第26卷 第12期 计 算 机 工 程 2000年12月
Vol.26 № 12 Computer Engineering December 2000
• 网络与通信 • 文章编号：1000—3428(2000)12—0139—03 文献标识码：A 中图分类号： TN911

C2构架风格在通信软件中的应用

潘明耀，夏宽理，汪 洋，林 娟
（复旦大学计算机系，200433； 上海贝尔公司，201206）

摘 要：该文以C2构架风格在电话交换机呼叫控制模块中的应用为例，提出了用该风格实现通信软件动态演化的一种方法。

关键词：构架；构件；连接器；包装器；模块

C2 Architecture Style Applied in Communication Software

PAN Mingyao, XIA Kuanli, WANG Yang, LIN Juan
(Computer Department,Fudan University，Shanghai 200433; Shanghai Bell Company，Shanghai 201206)

【Abstract】In this thesis we provide a method by giving an example of applying the C2 architecture style to the call control module in the telephone exchange.

【Key words】Architecture; Component; Connector; Wrapper; Module

1 问题的提出

伴随着新业务的出台，通信软件也在不断地进行升级。以目前我国广泛采用的S1240程控电话交换机为例，目前每隔1年左右，就需要进行1次升版。由于原来没有实现动态演化，只能进行离线升版。每一次升版都需要系统再启动，需停机约1小时左右，造成通信中断，给用户带来极大不便。同时，每次离线升版，电信部门都需要临时调整电路，厂家需要重做软件，对新软件程序和数据需要大量繁重的校验工作。当前我国约有近3000台S1240程控交换机在运行，一次全国范围大规模的升版工作都要给电信部门和提供商带来约500人年的工作量。因此，解决程控交换机的动态演化问题，实现在线升版，从而实现升版不中断通信，节约人力资源，已经迫在眉睫。

消息1、消息2：模块收到消息；
消息3、消息4、消息5：模块发出消息。

图1 软件模块可视作黑盒

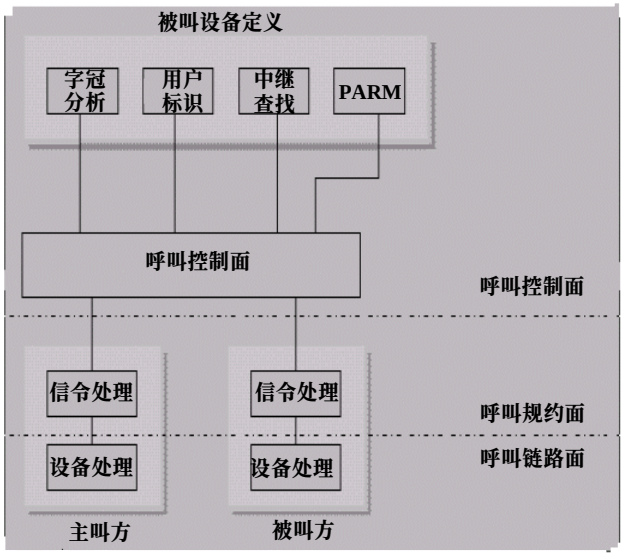

图2 呼叫处理相关的模块

像S1240这样的全分散程控电话交换机，其硬件和软件都是模块化的。系统的软件模块大部分是以有限状态机的形式，从外部可以将软件模块看作一个黑盒。模块间以异步消息进行通信，每一个模块能接受有限种类的消息进行处理，并能发出有限种消息，见图1。为区分以下所提到的构件间传递的消息，我们将软件模块间通信用的消息称为系统消息。

我们在实际工作中发现新业务增加和修改都是在呼叫控制层面上进行的（见图2）。改动最频繁的是系统中有关呼叫控制的CC（Call Control)模块。因此我们首先着手对该模块进行再工程，重新进行设计，使之能满足动态演化的需要，减少离线升版的次数。

