第22卷 第2期
2004年6月

应用科学学报
JOURNAL OF APPLIED SCIENCES

Vol. 22, No. 2
June 2004

文章编号：0255 - 8297(2004)02- 0211 - 06

基于虚构件的软件开发研究

刘 辉，缪淮扣，刘 静

（上海大学 计算机工程与科学学院，上海 200072）

摘 要：对构件的软件工业化生产中的两个主要问题：构件模型和构件组装进行讨论．在总结了最具代表性的国内外构件模型的成功与不足之后，文中提出了一个改进的构件模型．在此基础上，为方便构件的检索和组装而提出了虚构件的概念，最后提出引入虚构件概念之后的软件开发模式．

关键词：构件模型；软件体系结构；构建组装；虚构件；基于虚构件的软件开发

中图分类号：TP311.52 **文献标识码**：A

A Development Study of Software Based on Virtual Components

LIU Hui, MIAO Huai-kou, LIU Jing

(*School of Computer Engineering and Science, Shanghai University, Shanghai* 200072, *China*)

Abstract: Industrialized development of software is one of the most interesting but difficult research fields in software engineering. This article discusses the most important problems: component models and component composition. Investigating representative component models, this paper proposes a new component model. Based on this model, it brings forward a new concept: the virtual component. A development model with virtual components is explained in the end.

Key words: component model; software architecture; component composition; virtual component; VCBSD

因为软件复制的“零成本”，软件生产者开始考虑软件复用[1,2]，也就是利用别人生产的比较小的软件体搭建自己的系统或者是搭建一个比较大的可复用软件体，这种软件体就称为构件（组件）．构件已经成为软件复用的基本模块，现在出现了大量的商业化构件（commercial off the shelf COTS）以及企业内部使用的可复用构件．可复用构件模型也就成为了软件复用和软件工业化生产的关键技术，因为构件模型将决定构件对复用的支持力度，也决定了构件生产的相关工序．现在最具代表性的构件模型有OMG 的 CORBA，Microsoft 的 COM/COM＋和.NET构件模型 SUN 的 EJB．

但是构件的开发并不是复用的全部，它只是面向复用的开发（development for reuse），另一个重要的组成部分是基于复用的开发（development with reuse）．它主要是解决怎样利用现有的可复用构件组装成一个可执行的系统或者是更高层可复用构件，这是当前软件工程的一个热点，也是一个难点．

软件工业化生产的核心部分就是软件组装的自动化．我们正是从软件生产的这两个关键点入手，分析了现有主要模型的成功与不足之处而提出了比较完善的与现有模型相兼容的构件模型．然后在此模型的基础上研究基于构件的组装，为实现构件组装的自动化（或半自动化）而提出了虚构件的概念．

收稿日期：2003-02-26； **修订日期**：2003-09-04

基金项目：国家自然科学基金（60173030）；上海市教委科学与技术发展基金（02AK08）资助项目

作者简介：刘辉（1978-），男，福建长汀人，硕士生，缪淮扣（1953-），男，江苏淮阴人，教授，博导．

1 构件模型

要使用构件就得描述好构件.而描述一个构件的前提就是知道应该怎样描述它,也就是构件模型问题.随着构件的逐渐流行,许多组织开始了构件模型的标准化.OMG 为解决分布式构件的集成与交互而开发出 CORBA 模型.而 Microsoft 则在 OLE 的基础上提出了 COM、COM+的构件模型,从而实现了不同编程语言编制构件的二进制级别的兼容性.Sun 公司为其新式语言 Java 制定了 JavaBeans,并和 OMG 组织合作试图使得 Java Beans 可以和 CORBA 相互调用.

1.1 构件依赖性

现在许多构件模型都在接口中指明其对外要求的功能集合[3~5].其目的就是在接口部分就可以知道构件之间的依赖性,在某个构件所提供的接口发生变化的时候就可以知道哪些构件会受到影响.而且在构件组装的时候只要知道了接口的信息就可以了.而工业界使用的构件模型(比如微软的COM+)是把构件之间的依赖性隐藏在构件的实现体中,在接口中是不可见的.尽管这些模型指出了构件的依赖性,但它们也还有不完善的地方.

