C2 风格软件构架的演化研究

薛云皎　余枝强　钱乐秋　汪　洋
(复旦大学计算机科学与工程系软件工程实验室,上海 200433)
E-mail:yunjiaoxue@hotmail.com

摘　要　软件演化是软件生命周期中始终存在的变化活动,软件维护只是软件演化的特定阶段的活动和组成部分。由于软件复用思想的兴起,基于构件和构架的软件开发方法得到越来越广泛的应用,而相应的软件演化就成为组成软件系统的构件的演化和构架的演化。该文探讨了软件演化和构架演化的概念,研究了构架动态演化所涉及到的问题,并在分析软件构架风格的基础上重点研究了 C2 构架风格对构架动态演化的支持能力。该文提出的构架动态演化模型及其与 C2 风格的结合能够支持实际的构架演化。

关键词　构件　C2 构架风格　演化　动态演化

文章编号 1002-8331-(2002)11-0083-04　**文献标识码** A　**中图分类号** TP31

Research on Software Evolution Based on C2 Style Architecture

Xue Yunjiao　Yu Zhiqiang　Qian Leqiu　Wang Yang
(Laboratory of Software Engineering,Dept.of Computer Sci. and Tech.,
Fudan University,Shanghai 200433)

Abstract: Software evolution is a transforming activity all along in the lifecycle of software.The software maintenance is just an activity in specific period and a composing part of software evolution.The architecture-based and component-based development method is applied more and more widely because of the rise of software reuse theory,thus the corresponding software evolution turns into the evolution of components and architectures which construct software systems.This paper discusses the concept of software evolution and architecture evolution,then studies some questions on architecture dynamic evolving and emphasis on the supporting ability of C2 architecture style for architecture dynamic evolving based on analysis of software architecture style.The integration of C2 style and architecture dynamic evolving model put forward in the paper can support actual architecture evolution.

Keywords: Component,C2 Architecture Style,Evolution,Dynamic Evolving

软件维护和软件演化是在系统开发中经常要面对的两个重要问题。如何以一种良好的风格和技术来设计软件构架以使之在动态演化中保持软件系统的稳定性、一致性,是软件工程学界长期以来致力研究的一个问题。

软件维护和软件演化是两个相互联系又具有本质区别的概念。软件维护是对现有的已交付的软件系统进行修改,使得目标系统能够完成新的功能,或是在新的环境下完成同样的功能,主要是指在软件维护期的修改活动。而软件演化则是着眼于软件的整个生命周期,从系统功能行为的角度来观察系统的变化,这种变化是软件的一种向前的发展过程,主要体现在软件功能的不断完善。在软件维护期,通过具体的维护活动可以使系统不断向前演化。因此,软件维护和软件演化可以归结为这样一种关系:前者是后者特定阶段的活动,并且前者直接是后者的组成部分。二者之间关系可用图 1 表示。

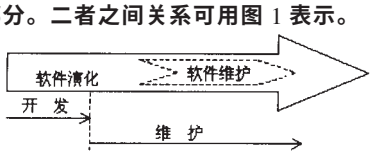

图 1　软件维护和软件演化

1　构架演化

在硬件工业标准化、专业化和集成化成功经验的基础上,基于软件构件、构架的开发方法已经逐步成为有效提高软件工业生产率的手段和人们研究的重点。其主要概念如下:

(1)构件(Component)

构件是指用以构筑软件系统的可以复用的软件元素。构件通常表现为不同的形态,形态的差异体现在结构的组织方式和所依赖的方法学范型上。

(2)软件构架(Software Architecture)

软件构架是一个系统视图,该视图描述以下内容:系统的主要构件;构件相对系统其它部分的可见行为,以及为了达到系统预定的功能构件之间所采取的交互和协作关系。

早期的基于软件构架的研究都主要集中在描述系统的静态表现形式上,系统的构架被认为是比较稳定的,在系统的整个生命周期中不易发生变化。系统的演化主要集中在构件的演化上。但是,随着社会竞争的不断激化,新需求要求不断改进软件,为了适应以上的变化特别是一些比较大的变化,在系统进行演化的过程中有可能要改变系统的软件构架。

作者简介:薛云皎,硕士研究生,研究方向:软件工程,构件库管理系统,基于构件、构架的软件开发方法。余枝强,硕士研究生,研究方向:软件工程,构件库管理系统,基于构件、构架的软件开发方法。钱乐秋,教授,博士生导师,研究方向:软件工程,构件库管理系统,构件生产与组装,软件测试,基于构件、构架的软件开发方法。汪洋,硕士研究生,研究方向:软件工程,基于构件、构架的软件开发方法。

如果从构架演化的时间来分,构架演化可以分为以下四个阶段。

(1)设计时构架的演化:在设计时,随着对系统的理解不断深入,系统的构架越来越清晰、完善,这本身就是一个构架的变化过程。在这个阶段,由于构架模型没有应用代码与之相对应,这时候的演化是相对简单和易于理解的,因为此时的演化仅仅是系统的一个抽象模型被改变了。

