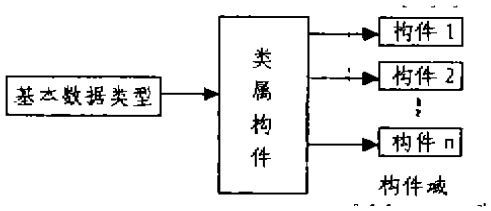

图4 基本数据类型作为类属构件编译时的参数

·抽象数据类型作为类属构件参数。抽象数据类型是一个值集和作用在值集上的操作集。与基本数据类型相比，抽象数据类型增加了对数据的操作部分，这使得它对自身有了一定的加工处理能力。

当抽象数据类型作为类属构件编译时的参数时，因为抽象数据类型本身既具有数据又具有对自己数据的加工能力，所以它可以扩充构件的数据和增加构件的功能，其实质是改变构件的结构和行为，提高了构件的适应性，根据问题空间的变化，这种机制可以在一定范围内扩充构件功能。该机制可以用图5来说明。

3.3 类属构件运行时的参数

类属构件在编译时读入类型参数（编译时的参数）、确定构件的类型，即在程序编译时构件已经含

维普资讯 http://www.cqvip.com

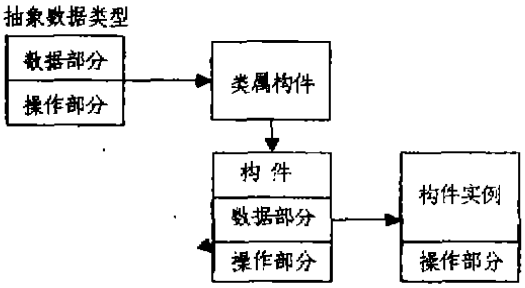

图5 抽象数据类型作为类属构件编译时的参数

有确定的类型，而类型的值则在程序运行时被动态读取。类属构件运行时的参数值就是类型的值，它们被解释后，可以用来改变构件的结构和行为。因为类属构件运行时的参数在程序中不是被加工的对象，而是可以通过值的改变而改变构件结构和行为的数据，所以我们称其为元数据。在下一节中，我们将详细讨论元数据的概念。

4 运行实例——元数据驱动的构件

4.1 元数据的概念

元数据是类属构件运行时参数的取值，是能改变构件结构和行为的数据。元数据独立于构件代码的编译，它是构件行为差异的表现形式。元数据在构件实例化时被读入，通过解释动态地改变构件的结构，控制构件的行为。

根据元数据的定义，我们可以看出元数据有以下特性：

·高度的抽象性：元数据的抽象层次比普通数据高，普通数据是被加工的对象，而元数据是可以改变构件结构和行为的数据，即元数据有了一定的控制能力。

·取值的稳定性：构件的结构和行为是确定有限的，一旦确定了元数据的描述之后，它的值集一般是稳定的。

·与编译的无关性：元数据是构件行为的抽象表示，一般以静态的方式存储，它与程序的编译是无关的。

·对类属构件很强的依赖性：元数据是类属构件的运行时参数的取值，它必须与类属构件相联系，离开了类属构件元数据就失去了其意义。

根据元数据值的相对稳定性这样特点，在应用系统中它通常以静态形式存储，如存储在静态文本、数据字典或数据库中。在构件实例化时，从其中读入元数据，生成具体的构件实例。系统使用数据库存储元数据，并提供工具管理它，是元数据存储发展的趋势。

4.2 元数据驱动的构件设计

元数据驱动的构件是类属构件的运行实例，它可以用图6表示：

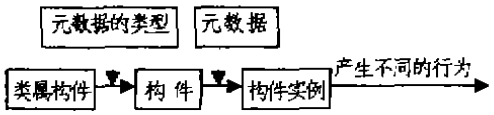

图6 元数据驱动的构件示意图

元数据驱动构件的产生过程是首先把元数据的类型作为参数传递给类属构件形成构件，然后在构件实例化成构件实例时，读入元数据，产生不同的结构和行为。

5 类属构件的实现

5.1 OO程序设计语言实现类属构件的方法

对于类似C++、Ada等强类型化编译型OO程序设计语言，其语言构造被严格地约束在3阶层次结构中，即2阶的模板、1阶的类（类和类型虽然是截然不同的两个概念，但在C++和Ada语言中，类同时也被当作类型来看待。）以及0阶的类实例，并且从模板到类的实例化操作完全由编译器在编译时实施，从类到类实例的实例化操作则在运行时实施。在这种条件下，类属构件可以映射为语言中的模板、相应元数据模式以及以模板方法代码为表现形式的元数据操纵协议。图3中的编译时的参数映射为模板参数，运行时的参数则映射为元数据。

