第32卷 第23期 计 算 机 工 程 2006年12月
Vol.32 No.23 Computer Engineering December 2006
·软件技术与数据库· 文章编号：1000—3428(2006)23—0055—03 文献标识码：A 中图分类号：TP311

基于通用连接器模型的复合构件的组装

许 毅，彭 鑫，赵文耘
(复旦大学计算机科学与工程系，上海 200433)

摘 要：基于构件的软件体系结构(SA)由构件与连接器组成，连接器作为构件间的交互实体在SA中扮演着重要角色。现有的连接器模型只能支持特定软件体系结构风格的组装，该文针对构件组装的核心问题：构件间交互的不匹配，采用复合构件组装的方法，提出一种通用连接器模型用以对不同连接器模型进行描述。给出4种复合构件组装机制及各种机制的组装方式，并对复合构件进行扩展，使其具有动态性，能较方便地实现动态体系结构。

关键词：软件体系结构；连接器；复合构件；动态体系结构

General Connector Model Oriented Complex Component Composition

XU Yi, PENG Xin, ZHAO Wenyun
(Department of Computer Science and Engineering, Fudan University, Shanghai 200433)

【Abstract】The component-oriented software architecture (SA) is composed by components and connectors, in which connector plays a very important role. Nonetheless, the existing connector models can only allow the specific SA style, and lacks a general connector model to support. A general connector model is presented to describe different connector types. Now, components' interactions are becoming the core of components' composition, an effect way to conquer the problem is using complex component. Based on the general connector model, four composition mechanisms are presented to fit component composition and dynamic software architecture.

【Key words】Software architecture (SA); Connector; Complex component; Dynamic software architecture

软件体系结构(Software Architecture, SA)是对软件总体结构的描述，即对其构件和构件间交互的高层组织的描述[1]。它作为一种高层的设计，对系统开发发挥着重要的影响，基于构件的SA的设计已成为软件系统设计中的核心问题。一个好的SA设计成为大型软件系统成功的重要因素。SA的最重要的一个贡献是将构件之间的交互显式表示为连接器(Connector)，并将连接器视为和构件同等重要的一阶实体。针对不同的SA风格，人们提出了众多的连接器模型，而连接器的研究也成为SA研究的热点问题。

同时，基于SA的构件组装(component composition)和组装推导(compositional reasoning)已成为基于构件的软件工程的关键技术。而在构件组装过程中，构件间交互的不匹配是其核心问题，一般通过制作复合构件及添加非功能属性来实现。文献[2]中采用6种构件组装机制来实现复合构件的制作。

本文提出了一个通用的连接器模型，模型中连接器由更细粒度的行为元素组成，非功能属性也被规约成行为元素。它能描述现有的各种连接器类型，而且还可以支持新的连接器类型，从而实现不同的SA风格。同时提出4种复合构件的组装机制，使用通用连接器模型来实现各种机制。通用连接器模型可以具有非功能属性，从而复合构件也具有非功能属性，并且利用通用连接器内部是基于消息的特征，将复合构件扩展为基于动态体系结构结构。

1 相关工作

SA中连接器的研究一直是一个研究热点，和本文相关的研究有：软件体系结构，连接器模型，复合构件。

在软件体系结构方面，文献[3]中，连接器被认为是和构件同等重要的一阶实体。连接器是软件体系结构设计层次上的一个概念，它作为构件间的交互应清晰地描述交互语义。对连接器进行系统的分类工作已经在进行[4]。同时构件间交互的不匹配是构件组装的难点，也是构件组装的核心问题，为此非功能属性被广泛地使用。文献[5]提出了使用包装器(wrapper)的方法来实现非功能属性。文献[5,6]将非功能属性作为连接器的一部分，可以自定义地添加。

在连接器模型方面，文献[6]中提出了一个细粒度的连接模型，它将非功能属性作为一个附加属性添加在连接器上，该模型与本文模型的最大区别在于前者没有将构件接入与消息转化的行为元素独立出来，而本文的模型使用单独的行为元素来完成转化功能，同时还有一个控制中心来对连接器中各行为元素进行控制。

在复合构件方面，文献[2]分析了构件组装及复合构件的组装机制，同时将复合构件组装分为6类：并行组装，选择组装，复制组装，顺序组装，中断组装，连接组装。而本文将采用通用连接器模型完成各种类型复合构件的组装，同时组装出的复合构件可以具有非功能属性及动态性，以适应动态体系结构的需求。