最重要的就是复合构件的表示方式(复合构件可以看成是相互依赖的构件的组合体).这些模型通常利用 instance 、connection 和 mapping 三个字段把子构件实例化[5].instance 字段是指出子构件的实例;connection 指出子构件之间的连接(调用)关系,也就是在这些构件的对外要求的接口与对外提供的接口之间进行连接;mapping 字段把复合构件的对外接口映射到子构件的对外接口.这种方法虽然简化了复合构件的组装,但是很明显限制了构件的复合.比如说,某个子构件只是出现在复合构件的某个过程中,而且是作为局部变量出现.那么这种在复合构件的总体描述中实例化的方法就没法描述出这种复合关系了.因为你不可能知道在这个函数中子构件的实例一共会有几个,也不知道它们什么时候被撤销.如果一定要在总体说明中实例化的话,只能是实例化最大的可能实例数,并且永远不被销毁,直到复合构件本身被销毁为止.有时甚至就是完全不可能的,比如在复合构件中有如下函数(方法):

```
void supercomponent : : function(int i)
{  subcomponent arr[]=new subcomponent
[i]  }
```

这里父构件 supercomponent 引用了子构件 subcomponent,但是 subcomponent 到底要实例化几个,则是由调用 function 的参数 i 决定的.这样它们的构件模型就无法决定怎么进行 subcomponent 的实例化 ,当然也就不能进行连接和映射了.

事实上,根本就没有必要进行子构件的实例化,也没有必要进行连接和映射(当然,这里主要是考虑黑盒复用).为了达到上面提出的目标,我们要做的仅仅是在复合构件中指出它引用了哪些构件.在组装的时候,只要取出它引用的构件,复合体本身可以自动进行子构件的实例化(比如 COM+就可以自动实例化).对于它到底是怎么实例化的,我们没有必要知道,因为它对于构件的组装没有任何意义.

1.2 形式化规约描述

现有的构件模型往往没有形式化的规约描述.非形式化的描述可以让你知道构件的接口是怎样的,但是它不能描述构件的功能是什么(虽然可以用自然语言来描述它,但自然语言有其模糊性,而且也不利于构件检索与组装的自动化).在 IDL 中描述的接口只有各个函数的签名,它没有规定接口的语义.也就是说,某个构件即使在语法上实现了该接口,也不能保证它在语义上实现了这个接口,这是因为完全不相关的函数也可以使用相同的函数签名.所以,我们必须在构件模型中引入形式化方法.

有些模型提到过形式化描述[3,4],但仅仅是为函数提供前后置谓词.脱离构件和接口单独对函数(方法)进行形式化描述有时难免力不从心.特别是对于有状态构件,它的某个方法的功能可能就是改变构件的状态,此时脱离构件(或接口)的函数描述是无法描述它的确切功能的.所以应该把形式化描述和相应的接口与构件相联系[6,7].在接口和构件一级提供对其整体状态的形式化描述(我们采用了广为流行的形式化语言 Object Z).比如一个计数构件 Counter,它有一个内部的计数变量 Count 和三个方法 Increase、Decease、Get.它们分别表示给计数变量加 1、减 1 和返回计数变量,它的形式化描述如图 1.很明显,不借助于构件(接口)级别的状态变量是不可能得出每个函数(方法描述)的功能的,所以我们认为应该引入接口级别的形式化.