(2)运行前构架的演化(静态的构架演化):构架中各部分对应的代码已经编译到系统中,但系统没有开始运行。由于系统没有运行,这个阶段的构架演化不需要考虑到系统的状态信息。这时候构架的演化也比较简单,一般表现在重新编译构架中变化部分的代码或对软件进行配置。这种演化在处于开发过程的系统中随处可见。

(3)安全模式运行时构架的演化:又称受限运行演化。系统运行在安全模式下,即系统的运行受到一定条件的限制。在该状态下系统构架的演化不会破坏系统的稳定性和一致性,但是演化的程度是受到限制的。系统必须提供保存当前系统的构架信息和动态演化的机制。构件动态地增加和删除是这种演化的一个例子。

(4)运行时的构架的演化:构件的演化在系统的运行过程中进行,在构架的演化过程中,要检查系统的状态,包括系统全局的状态和演化部分构件的内部状态,以保证系统的完整性和约束性不被破坏,演化后的系统能够正常运行。相对前面几种情况来说它是最复杂的,除了要求系统提供保存当前系统的构架信息和动态演化的机制外还要求有演化一致性检查的功能。

在构架演化的研究领域中的一个重要分支为前述提到的第四种情况:系统构架的动态演化。下面将结合构架风格中的C2风格做详细介绍。

2 C2风格

构架风格是指能够标识一类构架组织特征的模式,它定义了描述构架系统的词汇和组织构架系统的规则。一种构架风格代表了一种软件设计元素进行组织的特定模式。不同风格的构架对构架演化的支持度也是不同的,在下面将要讨论的C2风格由于它独特的消息连接机制和独立的连接器实体形式给构架的演化提供了有利的条件。

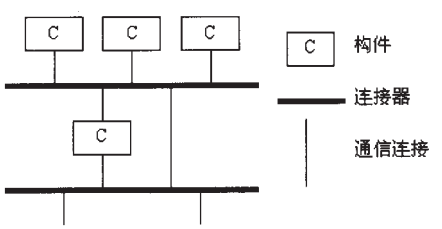

图2 C2构架风格

C2是一种基于分层结构,事件驱动的软件构架风格。图2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。构件的顶端接口用于发出请求、接收结果,底端接口则用于接收请求、发出结果。在这种结构下,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讯手段。构架中各元素(构件,连接器)之间的通讯只有通过连接器传递消息来实现。如图3,处于低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。

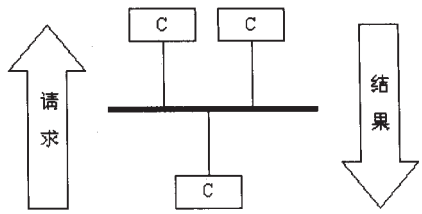

图3 构件间的服务请求

在C2风格中,连接器以独立实体的形式存在(许多构架风格都不具备这一特性)。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、演化部分的隔离及演化的一致性维护都可以通过连接器(Connector)来实现。

C2风格区别于其它构架风格的几个显著特点如下。

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讯是隐式的。它发出的消息有可能被多个构件所接收。

消息通讯机制(Message-based communication)

构件之间的交互只能通过消息传递实现。

多线程(Multi-threaded)

组成系统的各个构件有自己的状态,进程控制。

不共享地址空间(No assumption of shared address space)

组成系统的各个构件不共享地址空间。

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,对系统的动态演化提供了很大程度的支持。

3 C2风格与构架动态演化

正如前面提到的构件的动态演化一样,构架的动态演化同样是系统动态演化的要求。一些运行系统如航空控制系统和现在流行的基于Internet的关键业务的系统,要求系统24*7*365连续运行,每一次关闭系统都将带来巨大的损失。而这些系统面对不断出现的新需求和变化,也需要进行系统的演化。这种演化不仅会要求某个构件进行改进,而且有可能要求系统的构架发生变化。

3.1 构架动态演化的一些问题

构架动态演化的表现形式和前面提到的构架演化形式相同,但是演化的环境是软件运行的状态,下面是主要的表现形式和实现的功能。

动态构件的增加:在运行期增加新的构件,动态地增加系统的功能。

动态构件的删除:在运行期删除无用、过时、有问题的构件。

动态构件的替换、更新：在运行期用新的构件代替老的构件，动态地改变、完善系统的功能或提高系统的性能。

动态构件间拓扑结构的改变：在运行期改变构件之间的调用关系，动态地改变系统处理逻辑。

因为演化发生在软件的运行状态，所以要实现以上的构架演化并不像在静态时配置那么简单，系统必须提供构架动态演化的其它相关功能。以下是两个最基本功能。

(1)软件构架的动态维护功能

系统如果要支持构架动态演化，首先要求系统必须提供保存系统当前的构架信息的功能，因为构架在运行过程中是变化的，只有对系统当前的构架具有准确的描述，才能控制系统的构架的变化，这一点是其它系统都不具备的。其次要求系统可以支持构架的动态配置操作，也即构架的动态装配。无论是以上构架动态演化的哪种形式，都存在在系统运行过程中将构件连接到构架中或从构架中断开连接等问题。