基金项目：国家“863”计划基金资助项目(2004AA112070，2004AA113030，2005AA113120)；国家自然科学基金资助项目(60473061)
作者简介：许 毅(1981－)，男，硕士，主研方向：软件工程；彭 鑫，博士；赵文耘，教授
收稿日期：2005-12-23 **E-mail：**032021162@fudan.edu.cn

2 通用连接器模型

在基于构件的系统中，连接器作为构件间交互的实体，不同的 SA 风格中构件的交互方式也不尽相同，因此需要不同类型的连接器。连接器只是一个抽象的概念，不像构件那样有精确的描述，针对以上的需求，本文提出了一个连接器通用模型，连接器是由更细粒度的行为元素组成。

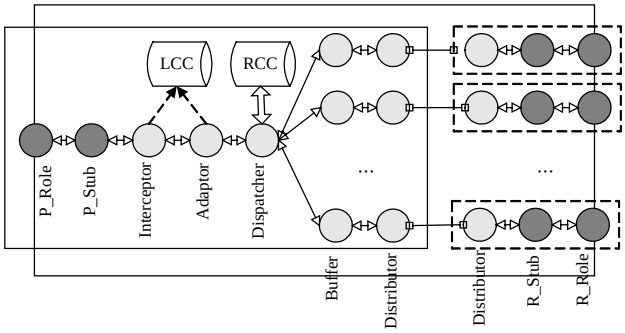


图 1 通用连接器模型

通用的连接器模型如图 1 所示，模型允许构件以各种方式接入，其内部使用消息通信。模型中各行为元素间通过通道相连，通道不形成回路。一个通道包括 2 部分：1 个正向信道，1 个逆向信道。Glue 内部通道是基于消息方式的，Glue 内部的行为元素应遵循统一的消息标准。Glue 内部的行为元素应该有缓存消息功能。模型包含 1 个逻辑控制中心(Logic Control Center, LCC)、1 个路由控制中心(Route Control Center, RCC)和各种行为元素(Element)。其中各行为元素的具体作用如下：

R_Role、P_Role：分别为服务构件和请求构件的接入点。

R_Stub、P_Stub：将各式各样的接入方式同消息进行相互转化。

Distributor：实现分布式这个非功能属性，图 1 中虚线框可单独分布在一个地址空间。

Dispatcher：负责各接口间的消息分派；当在连接器模型中选用了该行为元素时，则必须选用 Route Control Center 为之服务，如图 1 所示。

Adaptor：完成服务、请求构件间不匹配的转化；例如：完成参数匹配，类型转化。

Interceptor：决定是否阻塞该消息；或者返回该消息。

Buffer：决定是否缓存消息。

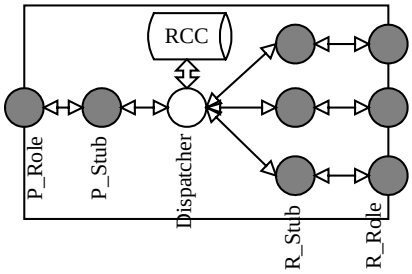
模型中两个控制中心的作用分别为：

LCC：作为连接器的逻辑中心，与具体系统的领域信息密切相关。负责处理连接器内部消息内容及格式的逻辑处理。逻辑控制中心应为各行为元素提供统一的接口，该接口接收消息的内容作为参数，并将进行处理后的结果作为返回值返回。

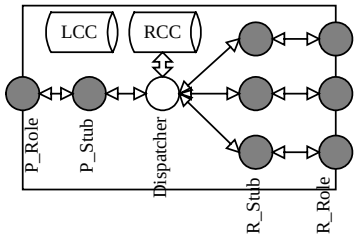
RCC：服务于 Dispatcher 行为元素；由于 Dispatcher 的扇入、扇出关系是一对多的关系，因此需要 RCC 为其提供路由选择功能。路由控制中心为 Dispatcher 提供统一的接口，该接口接收消息的内容作为参数，并将目的行为元素的地址作为返回值返回。

LCC 和 RCC 均与具体的系统领域知识相关，所以这部分需要进行手工编码。路由控制中心和必需行为元素在组成连接器时是必不可少的，而可选行为元素则可以按照设计者的意图进行插拔。通过行为元素的不同组合和配置，可以完成不同构架风格中所需连接器的功能，同时提供一些非功能属性。

