维普资讯 http://www.cqvip.com

PCCM:具有性能约束的构件模型

卢炎生　查虎平　徐丽萍

(华中科技大学计算机科学与技术学院　武汉430074)

摘　要　构件模型是构件复用的基础。本文根据3C和REBOOT模型提出一种具有性能约束的构件模型,实现具有性能约束的构件复用。首先简要分析了构件模型的研究现状以及复用现状,引出了在特殊领域的具有性能要求的构件复用;然后提出具有性能约束的构件模型及描述;接着给出实现关键技术的路线,并对模型进行了优点分析;最后指出了进一步的研究方向。

关键词　软件复用,软件构件,构件模型,性能约束

PCCM: Performance-Constraint Component Model

LU Yan-Sheng　ZHA Hu-Ping　XU Li-Ping

(Department of Computer Science and Technology, Huazhong University of Sci.&Tec., Wuhan 430074)

Abstract　Component model is the basis of component reuse. The paper is to synthesize 3C and REBOOT component models to put forward a Performance-Constraint Component Model-PCCM to implement component reuses with performance constraints. Firstly analyze research of component models and reuse status briefly to educe reuse requirement with performance constraints in special domains. Then a component model with performance constraints is proposed, description is specified. After that key technology route of implementation and excellence of the model are analyzed. Finally further research direction is given.

Keywords　Software reuse, Software component, Component model, Performance constraint

1　引言

软件复用技术已经在软件的开发中给软件业带来了多方面的益处,推动了软件标准化,大大缩减了软件开发的时间和资源,为软件业开辟了一条捷径。规范的构件描述和表示是进行良好复用的基础,对于构件描述,比较典型的模型有3C模型和REBOOT模型。

3C模型[1]是学术界普遍认同的一个具有指导性作用的构件模型,从概念、内容和语境三个方面来描述构件,具有很强的表达描述能力。概念是关于"构件做什么"的抽象描述;内容是概念的具体实现;语境是描述构件和外围环境在概念和内容级的关系,进一步可分为:概念级语境(Conceptual Context)—描述构件间接口和语义方面的关系,操作级语境(Operational Context)—刻画构件中被操作数据的特征(如类型和操作)以及实现级语境(Implementational Context)—描述构件间在实现方面的依赖关系。REBOOT模型[2]是一个基于已有软构件的刻面分类和检索模型,用有限维信息空间的术语组合从若干个刻面的综合角度来刻画一个构件,将构件的描述术语空间划分成不同的刻面,有利于构件的查询检索。具体到REBOOT项目中提出了4个刻面:依赖、抽象、操作和操作对象。

三大主流工业标准构件模型是OMG的CORBA/OM、MICROSOFT的OLE/DCOM和JAVA的Enterprise JavaBean。CORBA/OM从属性、功能接口和依赖关系来描述构件。这些模型的关键技术提供了对各种功能模块进行构件化处理以及在此基础上进行功能复用。还有RESOLVE模型以及北大青鸟构件模型等。北大青鸟构件模型[3]采用刻面分类模型的思想,定义了使用环境、应用领域、功能、层次和表示方法5个刻面。

从上述模型可以看出以往对复用的研究侧重于功能复用,模型侧重于构件功能以及构件功能接口相关的内部和外部接口交互的描述。但是要使软件开发真正地走上工程化和产业化道路,那么各个领域都应该可以实现复用。不同的领域有各自的特点,必然导致对复用的要求不仅仅是功能,针对领域特点还会有相应的性能要求,例如军事、航空航天领域的应用对时间的要求很高。因此要将软件复用真正广泛深入到各个领域中去,具有性能约束的软件复用是必然的也是必需的。

本文提出一种具有性能约束的软构件模型PCCM(Performance-Constraint Component Model),实现具有性能约束的构件复用。它的主要特点是综合3C模型和REBOOT模型的思想,进行扩充,增加性能描述和性能保障机制,既能很好地描述和体现构件的3C特性,也能从多个刻面来刻画构件,从而为构件的提取提供更多的刻面信息,提高提取的速度,使之更好适用于具有较高性能要求的应用领域。根据复用对象的不同,可以将复用分为过程复用和产品复用[4],本文只对面向产品复用的构件进行描述。

2　具有性能约束的构件模型PCCM

2.1　理论基础

一个软件系统的体系结构定义了组成系统的构件与构件之间相互作用的关系[5]。PCCM采用插头插座式体系结构,它具有较好的性质,在构件集成时不必关心内部的实现细节,支

