第33卷 第23期 计 算 机 工 程 2007年12月
Vol.33 No.23 Computer Engineering December 2007
·软件技术与数据库· 文章编号：1000—3428(2007)23—0043—02 文献标识码：A 中图分类号：TP311

基于组合连接器的潜在无限动态结构的规范

虞莉娟[1]，熊惠民[2,3]，应 时[3]
(1. 武汉理工大学自动化学院，武汉 430070；2. 华中师范大学数学与统计学学院，武汉 430079；
3. 武汉大学软件工程国家重点实验室/计算机学院，武汉 430072)

摘 要：动态软件体系结构的建模与分析是复杂软件系统设计的一个重要问题。基于体系结构描述语言Wright，提出了一种规范潜在无限动态结构的形式化方法。为了便于使用递归机制，引入了组合连接器和动态角色的概念，从而实现了动态体系结构的逐层展开。实例说明，该方法能为动态体系结构的设计提供一种增量式的开发方式，适用于连接器重用的目的。

关键词：潜在无限动态结构；组合连接器；动态角色；Wright

Specification for Potentially Unbounded Dynamic Structure Using Composite Connector

YU Li-juan[1], XIONG Hui-min[2,3], YING Shi[3]
(1. School of Automation, Wuhan University of Technology, Wuhan 430070; 2. Department of Mathematics & Statistics, Huazhong Normal University, Wuhan 430079; 3. State Key Laboratory of Software Engineering/Computer School, Wuhan University, Wuhan 430072)

【Abstract】 A critical issue for complex software architecture design is modeling and analysis of dynamic architecture. This paper argues for an approach to specification of potentially unbounded dynamic structure based on Wright, an architecture description language. To support recursion mechanisms in software architecture specification, it identifies the concepts of composite connector and dynamic role, and illustrates how it can be used to create dynamic extension of structure. It provides mechanisms for designing dynamic architecture in an incremental way, and is available for architectural connectors reuse.

【Key words】 potentially unbounded dynamic structure; composite connector; dynamic role; Wright

动态软件体系结构是当前软件体系结构最重要的研究方向之一[1]，其中一个重要问题是关于潜在无限动态结构的描述。所谓潜在无限动态结构，指的是在动态软件体系结构的结构演化中，系统的拓扑结构能够按某种模式无限地扩展[2]。例如，在一些复杂的大型计算系统中，为了节省资源和提高运算速度，往往要求系统根据实时计算的需要可伸缩地产生相应规模的体系结构。

在过去的研究中，笔者面向体系结构规范的重用提出了基于反射机制的连接器组合方法。这种方法引入组合元操作扩展了Wright[3]体系结构描述语言，获得了组合连接器的概念[4]。本文试图基于这一概念提出规范潜在无限动态结构的形式化方法。

1 组合连接器与动态角色

1.1 组合连接器与组合元操作

按照反射的观点，把一个组合连接器规范分为基级和元级。基级用Wright描述各个简单的连接器，元级则描述简单连接器的组合。为了实现连接器组合这种元计算，元级中需要引入组合元操作。

由于Wright基于角色和粘结来规范连接器，并把它们都看作进程，而组合连接器的每个角色以及角色间的粘结都是由简单连接器经过组合元操作得到的，因此连接器组合的元操作实质上也可以看作是进程间的运算。

本文借用CSP进程间的运算[5]，构造了一组连接器组合的元操作，见表1。

表1 组合元操作及其语义解释

类型	组合元操作	语义解释
关于角色的元操作	Substitute	一个角色替代另一个角色
	DecisionMerge	一个角色不确定选择充当多个角色
	ChoiceMerge	一个角色选择充当多个角色
	FollowMerge	一个角色依序充当多个角色
	InterruptMerge	一个角色顺序中断充当多个角色
关于粘结的元操作	Decision	不确定选择多个粘结协议
	Choice	选择多个粘结协议
	Follow	顺序执行多个粘结协议
	Interrupt	顺序外中断执行多个粘结协议
	IntroInterrupt	顺序内中断执行多个粘结协议

1.2 动态角色

为了便于规范动态体系结构，允许在连接器的组合描述中将其角色声明为动态特性，从而提出动态角色的概念。

由于软件体系结构的动态性可直接表现为连接器交互协议的更替，因此动态角色的动态性指的是，某运行时刻它与组件端口进行绑定与否，取决于此时的交互协议是否需要该角色参与。若是，则动态角色就要被相应的组件端口实例化；若不是，则动态角色就不被相应的组件端口实例化，如果已经实例化，那么该实例化也要被取消。

基金项目：国家自然科学基金资助项目(60473066)；湖北省青年杰出人才基金资助项目(2003ABB004)
作者简介：虞莉娟(1975－)，女，讲师，主研方向：自动控制理论及应用；熊惠民，讲师；应 时，教授
收稿日期：2006-12-10 **E-mail：**xiong_hm@tom.com

2 潜在无限动态结构的规范方法

由于潜在无限结构内部包含了同样的一种结构，因此为了描述它必须采用递归机制。众所周知，在 Wright 中，一个体系结构也可以抽象为一个层次化组件。因此，如果基于 Wright 的组合机制层次构造一个组件，使得它的子结构中还包含同一类型的组件实例，那么一个体系结构就能与自身直接连接，从而具备扩展自身的潜力。

为了表达动态性，还需要在层次化组件中包含一个动态组合连接器。如前所述，组合连接器的动态角色只有在交互协议需要它时才被预先定义好的相应的组件端口实例化。这样，如果把组合连接器的动态角色配属到同一类型层次化组件中的某个端口上，那么角色实例化的过程就可以不断交替下去，从而形成一个潜在无限的动态结构。