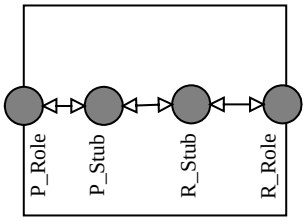
通过通用连接器模型，可以描述各种类型的连接器模型，基于文献[3,10]中对连接器的分类，本文对 3 大类连接器模型进行描述：(1)过程调用型；(2)消息型；(3)数据流型。通过对通用连接器模型中行为元素的选择，可以实现以上 3 大类连接器类型。易于操作，在各类连接器中不加入任何实现非功能属性的行为元素。



(a)接口连接式连接器模型



(b)C2 连接器模型



(c)管道-过滤器连接器模型

图 2 各种类型连接器模型

过程调用型接口连接式风格模型如图 2(a)所示。

R_Role、P_Role 作为过程、方法调用的行为规约接口，过程、方法和 Glue 内部消息一一对应。RCC 中记录了 Glue 内部消息分派的配置信息。Dispatcher 是完成服务构件与请求构件之间方法匹配的实际操作者，当 Dispatcher 接收到一个消息时，它查询 RCC 来获取该消息的目的地，从而将消息发往具体的 Stub。

限于篇幅，只对过程调用型进行描述，以上 3 种类型的连接器均可以自由地添加非功能属性行为元素，如添加 Adaptor 行为元素完成消息数据格式的转化。

3 复合构件组装

在基于构件的软件开发的过程中，遇到的一个棘手问题就是所选择的构件之间交互的不匹配。主要包括：(1)新选择的构件与其它构件所采用的数据格式不一致；(2)构件的输出可能溢出其后继构件的输入；(3)系统运行在不同的情况下，使用不同类型的具有相同功能的构件；(4)某些构件可能需要其它几个构件同时提供输入才能运行；(5)构件库中可能没有满足需求的构件，需要几个构件组装在一起来提供一个逻辑上完整的功能。基于以上问题，提出了 4 种基本构件组装机制：(1)修饰组装；(2)顺序组装；(3)选择组装；(4)并行组装。4 种组装机制相互补充，可以形成更加复杂的组装。

接口绑定是建立复合构件组装的外部接口和内部接口的对应关系。可以通过连接器完成复合构件中构件的接口绑定，通过各种不同的连接器可以完成一些复杂的接口绑定，以实现所需的复合构件。基于通用连接器模型的复合构件组装使用通用连接器模型来进行外部接口和内部接口的绑定。由于通用连接器据具有可扩展的特性，因此由其组装成的复合构

件也具有可扩展的特性，同时通用连接器内部是基于消息传递的，从而使组装出来的复合构件具有动态性。同时复合构件内部应该有一个配置中心(Configuration Center, CC)，用于保存复合构件各组成部分的连接情况和配置信息。

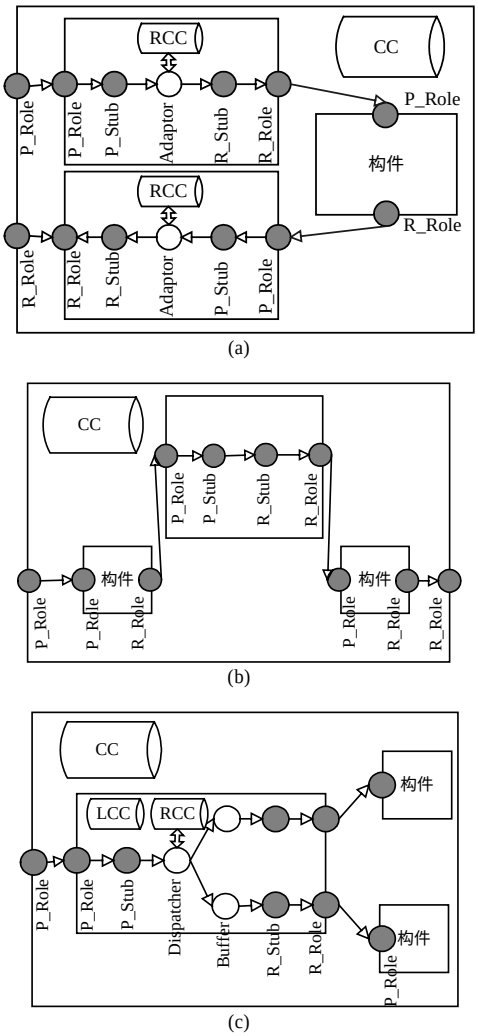


图 3 各种类型复合构件组装

使用通用连接器模型可以实现 4 种基本的组装机制：

(1)修饰组装

修饰组装模型如图 3(a)所示。

在构件的输入和输出端各设置一个连接器，连接器中包含行为元素 Adaptor 和 Interceptor。Adaptor 可以进行输入输出格式的转化，Interceptor 可以控制是否阻塞相应的消息，从而解决了构件输出溢出其后继构件的输入的问题。行为元素 Adaptor 将消息内容作为参数传送给 LCC，LCC 将返回转化消息格式后的消息内容，行为元素 Interceptor 将消息内容作为参数传送给 LCC，LCC 将返回是否阻塞该消息的应答。