*)本文得到国防预研基金项目(10104010201)资助。卢炎生　教授,博士生导师,主要研究方向为软件工程、特种数据库。查虎平　硕士,研究方向为软件工程。徐丽萍　副教授,研究方向为软件工程。

维普资讯 http://www.cqvip.com

持构件在规约层次上的集成[5]。

首先引入两个概念[5]。

定义1 原子构件是在系统开发中无须再分的最小基本单元，有其对应的实现体。

定义2 复合构件是由多个原子构件组装而成的大粒度构件，在规约层次上表达了成员构件之间的复合。逻辑复合构件本身不对应于任何实现体，物理复合构件有对应的实现体。

复杂的系统由更低层次的复杂子系统组装而成，可以看作多层次、包含多个嵌套软构件的大粒度复合构件。

定义3 构件模板是一类提供相似功能的构件的抽象描述，构件是构件模板的一个实例，正如面向对象中对象是类的实例。

定义4 T=〈F，A〉，这里我们用 T 代表模板 Template，F 代表框架 Frame，A 代表体系结构 Architecture[6]。

F 定义了任何 T 的一个实例构件所拥有的接口集 Interfaces，包括输入接口 Requires-Interface（代表了环境为它提供的服务）和输出接口 Provides-Interface（代表了它为环境提供的服务接口），即 Interfaces＝{Provides-Interfaces，Requires-Interfaces}。F 反映了 T 的黑盒视图。如果将 T 看作是一个有穷状态机，那么 Requires-Interface 可以看作是有穷状态机的输入 Input，Provides-Interface 可以看作是输出 Output。

体系结构 A 描绘了模板内包含的子构件之间的交互。一般地，对于一个输入输出固定的有穷状态机，它的内部构造不是唯一的，可以通过多种构造组合来实现这样的输入输出的有穷状态机。同样这样的原理也适用于构件模板，对于一个 F，可以用多个 A 来实现。一个 A 描绘了 F 的一个实现结构，并通过子构件的接口联系来规定子构件间的交互。可以说一个 A 反映了 T 的一个特定灰盒视图。T 模板的图形化表示如图1。

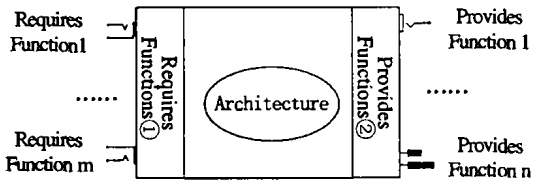

图1 模板示意图

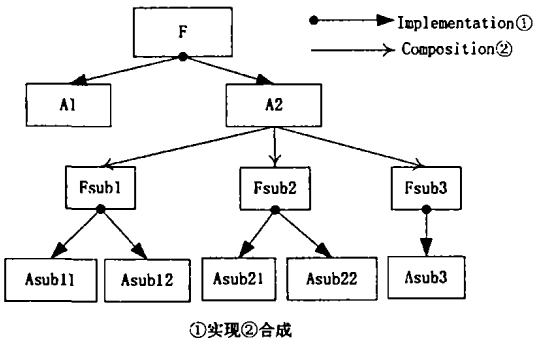

图2 模板层次结构图

在设计阶段 F 和 A 确定一个构件。从图2可以看出，F 可以实现为 A_1，A_1 实现为一个原子构件，同样 A 也可以实现为 A_2，A_2 由子构件 sub1、sub2和 sub3组装而成，在 A_2 所在的层次只能看到 sub1、sub2和 sub3的 F_{sub1}、F_{sub2} 和 F_{sub3}，这样交替下去形成构件的嵌套组装结构。A 的内部细节和 F 的分离可以实现每一个子构件方便的替换，从而实现软件的“即插即用”。

2.2 设计原理

构件模型的确立和具体的软件方法学有着密切关系，不同范型的方法学必然导致不同的模型。OO 技术能对软件复用提供更有力的支持，面向对象开发过程的各个阶段可交付使用的特点使得不仅仅是代码支持复用，需求分析和设计都支持复用[7]，因此 PCCM 遵循 OO 范型，其构件结构符合 OO 风范。

一个构件的规约描述中描述的功能是不随时间、空间的不同而变化的，而性能与功能的不同之处在于：性能是动态体现的，随时间、空间的不同会发生变化。同一个构件在同一个系统上不同时间所表现的性能往往是不同的，因为性能与运行时的系统环境密切相关。所以构件描述应包含足够详细的性能信息，以使构件被调用时可以利用详细的性能描述信息来进行构件运行环境配置。具体的性能需求要具体分析，对于实时来说由于截止期、任务的同步以及共享资源对达到实时起主要的作用，因此构件必须提供对应性能下的共享资源、线程数、线程优先级以及线程同步等信息使得系统利用这些信息达到实时要求。