那么接口一级的形式化规格说明会不会和方法一级的规格说明一样,也就是说,如果不借助于上一级(组件之于接口,接口之于方法)能否作出完整

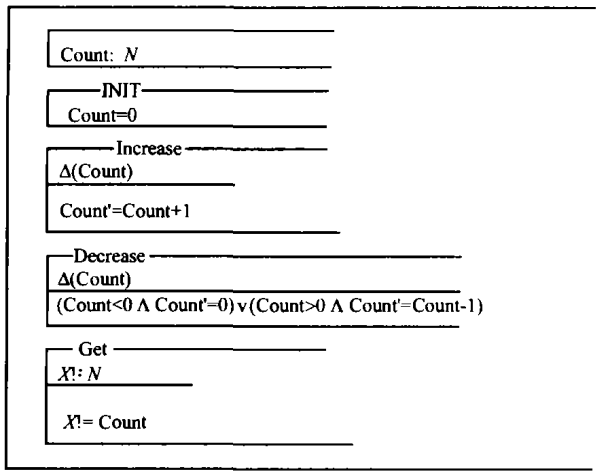

图 1 计数器的形式化描述

Fig. 1 Formal description of a counter

的描述.

先来看接口的作用.接口的引入不外乎就是为了在实现该接口的不同组件可以互相替代,而不会影响程序的行为特性.也就是说,接口的不同实现应该具有完全相同的语义.换句话说,接口的语义应该是独立的,与其实现的载体(构件)无关.所以,接口应该是可以自描述的,可以不借助于组件而写出完整的形式规格说明.

但是,无论是COM还是EJB接口,都没有规定接口语义的独立性.这主要是因为它们使用的接口描述语言IDL没有语义约束功能.我们要求在接口级别上给出形式化的规格说明,也就相应地保证了接口的语义独立性.

既然接口是自描述的、独立于组件的,那么形式化规约的描述最好的办法就是:在接口级别上描述接口的状态,在方法(函数)级别上给出方法的规格说明(可以使用接口的状态),而组件一级则不必给出形式规格说明.

1.3 改进的构件模型

基于上面的研究与探讨,我们提出改进的构件模型如下:

Component : : =〈Provides, References, Version, Cryptographic key〉

Provides : : =〈ProInterface〉|〈ProInterface, Provides〉

ProInterface : : =〈Specification, Functions〉

Functions : : =〈Function〉|〈Function, Functions〉|ε

Function : : =〈signature, Specification〉

References : : =〈RefInterfaces, RefComponents〉

RefInterfaces : : =〈RefInterface〉|〈RefInterface, RefInterfaces〉|ε

RefInterface : : =〈Specification, Functions, Implementation_Component〉

RefComponents : : =〈RefComponent〉|〈RefComponent, RefComponents〉|ε

RefComponent : : =〈Provides, Version, Cryptographic key〉

Implementation_ Component : : =〈Component Name, Version, Cryptographic Key〉

(Pro-表示对外提供的;Ref-表示对外要求的)

Specification:是形式化规约描述,共分两级即接口级和方法级.

2 基于虚构件的软件开发(VCBSD)

为了方便基于构件的开发(CBD),人们进行了许多有意义的尝试.比如构件运行时的概念,它大大地方便了构件的发布和配置.与此相似的还有Agent Component的概念.Martin认为构件只要具备了以下六个特性就可以成为Agent Component:Collaborative、autonomous 、adaptable 、knowledgeable 、persistent、mobile .他还为Agent提出了Agency和Agent platform的概念.事实上这是对COM+ 和COM+ Service的扩展和改进.Microsoft和IBM联合制定了Web Service和SOAP协议的标准.它利用XML和HTTP实现了不同平台、不同编程语言之间的服务调用[8].事实上,你可以把Web Service看成是一个新型的构件,一种无状态的(相当于EJB中的无状态BEAN)运行于Web Server上的独立的可供任何平台任何语言调用的构件.