2 C2构架风格简介

2.1 C2构架风格结构

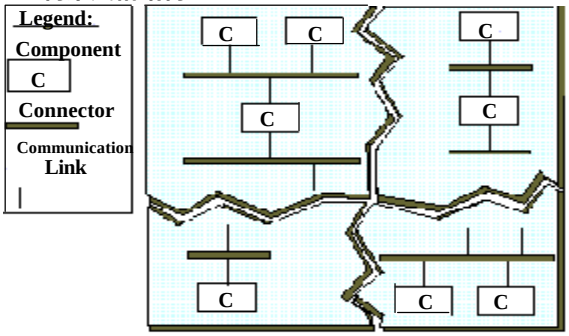

图3 C2构架风格

C2是一种基于分层结构，事件驱动的软件构架风格。图3是一个典型的C2风格构架。C2构架中的基本元素是构件 (Component)，连接器(Connector)。每个构件定义有一个顶端接口和一个底端接口，通过这两个接口连接到构架中，这使得构架中构件的增加、删除、重组更为简单方便。每个连接器也定义有顶端接口和底端接口，但接口的数量与连接

作者简介：潘明耀(1967～)，男，高级工程师，在职研究生，主要研究方向为软件工程；夏宽理，教授；汪 洋，研究生；林 娟，工程师

收稿日期：2000-06-08

在其上的构件和连接器的数量有关，这也有利于实现在运行时的动态绑定。构件之间不存在直接的通信手段。构架中各元素（构件，连接器）之间的通信只有通过连接器传递消息来实现。如图4。处于底层的构件向高层的构件发出服务请求消息(Requests)。消息经由连接器送到相应的构件。处理完成后由该构件将结果信息(Notifications)经连接器送到低层相应的构件。

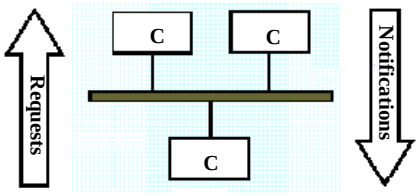

图4 构件间的服务请求

2.2 C2构架风格的特点

在C2风格中，连接器以独立实体存在(在这一点上许多构架风格都不具备)。由连接器将各个构件连接起来并充当它们交互的中间件，从而将构件的接口需求与功能需求相分离。这样在构架的演化中对演化的处理策略，对演化部分的隔离，及对演化的一致性维护都可以通过连接器来实现的。

C2风格区别于其他软件风格的几个显著特点如下：(1)基底独立性(Substrate independence) 在整个构架中，构件只知道处于它上层的构件而不知道它下层的构件。构件通过发送一个请求消息利用上层的构件的服务，而构件与下层构件的通信是隐式的。它发出的消息有可能被多个构件所接受；(2)消息通信机制(Message-based communication) 构件之间的交互只有通过信息传递实现；(3)多线程(Multi-threaded) 组成系统的各个构件有自己的状态，进程控制；(4)不共享地址空间(No assumption of shared address space) 组成系统中的各个构件不共享地址空间；(5)实现与构架分离(Implementation separate from architecture) 实现技术与系统的概念构架相分离。

C2风格的以上这些特点减少了构件之间的相互依赖性，有利于实现系统构架的变化，对系统演化及动态演化提供了很大程度的支持。

2.3 C2 构架演化支持框架

```
Component Interface          Architecture Interface
start()                      start()
finish()                     finish()
handle(request)              handle(request)
handle(notification)         handle(notification)
                             addComponent(component)
Connector Interface          removeComponent(component)
start()                      isExistingComponent(component)
finish()                     enumComponents()
handle(request)              addConnector(connector)
handle(notification)         removeConnector(connector)
addTopPort()                 isExistingConnector(connector)
removeTopPort()              enumConnectors()
addBottomPort()              weld(connector,component)
removeBottomPort()           weld(component,connector)
                             weld(connector,connector)
                             unweld(connector,component)
                             unweld(component,connector)
                             unweld(connector,connector)
                             isWelded(entity,entity)
                             entitiesBelow(entity)
                             entitiesAbove(entity)
                             enumWelds()
```

图5 C2构架的基本类框架

Component，Connector，Architecture是3个基本类。其中方法start和finish是用来初始化一个构件的运行和结束一个构件的运行。handle方法用来处理各对象之间的消息传递。addTopPort，addBottomPort，weld用来在运行时改变构架的组成元素和各元素之间的连接拓扑结构。enumComponent等方法用来查询构架自身组织和结构。C2框架的这些方法有利于描述和实现系统构架的动态变化。开发者可以通过对上述对象的继承从而复用这些方法。