为了模型的通用性，构件所在的系统平台不做特别要求，那么要达到要求的性能就难以完全依赖于操作系统来实现，因此需要一个性能保障机制保障构件在运行时达到要求的性能，不同的性能约束设置对应的性能控制接口来利用构件提供的性能信息设置构件的运行环境来达到要求的性能，譬如为了实现实时，利用实时控制接口规定构件运行的开始和结束时间以及设置构件运行时的线程数、线程的优先级等等触发系统来实现。

因此 PCCM 除了功能接口维，为了达到性能约束的要求增加了性能接口维，形成了{功能接口，性能接口}二维构件模型。

定义5 T=〈F，A〉。其中 F=〈Function，Performance〉，A 的解释同定义4。

1）功能维：强调功能的描述，每一个具体构件接口不一样；

2）性能维：可根据性能要求设置多个性能控制接口 $PMcontroller_1$，$PMcontroller_2$，…，$PMcontroller_k$ 来达到要求的性能控制，不同的性能需求对应于不同的性能控制接口。

PCCM 示意图如图3所示。

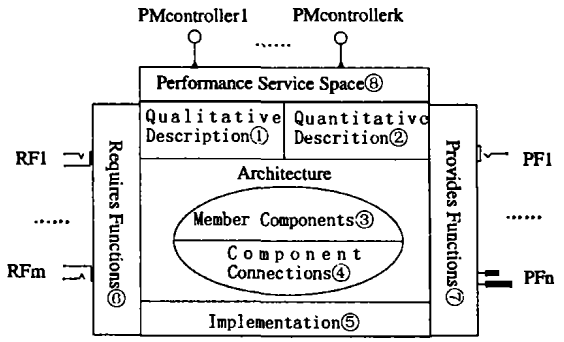

图3 PCCM 示意图

图中：定性化描述指如构件名称、编写语言、软硬件环境、

维普资讯 http://www.cqvip.com

关键词、包含的文件、作者、发布日期、使用次数等等的传统意义上的描述方面；

定量化的描述指性能描述，如和实时对应的有最坏执行时间(配套的系统资源的提供以及配置，如共享资源、线程数、线程优先级、线程同步)等；

性能服务空间是性能控制接口的集合，通过这些接口来保障尽可能地达到期望的性能。主要工作有：读取定量化描述(性能描述)；根据性能描述初始化构件运行环境；调用构件的功能接口。

2.3 模型描述

PCCM 主要从如下5部分来描述构件，突出构件的性能描述以及性能控制接口，注重构件的易理解性、封装性及构件间关系，通过给构件提供明确的对外接口实现服务提供者和其服务请求者的分离。

(1)一般描述。包括定性化描述和定量化描述，这些描述为构件库提供描述空间，是构件提取的基础。

(2)输入输出功能接口描述。分离了服务提供者和服务请求者。输入功能接口代表了环境为它提供的服务，对应于3C的 Conceptual Context，输出功能接口代表了它为环境提供的服务对应于3C 的 Concept。

(3)性能控制接口描述。体现性能约束。对一个合成构件的性能要求根据合成策略(合成策略将会在其他相关的文章中介绍)将性能约束分解给 A 中的各子构件对应的性能控制接口，……，如此循环下去，最后性能约束总可以从一个大粒度合成构件分解到最低层次的各个子构件。

(4)成员构件以及接口之间的联系。对于复合构件要声明包含的构件，同时构件之间存在着如下几种接口之间的联系：

i) 接口的绑定：将外部接口分配到 A 中的成员构件接口；

ii) 接口的连接：成员构件之间的 Provides-Function、Requires-Function 的连接(内部接口)；

iii) 接口的作废：声明某些成员构件的某些接口是不需要的。

(5)实现体。构件的具体实现部分，是实际完成被请求服务的系统。对应于 3C 的 Implementational Context 和 Content，一般的复用者不关心这一部分，除非进行白盒复用。

PCCM 的 BNF 描述：

```
构件::=〈一般描述，构件规约，构件实现〉
一般描述::=〈定性化描述，定量化描述〉
定性化描述::=〈一般描述面〉
定量化描述::=〈性能描述面〉
构件规约::=〈接口规约，结构规约〉
接口规约::=〈功能接口，性能接口〉
功能接口::=〈对外请求的功能集合，对外提供的功能集合〉
性能接口::=〈性能控制的功能集合〉
结构规约::=〈原子构件结构〉|〈复合构件结构〉
原子构件结构::=〈构件实现的引用〉
合成构件结构::=〈引用的成员构件类型，实例声明，接口联系〉
接口联系::=〈内部接口和外部接口的绑定，内部接口间的连接，内部
          接口的作废〉
构件实现::=〈构件实现体地址的引用〉
```