但是我们知道,从最初的函数库到OLE、CORBAR甚至现在的COM+ 、EJB、CLR(common language runtime)都没有能够促使基于构件的软件工业化生产变成现实.其中的一个主要原因(不考虑非技术因素)就是构件的检索和组装.在编程中复用构件的办法就是一个一个地学习这些构件(比如MFC类库)然后使用的时候就只能凭你的记忆或者通过构件的名字在帮助文档(比如MSDN)中查找该构件.这种方式下的构件复用大大地增加了复用者的

负担.现在很多学者都提出了构件库的概念,也研究出了很多相关的检索技术[9].复用者对想要的构件进行描述,然后利用相应的库检索工具检索出能满足需要的构件.但是,一个大的系统往往是由几十甚至上百个小构件组成的,如果每个构件都要进行描述然后从构件库中检索的话,那烦琐也是可想而知的.

2.1 虚构件概念

我们从 Design Space[10]的概念中得到一些启发.Design Space 利用领域的专门知识来帮助软件开发人员完成从功能规约到软件体系结构的选择.也就是利用某个领域的知识建立一个 Design Space,该领域的软件开发人员只要给定功能和性能等参数就可以确定一个合适的软件体系结构,这就使得软件体系结构的分析和选择变得很容易了.

那么我们是不是也可以建立一个类似于 Design Space 的概念来帮助开发人员自动完成构件的选择和组装呢?于是我们提出了虚构件(virtual component)的概念.

虚构件是介于构架和构件之间的,为帮助开发人员进行构件的查找和组装的一个可执行程序块(如图 2).

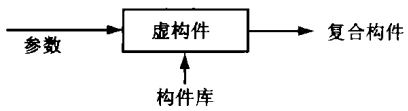

图 2 虚构件示意图

Fig. 2 Sketch map of virtual component

软件开发人员只要对虚构件设置相应的参数(用以指出你的特定需求),并执行虚构件,它就会在现有的构件库中检索出合适的小构件,并把它们组装成一个能满足开发人员特定要求的复合构件.软件开发人员利用它所生成的复合构件来组装更大一级的复合构件或者软件系统,所以从某种意义上说,它实现了构件(系统)组装的自动化.

我们说它是"构件"是因为在分析应用系统的高层结构的时候,你可以把虚构件当成一般的构件使用.说它是"虚"的,是因为它本质上不是真的构件,它的输出才是你所要的构件.虚构件应该专注于某个专门领域或者某个方面,因为只有这样才能抽取出公共的特征及参数.在这一点上,它和领域构架有点类似.它们都是把构件组装的知识用一个构架(或虚构件)的形式来封装,但是它们的区别也是很明显的.首先,领域构架通常是一种图形或文字的描述,它的复用主要是给开发人员一个参考.而虚构件是一个可执行的程序体,你只要输入合适的性能等参数,它就会自动给你查找子构件并把它们组装成一个复合构件.它大大简化了构件的检索和组装.其次,领域构架一般是作为软件系统的整体骨架,而虚构件(确切地说是虚构件生成的复合构件)仅仅是软件系统的一个构件而已.从这一点上说,虚构件和领域构架是共存的且相互协作的.

构件有了生产者和消费者,很明显也应该有销售商.以后虚构件会在构件销售商的服务器上大量使用.构件消费者查找到合适的虚构件之后,设定参数并执行它.虚构件就会自动报出用到的小构件.如果服用者需要使用的话,就购买相应的小构件,而不是用于支撑虚构件的所有小构件.

那么我们为什么不把这些小构件的所有排列组合(有些组合没有意义)都事先组合好,然后把组合好的复合构件都存入库中以备查询呢?是因为这样做有诸多不妥之处.首先是资源空间的巨大浪费,因为每个小构件都要有许许多多相同的拷贝.其次是不利于检索,现在的构件检索一般是基于构件的功能和应用领域的,它无法区分性能等特性.也就是说,一次检索会返回所有的组合好的复合构件,而具体的选择只能留给构件复用者,这无疑加重了复用者的负担,也加大了查询网络的负载.而使用虚构件则只返回虚构件本身,复用者只要设定相关的性能等参数,虚构件就会自动提取出合适的复合构件.这样既避免了在检索的时候使用性能等参数,也过滤了不符合特定需求的构件.