3 C2构架风格在呼叫控制模块中的实现

3.1 呼叫控制模块的基本构架

用C2构架风格设计的呼叫控制模块的构架如图6所示。

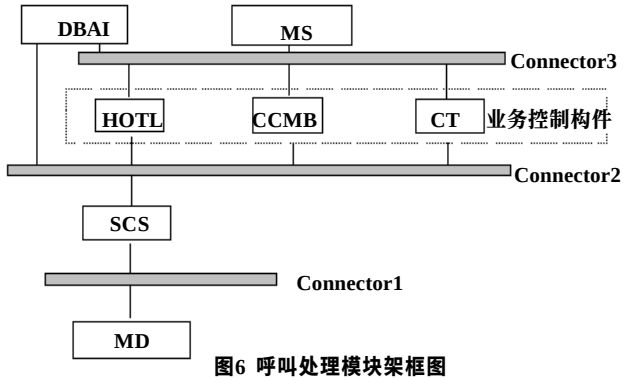

图6 呼叫处理模块架框图

3.1.1 包装器

我们对呼叫控制模块再工程设计后的架构如图5所示。由于该部分软件在整个系统中只是一个模块，为了使重新设计后的呼叫处理模块对系统不产生影响，我们用包装器对之进行了包装。

在本文第1部分已经提到程控交换机软件之间的通信是靠系统消息传递来实现的。基于这个特点，包装器被分为上、下两个部分。下部分为系统消息接受构件MR，它专门负责接受外部模块发送来的系统消息，进行拆包后，将主叫号码、被叫号码及业务类型等信息经连结器1发送给业务构件选择构件SCS。包装器的上半部是系统消息发送构件MS。它执行的是将业务控制构件发送来的消息，经过包装后，以系统消息的形式发往其它模块。

通过系统消息接受构件和系统消息发送构件，我们将呼叫处理模块呼叫控制模块包装起来，使其和原有系统能实现完全兼容。

3.1.2 业务构件选择SCS

业务构件选择构件SCS接收系统消息接受构件所传递来的消息，根据其中的业务类型信息，将消息传递给相应的业务控制构件，调度它来完成相应的呼叫控制工作。

业务类型	业务控制构件
热线业务	HOTL
遇忙回叫	CCMB
叫醒业务	WAKE
…	…
呼叫转移	CT

图7 业务调度表

为了完成业务调度，该构件维护着一张表，称为业务调度表（见图7）。该表存储着各业务类型和与其相对应的业务控制构件的标识。业务调度构件根据业务类型，查询该表后，得出相应的业务控制构件标识，并通过连结器2将请求发往该业务控制构件。

3.1.3 业务控制构件

业务控制构件是呼叫处理模块的核心部分，这些构件负

责呼叫处理过程的逻辑控制，包括各新业务的逻辑控制。它们通过请求系统消息发送构件，向系统内部电路控制、信令控制等其它模块发出系统消息，完成新业务各项功能。

3.1.4 数据库接口构件DBAI

在呼叫控制模块的整个处理过程中，功能调度构件需要查询业务调度表，各业务处理构件要查询和修改如主叫、被叫用户信息等数据。因此，以上构件需要对数据库进行操作，这些操作是通过请求数据库接口构件DBAI完成的，数据库接口构件完成操作后，将结果经连接器2和3回送给请求构件。

3.2 呼叫控制模块演化示例

以下我们通过两个例子来介绍应用C2构架风格后的呼叫控制模块的在线演化过程。

3.2.1 构件的增加

当电信部门需要开放新业务时，我们只需在呼叫控制模块中增加新的构件即可完成。以增加来电显示业务为例，我们需在模块中增加CLIP构件。过程如下:

（1）连接器添加相关端口

```
Connector3.addBottomPort( );
Connector2.addTopPort ( );
```

（2）连结新构件CLIP

```
CC.weld (Connector3,CLIP);     //CC为呼叫处理模块的构架名
CC.weld (CLIP,Connector2);
```