2.4 一个实例

下面给出实时性能约束作为例示。

实时体现在对时间的要求上，要求事务的执行或者是快的响应速度，或者是恰到好处，既不快又不慢。对构件运行的实时要求体现为及时地返回结果。对构件运行时间性能起关键作用的有运行时的线程数、线程优先级协议、线程同步、共享资源等等。

构件包含的性能信息为：

WorstRuntime (Thread, ThreadPriorityProtocol, ThreadSynchronization, ShareSource,……)

一次具有实时要求的构件调用过程为：

① 应用程序向中间件机制发出请求 function(data, timeconstraint)；

② 在构件库中查询满足 function 和 timeconstraint 的构件；

③ 中间件机制向选中的构件发出请求 function(data, timeconstraint)；

④ 请求传递到构件的实时控制接口 RTCI(Real-Time-Control-Interface)；

⑤ RTCI 获取构件的性能描述信息，初始化构件运行时的环境，如线程、线程优先级、线程优先级协议；

⑥ 在⑤的环境下根据③调用各功能接口；

⑦ 返回结果给中间件。

3 实现技术路线及模型优点

3.1 实现技术路线

现有的软构件技术已经实现了功能的复用，我们在此基础上施加了性能约束。构件性能评价在国外已经有了一些研究[7]。那么将功能复用技术和构件性能评价相结合，加以合理的系统资源配置以及调用，就可以实现具有性能要求的软件复用。

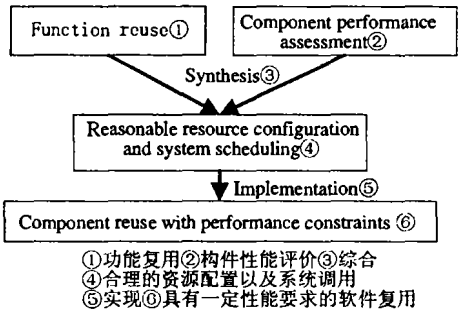

图4 具有性能约束的构件复用实现技术路线

3.2 模型优点

PCCM 模型突出点在于不仅考虑了复用的功能要求而且考虑了在特殊领域复用的性能要求，使构件具有详尽的性能描述；在模型的设计上将构件的一般描述分为定性化描述和定量化描述两部分，这种分离利于构件的检索，对于不具有性能约束的复用，就不必检索定量化描述空间，节省了检索的开销，又兼顾了具有性能约束的构件检索；构件运行时性能的保障不依赖于所在平台，设置性能保障机制来保障构件运行时的性能。

结束语 PCCM 是一个通用的模型，不仅可以描述一般的和具有性能约束的构件，重要的是在此基础上可以不断地采用继承的思想来实现具体性能约束的构件模型。我们还将进一步开展相关性能及相应支撑技术研究，如构件描述语言 CDL、构件接口定义语言 CIDL、CIDL-to-C++的编译，并构造出一个实用的模型。

参 考 文 献

1 Whittle B, Ratcliffe M. Software component interface description for reuse. Software Engineering Journal, 1993, 8(6): 307～318

2 Morel J, Faget M J. The REBOOT environment. Software

维普资讯 http://www.cqvip.com

Reusability. In: Proc. Advances in Software Reuse, 1993. 80～88
3 http://home.sei.pku.edu.cn/95
4 Mili H, Mili F, Mili A. Reusing software: Issues and research directions. IEEE Transactions on Software Engineering, 1995, 21(6)
5 张世琨,张文娟,常欣,王立福,杨芙清. 基于软件体系结构的可复用构件制作和组装. 软件学报,2001,12(9):1351～1359
6 Plasil F. Behavior protocols for software components. IEEE Transaction on Software Engineering, 2002,28(11): 1056～1076
7 Opdahl A L, Sindre G, Vetland V. Performance consideration in Object-Oriented reuse. Selected Papers from the Second International Workshop on, 1993. 142～151
8 Stephen Sau, Taweponsomkiat C. An approach to object-oriented component customization for real-time software development. Computer Society, 2002
9 David L, Lary C, Augustin M. Specification and analysis of system architecture using Rapide [J]. IEEE Transaction on Software Engineering, 1995, 21(4): 336～355
10 Medvidovic N, Rosenblum D S. Domains of concern in software architecture and architecture description [A]. SHAW M. Proc. of' 97 USENIX [C], California, 1997. 199～212