(2)顺序组装

顺序组装模型如图 3(b)所示。

数据流将依次通过各个构件，各构件完成对相应数据流的处理后，将数据流输出。 任意两构件之间通过连接器进行相连，连接器在顺序组装过程起到连接和缓存数据的作用。

(3)选择组装

选择组装模型如图 3(c)所示。

通过一个连接器来将多个构件的内部接口绑定到一个外部接口，连接器内部通过行为元素 Dispatcher 来分派消息到指定的 Stub 上。当 Dispatcher 收到一个消息时，它将消息内容作为参数传送给 RCC，RCC 根据消息内容查询该消息的目标 Stub，然后将该 Stub 作为结果返回给 Dispatcher。

(4)并行组装

同选择组装相似，通过一个连接器来将多个构件的内部接口绑定到一个外部接口，连接器内部通过行为元素 Dispatcher 来分派消息到指定的 Stub 上。与选择组装不同的地方就是当 Dispatcher 收到一个消息时，它将消息广播到各个 Stub，各个构件在接收到消息后并行运行。

以上 4 种构件组装机制不是绝对对立的，它们之间可以互相组合，例如修饰组装可以修饰其余各种组装机制。通过这种途径，复杂的复合构件可以容易地生成。

由于连接器模型内部采用了基于消息的调用方式，添加一个缓存功能行为元素，该行为元素在接收到消息时会查询 LCC 以获得是否缓存消息的应答，因此可以很方便地实现复合构件的动态演化。同时一个 SA 可以看成一个复杂的复合构件，从而实现了一个多层次、全面的 SA 的动态演化模型，如图 3(c)所示。

当复合构件的 CC 收到一个替换消息时，它将分析是哪个连接器控制该构件的输入，然后通知相应的连接器，连接器的 LCC 记录哪个 Buffer 行为元素需要缓存数据，这样，该 Buffer 行为元素将缓存在此之后的所有消息。同时 LCC 会给 CC 一个应答，CC 在接收到该应答后，查询需要替换构件是否处于繁忙状态，当替换构件处于空闲状态则替换它，否则等待直到它处于空闲状态。当所需的构件被替换后，CC 通知相应连接器 LCC 取消 Buffer 行为元素对消息的缓存，在这一个过程中，保证了整个系统状态的一致性和正确性。

4 结论及下一步工作

软件工程的一个研究热点是对 SA 的研究，本文着眼于连接器，提出了通用连接器模型，对不同连接器类型进行了描述；而且可以自定义连接器并加入非功能属性；基于通用连接器模型，完成了 4 种复合构件组装机制描述，并对其在非功能属性和动态性上进行了扩展，从而使复合构件能轻易地使用在动态体系结构中。

本文已经在工具开发中得到应用，进一步的研究工作包括在通用连接器模型的基础上进行相关 Case 工具的开发、连接器代码的自动生成，并进行更多的实例建模和分析。

参考文献

1 Garlan D, Shaw M. An Introduction to Software Architecture(First Edition)[M]. New Jersey: World Scientific Publishing, 1993: 1-39.

2 任洪敏，钱乐秋. 构件组装及其形式化推导研究[J]. 软件学报, 2003, 14(6): 1066-1074.

3 Allen R, Garlan D. A Formal Basis for Architectural Connection[J]. ACM Trans. on Software Engineering and Methodology, 1997, 6(3): 213-249.

4 Mehta N R, Medvidovic N. Understanding Software Connector Compatibilities Using A Connector Taxonomy[C]. Proceedings of the 1st Workshop on Software Design and Architecture, Bangalore, India, 2002.

5 Spitznagel B, Garlan D. A Compositional Formalization of Connector Wrappers[C]. Proceedings of the 25th International Conference on Software Engineering, 2003: 374-384.

6 Bures T, Plasil F. Scalable Element Based Connectors[C]. Proc. of SERA, San Francisco, USA, 2003.