（3）启动新构件

```
CLIP.start();
```

（4）在业务调度表中增加一元组，将来电显示业务与CLIP构件相对应。完成以上操作后，呼叫控制模块便能处理来电显示业务了，此时的构架图如图8所示。

3.2.2 构件的修改

有时在增加新业务时，只需对原有构件进行修改。例如原来已有无条件呼叫转移业务，处理该业务的构件是CT。当要增加遇忙转移业务时，只需修改构件CT，在此将修改后的构件称为CTnew。

（1）连结器添加新端口

```
Connector3.addBottomPort();
C2.addTopPoint();
```

（2）将新构件连结进构架

```
CC.weld (C3,CTnew);
CC.weld(CTnew,C2);
```

（3）启动新构件

```
CTnew.start( );
```

（4）修改业务调度表，将无条件转移与新增的遇忙转移业务对应到新的CTnew构件，从此，所有呼叫转移业务将由CTnew构件处理。

（5）终止旧构件

```
CT.finish( );
```

（6）将旧的构件脱离出构架

```
CC.unweld (Connetor3,CT);
```

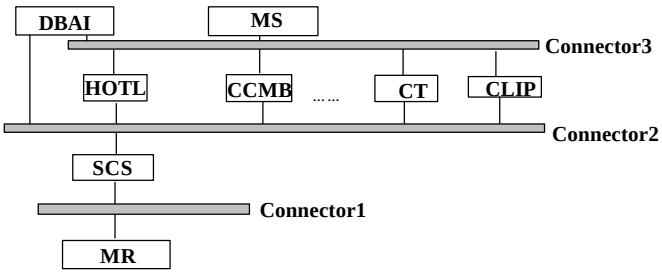

图8 增加CLIP构件后的呼叫处理模块构架图

```
CC.unweld(CT,Connector2);
```

（7）去除无用的端口

```
Connector3.removeBottomPort(i); // i为原连结CT的端口
Connector2.removeTopPort(j); //j为原连结CT的端口
```

经修改后的构架图如图9所示。

业务类型	业务控制构件
热线业务	HOTL
遇忙回叫	CCMB
叫醒业务	WAKE
…	…
呼叫转移	CT
来电显示	CLIP

图9 增加来电显示后的业务调度表

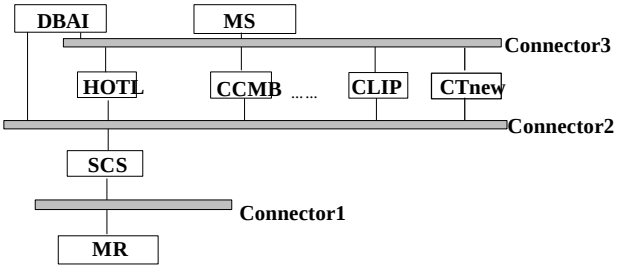

图10 修改CT构件后的呼叫处理模块构架图

业务类型	业务控制构件
热线业务	HOTL
遇忙回叫	CCMB
叫醒业务	WAKE
… …	
呼叫转移	CTnew
来电显示	CLIP

图11 修改呼叫转移业务后的业务调度表

实际实现中以上过程是用一个人机命令程序完成的。

4 进一步的工作

以上工作是我们对呼叫处理模块的再工程工作。这一工作能使呼叫处理模块能实现动态演化，从而减少了交换机的离线升版工作。但由于其余模块还未实现动态演化，因而还不能完全实现离线升版。在今后一段时间内，我们将继续对其它模块进行改造，从而完全实现交换机的在线升版。

5 总结

C2构架风格的灵活性及易实现的特性，在系统软件的动态演化过程中是十分有用的。而大部分的通信系统都要求系统工作不能中断，从而其软件需要进行动态演化。因此，C2构架风格应能在通信软件中得到充分应用。当然，目前大部分的通信软件都不是以C2构架风格实现的。因此，一些再工程工作不可避免。但一旦引入该风格后，将能大大减少今后软件演化的成本，其优越性将会得到充分显示。

参考文献

1 Oreizy P.Architecture-based Runtime Software Evolution. In the Proceedings of the International Conference on Software Engineering，1998

2 Dynamic Structure in Software Architectures. Fourth SIGSOFT Symposium on the Foundations of Software Engineering,San Francisco, 1996

3 Architectural Support for Dynamic Reconfiguration of Distributed Workflow Applications. IEE,1998-10