(上接第88页)

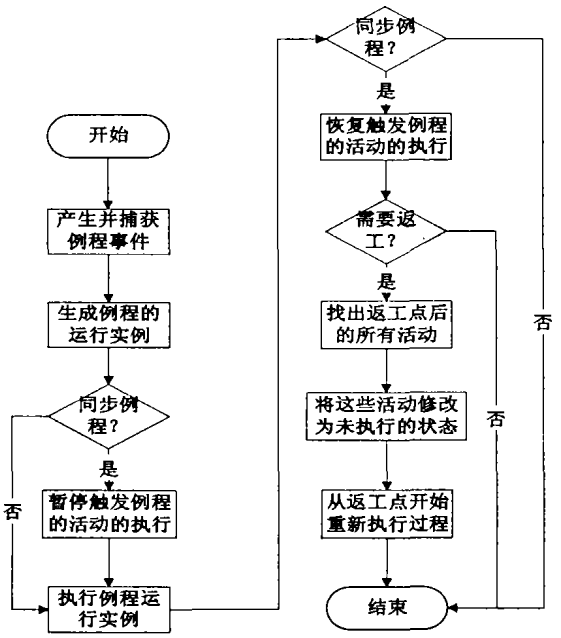

图3

2. 执行例程

例程运行实例的执行与其它过程的执行是完全相同的。

3. 例程结束的处理

例程运行实例执行结束后,过程引擎首先判断例程的类型。如果是异步例程,过程引擎不需要进行任何处理。

如果是同步例程执行结束,过程引擎首先将触发该例程的活动的状态修改为正执行的状态,以恢复该活动的执行。

此外,如果需要返工,过程引擎找出过程程序中在返工点后的所有活动,将这些活动的状态修改为未执行的状态,然后从返工点开始重新执行过程程序。

小结 CPMS 是一个基于 CMM 的过程支持系统,它需要能够支持符合 CMM 规范的过程。而 CMM 实际应用中除了一般的软件过程外,还存在着一类特殊的过程——庇护性过程,并且 CMM 的实现离不开庇护性过程的支持。

经过对庇护性过程的特性进行分析抽象后我们发现,能够支持庇护性过程的过程模型必须具有随机启动、流程固定、可返工三个特性。其中,随机启动是最本质的特性,并且它很类似于程序设计语言中的异常处理。而通过将过程模型与程序设计语言进行类比发现,过程模型的三种基本结构只具有确定性,不足以实现随机启动的特点。因此对过程模型扩充了类似异常处理的机制,在 CPMS 中,该机制称为例程处理机制。

CPMS 的例程处理机制的设计参考了 Java 的异常处理机制,同时根据应用需求进行适当的修改,主要表现在使用固定的例程处理程序、不中断过程程序而能够恢复其执行、可返工三个方面。

本文给出了 CPMS 的例程处理机制的详细描述,并给出实现方案。

致谢:本文受到了国家863项目"基于 CMM 的软件质量保障平台及应用"(2001AA113090)支持。

参 考 文 献

1 Strohmeier A, Chachkov S. A Side-by-Side Comparison of Exception Handling in Ada and Java, Version 1.1. ACM Press, 2001
2 Ellmer E. Three Perspectives on Process Supporting Environments: Integrating Workflow and Software Process Research. In: Proc. of the Intl. Symposium on Applied Corporate Computing, Oct. 1995. 25～27
3 Goodenough J B. Exception handling: Issues and a proposed notation. Communications of the ACM, 1975, 18(12)
4 Osterweil L. Software processes are software too. In: Proc. of the Ninth Intl. Conf. on Software Engineering. IEEE, 1987
5 Jalote P. CMM in Practice - Processes for Executing Software Projects at Infosys. Pearson Education, Inc., 2000
6 Bruynooghe R F, Parker J M, Rowles J S. PSS: A System for Process Enactment. In: Proc. of the First Intl. Conf. on the Software Process. IEEE Computer Society Press, 1991
7 Humphrey W S. Managing the Software Process. Addison-Wesley, 1989
8 Pressman R S 著,梅宏译. 软件工程——实践者的研究方法(第5版). 机械工业出版社,2002
9 卡耐基梅隆大学软件工程研究所编著,刘孟仁等译. 能力成熟度模型(CMM):软件过程改进指南. 电子工业出版社,2001