对于类似Smalltalk等弱类型化动态OO程序设计语言，其语言构造建立在依次由元-元-元类（Meta-Meta-Meta Class）、元-元类（Meta-Meta Class）、元类（Meta Class）、类（Class）和实例（Instance）构成的递归元体系结构之上[6]，理论上可以定义任意高阶的语言构造，并且所有由n+1阶到n阶的实例化操作都映射为对n+1阶语言构造对象运行时的消息传递序列。类属构件在这里映射为施加强类型约束之后的元类、相应元数据模式以及以元类方法代码为表现形式的元数据操纵协议，但是编译时与运行时的时态界限已经模糊，所有的实例化操作均可统一为由n+1阶到n阶的实例化操作。

5.2 应用实例

在设计和实现面向C³I领域的DSSA过程中，

（下转第80页）

维普资讯 http://www.cqvip.com

必由中转服务器中转，中转服务器也是一个 SMTP 邮件服务器。多媒体电子邮件程序模块功能简述如下：

(1)**系统设置** 用户设置 SMTP 服务器的域名和 IP 地址、用户的电子邮箱地址、用户档案信息以及多媒体数据的编码方法等系统信息。

(2)**邮件发送** 在用户发送多媒体邮件之前，必须制作一个多媒体邮件，这里包括视频及语音信息的采集与压缩，生成一个符合 RFC822规范的邮件格式，并将其中各种信息对象存入多媒体数据库，用基64编码对该多媒体数据进行处理，从而形成一个符合 SMTP 协议的多媒体电子邮件，最后将其发送到 SMTP 邮件服务器。

(3)**接收邮件** 用户从 POP3服务器获取多媒体电子邮件，由于该多媒体邮件不能直接显示，因此必须进行处理：首先从邮件中析取各种多媒体对象，用基64编码对多媒体数据库进行处理，并将这些数据存入多媒体数据库，再根据这些多媒体数据为用户播放和显示邮件中的多媒体对象。

(4)**邮箱处理** 用户原始邮箱有3个：接收邮箱、发送邮箱和垃圾邮箱。它们分别用来存贮接收、已发送和删除的多媒体邮件副本，用户也可以根据需要建立新的邮箱或删除自己建立的邮箱。

(5)**辅助功能** 辅助功能包括用户地址薄、用户签名(包括多媒体信息签名)、邮件转存、分类以及多媒体格式转换等等。

结束语 在 Internet 上，电子邮件已成为人们不可缺少的通信工具，但人们所殷切盼望的多媒体电子邮件目前还处于研究开发阶段。本文所设计的多媒体电子邮件在许多关键技术方面都作了有益的探讨，所研制的多媒体客户软件已在中文 WIN98环境下实现。用 VB、VC 和汇编语言编程，其设计的大部分功能已经实用化，并开始试运行。但全部功能的整体离人们对多媒体电子邮件的期望还有一段距离，我们将在这一领域作进一步的努力，特别是在多媒体数据的存贮、压缩和同步播放方面，作进一步优化，使多媒体邮件技术完全实用化。

在软件的编程方面，区庆彬和叶菲同学作了一定的工作，在此表示感谢。

参 考 文 献

1 Freed N et al. Multipurpose Internet Mail Extensions Part Four: Registration Procedures. RFC 2048, Nov. 1996

2 Postel J. Simply Mail Transport Protocol (FRC821). 1982.8

3 Myers J. Post Office Protocol-Version 3 (rfc1939). 1996.3

4 巩志国，等. 多媒体数据库系统的体系结构. 计算机科学，1998，25(6)

5 郑庆华，等. 多媒体数据库的主要问题、研究内容与实现方法. 计算机工程与应用，1998.1

6 石岩，等. 计费邮件服务器研究与实现. 计算机工程与应用，1998.12

7 陈坚，陈涛 Windows 95多媒体应用程序设计技术 西安电子科技大学出版社. 1997.3

8 ftp://ftp.isi.edu/internet-drafts

(上接第86页)

类属构件被应用于 DSSA 构件模型的构造中，其中支持构件协作交互的分布构件总线已经得到应用的验证，元数据成功地被用于参数化构件的分布拓扑结构、构件组服务的配置和外部请求的指派策略，从而使得由上述 DSSA 开发的构件应用能够在运行时(不需中止运行并重新编译、链接和装入)动态配置，这种能力正是关键应用系统的开发所需要的。

参 考 文 献

1 Mili H, Mili F, Mili A. Reusing Software: Issues and Research Directions. IEEE Trans. Software Engineering, 1995, 21(6): 528～561

2 Unified Modeling Language Summary Version 1.1, Rational Software Corporation, Sept 1 1997

3 Unified Modeling Language Notation Guide Version 1.1, Rational Software Corporation, Sept 1 1997

4 D'Alessandro M. et al. The generic reusable component-An Approach to Reuse Hierarchical OO Designs In. Proc. Advances in Software Reuse. Los Alamitos, California: IEEE Computer Society Press, 1993. 39～46

5 李景峰. 基于可重用构件的软件开发模型：[硕士学位论文] 西安：西安电子科技大学，1999

6 Goldenberg A, et al. Smalltalk-80: The language and its implementation. Addison-Wesley, 1983