上述规范方法实现了动态体系结构的逐层展开：在系统演化的初始阶段，它只有一层结构。随着系统的运行，组件交互协议可能会需要该层动态角色的参与，这样与之相配属的组件就会被实例化，即该组件对应的子结构就要被“ 展开 ”，从而系统就获得了两层结构。在系统的第 2 层结构，如果需要与之相连的子结构也要被 “ 展开 ”，同样也含有动态角色。如此下去，随着子结构一层一层地展开，系统就获得了一个潜在无限动态结构。

3 一个潜在无限动态结构的例子

下面通过一个经典的例子[1]来说明笔者的方法。考虑这样一个问题：求前n个奇素数。可以采用经典的Eratosthenes筛法来完成搜寻奇素数的任务，从而得到如图 1 所示的软件体系结构。

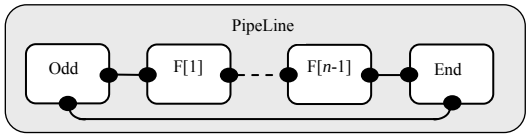

图 1 Eratosthenes 素数筛法的软件体系结构描述

在图 1 中，组件 Odd 不断地产生奇数，并将这些奇数送给后续的组件 F[1]来处理；F[1]将它得到的第 1 个奇数赋给它的一个局部变量 first，并打印其值，从而得到第 1 个素数，然后再将后续得到的奇数逐个与 first 进行比较，若两者不能整除，就将该奇数继续往后传递；组件 F[2], F[3], …, F[n-1]按照 F[1]相同的方式来处理，而 End 则在将它得到的第 1 个奇数作为素数打印出来后，就通知组件 Odd 不再继续产生奇数，从而终止整个计算过程。

在上述计算中，F[1], F[2], …, F[n-1]是相同类型的组件，该组件类型用体系结构描述语言 Wright 进行规范。

```
component Prime=
port input =read→input →read-eof →close → §
port output = write→ output Π close → §
computation = let OnlyPass= input.read→ compare →
(yes→OnlyPass →no→output.write→OnlyPass)
→input.read-eof→input.close→output.close→§
in input.read →setfirst→print→OnlyPass
→input.read-eof→input.close→output.close→§
```

F[1], F[2], …, F[n-1]之间的连接器以及 F[1]与 Odd 之间，F[n-1]与 End 之间的连接器都是管道过滤器类型，该类型的 Wright 规范如下：

```
connector Pipe=
role writer = write → writer Π close → §
role reader = let ExitOnly = close → §
in let DoRead = read → reader
→read-eof→ ExitOnly
in DoRead Π ExitOnly
glue = let ReadOnly = reader.read → ReadOnly
                →reader.read-eof→reader.close→§
→reader.close→§
in let WriteOnly = writer.write → WriteOnly
→writer.close →§
in writer.write → glue →reader.read → glue
→writer.close → ReadOnly
→reader.close → WriteOnly
```

其他组件和连接器的形式规范这里从略。

显然，在这一问题中，问题规模 n 决定了 Prime 类型组件的个数(为 n-1)，从而该软件系统的体系结构规模在设计时可以完全确定下来，即它是静态的。

考虑另一个问题：求 1-n 之间的所有奇素数。显然，可以采用前述的算法，从而可以重用以前获得的组件和连接器。但与前述问题不同，Prime 类型的组件个数在设计时不能确定，它要随计算的不断进行而逐渐地添加上去，即这一问题的软件体系结构是动态的，属于潜在无限结构的类型。

按照组合连接器的观点，该系统中应存在着两种类型的连接器，它们分别用来协调系统某次扩展前后相关组件间的交互行为。显然，扩展之后的连接器类型就是已定义的连接器类型 Pipe，而扩展之前的连接器类型还需要定义，其规范如下：

```
connector Single=
role writer = close → §
glue = writer.close → §
```

连接器 Single 只有一个角色，粘结对它并不产生实质性的约束。另外，与连接器 Pipe 的 writer 角色规范相比，Single 的角色还缺少一个选择分支，该分支描述了 writer 角色执行 write 事件后的行为。显然，write 事件只在系统需要扩展时才会发生，因此它不属于 Single 连接器描述的事件。

一个动态组合连接器类型 Dpipe 定义如下：

```
connector   Dpipe=
composite:   c1: Single
             c2: Pipe
role writer = Decision(c1.writer, c2.writer)
dyn role reader = Substitute(c1.reader)
glue = IntroInterrupt (c1.glue, writer.write, c2.glue)
```

该形式规范首先通过 composite 描述说明 DPipe 是一个组合连接器，并且它是通过一个 Single 类型的连接器和一个 Pipe 类型的连接器组合得到的。

该形式规范接着描述了组合连接器包含的 2 个角色，它说明了 writer 角色是两子角色 c1.writer 与 c2.writer 的不确定性选择合一，reader 角色则是子角色 c1.reader 的替换。特别是，reader 角色规范为动态角色，这与笔者的想法相符：reader 角色只有在体系结构扩展的时候才需要被组件端口实例化。

该形式规范最后描述了组合连接器的粘结。它通过函数 IntroInterrupt 将两个子粘结进程 cs1.glue 与 cs2.glue 顺序中断地组合起来，并实现了两种通信协议的动态过渡。由于这里的中断事件是组合连接器的角色事件 writer.write，因此它使用组合元操作 IntroInterrupt。

有了上述动态组合连接器以后，就可以借助递归机制规范潜在无限动态结构。

（下转第 56 页）