2.2 虚构件模型

为了实现构件的统一性,最好是用一般构件的描述模型来描述虚构件.事实上,虚构件是一个可执行的构件,所以完全可以用构件模型来描述虚构件.但是虚构件的执行需要用到底层的小构件,所以在它的依赖性列表中应该列出可能用到的小构件.

但是,这种模型描述的是虚构件本身,而不是它可以生成的复合构件(也就是复用者真正需要的构件).构件复用者在查找构件的时候,使用的是所需构件的描述,而在查找时候,搜索引擎却用它和虚构件的描述相比较,所以这种搜索是注定要失败的.

为了解决这个问题,就需要在虚构件的描述中加入一个额外的部分:对可能输出的构件的描述.这样在查找的时候,就可以使用这些描述与用于查询

的描述相匹配.

2.3　一个示例

我们来看一个例子.这是一个分布式软件的通信部分.它包含了多个小构件,包括IP、TCP、UDP、SOCKET和HTTP等5个小构件.SOCKET编程灵活高效,但难度较大;HTTP简单易用,但只限于传送文本而无法传送二进制数据.在SOCKET中可以使用TCP/IP或者是UDP/IP的通信模式.TCP是基于流的有连接的传输协议,所以比较可靠,但同时速度受到影响.UDP是无连接的报文传输协议,它快速但不可靠.如果你对通信模块很熟,那么你就可以建立一个通信虚构件(如图3所示),以方便那些对通信的底层协议不熟悉的或者是不愿意花上几个小时来组装一个通信复合构件的开发人员使用.图中的每个输入参数表示相应的通信性能在系统中被优先考虑的优先级.比如说,"速度"参数比"可靠性"参数大就表示通信速度性能的优先级比可靠性要来得高.如果通信的速度最重要,那么很明显应该使用UDP协议.所以只要把输入参数的"速度"设成最大值,通信虚构件就可以生成图4所示的复合构件.相反地,如果输入参数中"易用性"和"可靠性"远大于"速度",那么应该使用HTTP协议,从而生成图5所示的复合构件.当然还可以有其他的输入参数组合.至于哪种输入参数的组合应该输出怎样的复合构件,这是一个领域知识,是在设计这个通信虚构件时就确定了并封装于这个构件中.虚构件的使用者是没有必要知道这种知识的.

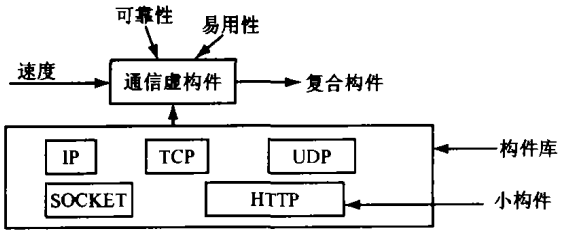

图3　通信虚构件模式

Fig. 3　Mode of communication virtual component

2.4　基于虚构件的软件开发模式

以后构件的查找与组装很大一部分是由构件的销售商(而不是开发人员,或者说消费者)完成的.虚构件在其中扮演了最为重要的角色.正如前面描述的一样,开发人员只要在销售商的网络上寻找合适的虚构件,然后设置并执行它,按照它的提示购买相应的小构件,这样就可以得到想要的复合构件了.当然,这个时候的销售商已经不是纯粹的销售了,而是

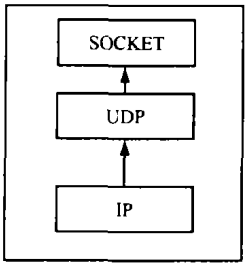

图4　速度优先生成的复合构件

Fig. 4　Speed-first composed component

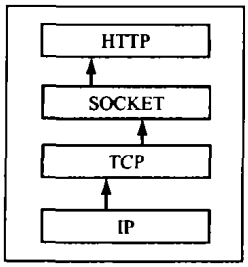

图5　快速稳定的复合构件

Fig. 5　Easiness and reliability-first composed

构件(虚构件)的开发商.开发与发售模式如图6所示.

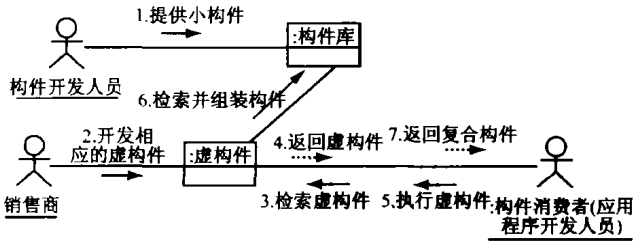

图6　组件开发与销售模式

Fig. 6　Mode of component development and sales

3　结论和今后的工作

构件技术的进步使得基于构件的开发(CBD)越来越流行.但现在的构件模型还有许多不足,基于构件的软件开发(CBSD)技术也尚未完善.我们吸取了众家之长,并提出了一些新的概念和方法,最终提出了一种比较完善的(当然远未达到完美)构件模型.在探讨了当今最新的CBSD技术之后,为解决构件的检索和组装而提出了一个新技术"虚构件",它能较好地实现领域或专门知识的封装,可以帮助构件的复用人员利用构件库中的构件复合出符合特定要求的构件.它实现了构件复合(组装)的某种意义上的自动化(半自动化).

参考文献:

[1] Maurizio Morisio, Michel EZran, Colin Tully. Success and failure factors in software reuse[J]. IEEE Transaction on Software Engineering, 2002, 28(4): 340－357.

[2] 杨芙清. 软件复用及相关技术[J]. 计算机科学, 1999,26(5):1－4.

[3] 北京大学青鸟项目组. 青鸟构件模型[R]. 北京:北京

大学计算机科学系,1997.

[4] 北京大学青鸟项目组. 青鸟构件描述语言[R]. 北京:北京大学计算机科学系,1997.

[5] 张世昆,张文娟,常欣,等. 基于软件体系结构的可复用构件制作和组装[J]. 软件学报,2001,13(9):1451-1459.

[6] Maritta Heisel, Thomas Santen, Jeanine Souquieres. Toward a formal model of software components[A]. Chris George and Huaikou Miao Eds. 4nd International Conference on Formal Engineering Method (ICFEM 2002)[C]. Shanghai ,China,2002. 57-68.

[7] Liu Jing, Miao HuaiKou, Gao XiaoLei. A specification-based software construction framework for reuse[A]. Chris George and Huaikou Miao Eds. 4nd International Conference on Formal Engineering Method (ICFEM 2002)[C]. Shanghai,China,2002. 69-78.

[8] Microsoft Company . MSDN [EB/CD]. 2002.

[9] Zaremski A M, Wing J M. Signature matching: A tool for using software libraries[J]. ACM Trans on Software Engineering and Methodology, 1995, 4(2):156-180.

[10] Marry Shaw, David Garland. Software Architecture [M]. Prentice Hall, 1996.

下期发表论文摘要预报

用于行对角占优信道矩阵的MIMO盲均衡算法

刘谦雷, 杨绿溪

(东南大学 无线电工程系,江苏 南京 210096)

摘 要: MIMO通信信道提出了一种基于矩阵初等变换的均衡算法,将其用于行对角占优的MIMO系统,则得到相应的MIMO盲均衡算法. 该算法基于通信源信号的统计特性、利用数学上通过矩阵初等变换实现矩阵对角化的原理对MIMO信道矩阵实现了对角化. 将该盲均衡算法用于常规的MIMO系统,则可以克服系统因突发干扰等因素产生的常规均衡器均衡参数的估计偏差、提高系统的误码性能. 仿真实验证实,本文提出的MIMO盲均衡算法在行对角占优信道矩阵的均衡和常规均衡器均衡参数估计偏差的克服中均取得了良好的效果.