(2)构件维护管理功能

在系统运行期增加、删除、替换构件，都面临着在系统运行期动态地创建（初始化）、释放构件的问题。在初始化构件时，需要根据系统的运行情况设置构件相应的初始化状态。在释放某个构件时，应保证该构件处于非激活状态。

结合上面的系统对构架的辅助功能，可以将演化过程分解为以下几个基本操作。

(1)创建构件：初始化一个新构件，此时构件并未连接到构架中。

(2)删除构件：释放构件，被释放的构件应该已经脱离构架。

(3)构件的装卸，即装配构件到构架中和从构架中卸下构件，通过构件的装卸来改变构架的拓扑结构。

(4)查询构架中构件的基本信息，如版本号、环境说明。

(5)查询构架的整个拓扑结构，由此可以定位到指定的构件。

以运行时构件的替换为例，其整个具体动态演化操作过程为：

(1)在做任何的构架演化之前都要通过整个构架的结构信息来定位构件。

(2)在替换构件前先查询原有构件的基本信息如版本、环境等，对比新老构件的相应信息，保证新构件的版本确实比老的构件版本新。

(3)增加新构件，断开老构件与构架的关系，将新构件连接到构架中，再删除老的构件。值得注意的是，在从构架上断开老构件时必须先查询老构件的状态。如果老构件处于非激活状态（没有正在处理的事务），那么可以简单地将其从构架中断开。如果老构件正在处理事务，那么又有两种处理方式：一是等待老构件处理完毕以后再从构架中断开，二是先保存老构件处理事务的现场，再中断处理过程，等到新构件连接到构架时恢复处理事务的现场，继续中断的处理过程。相对来说第一种方法处理比较简单，而第二种方法要求建立适当的机制来处理其中的失败操作，如可以象数据库事务一样，将每一步的操作划分为原子操作，如果其中一个原子操作失败了，就回滚整个演化过程。这样做的目的是避免让系统处于一个不稳定的状态。

基于构架的动态装卸是构架动态演化的核心问题，为了更好地实现动态装卸和避免装卸中的一些问题。这里对组成构架的构件和连接器也提出了新的要求。

构件是实现系统功能的实体。一般把构件看成一个黑盒子，它可以有自己的状态、信息和控制线程，并可以用任何语言实现。把构件看成黑盒子有利于 OTS（Off-The-Shelf）构件的重用。然而并不是所有的 OTS 构件都支持构架的动态演化。为了支持构架动态演化，构件不能够通过直接引用来进行通讯。构件之间的通讯应通过连接器来进行，从而降低了构件之间的耦合度，使得构件的动态装配到构架中变为动态地装配到相应的连接器中。构件必须提供参与构架动态演化的基本功能：一是构件应提供在运行环境下被动态地载入和联接的功能。如今流行的 CORBA、COM、JAVA 构件都具备该特性。二是构件应提供和连接器动态绑定的功能。这些必需的功能对各个构件都是相同的，一般可以设计一个通用的包装器或代理构件来进行复用。

在构件动态演化中，连接器必须作为一个独立的实体存在。只有将连接器独立出来，才能将各个构件真正隔离开。在有些构架风格中，连接器只在设计阶段以独立的实体存在，如在 Unicon 中，实现阶段连接器被替换为进程调用和数据共享。相应的构件的绑定在系统的实现阶段也固定下来了，结果在系统运行时要改变绑定关系非常困难。另一方面，象构件一样，连接器也要有相关的功能来支持动态的演化：动态的载入和绑定。在构架的动态演化中，连接器的作用主要表现在：

(1)连接器能够实现不同的演化策略。例如：如果要支持瞬间的构件替换，连接器可以将通讯从老构件切换到新构件。如果要实现渐进式的构件替换，就可以将老的服务请求定位到老构件，新的服务请求定位到新的构件。

(2)连接器可以隔离演化部分，防止演化的影响扩散，将演化局部化。例如：如果一个构件在系统运行时不可用了。那么连接器将发给它的服务请求用临时队列保存起来直到该构件恢复正常工作。结果达到了其它构件与该变化相隔离的效果。

构架动态演化的实施过程：一般情况下，构架动态演化是和运行系统绑定在一起的，即由运行系统自己实现演化过程。对于一些特殊的软件系统，如果它们的实现相对简单独立，可以用一个外部的演化管理环境来实现。整个动态演化过程先根据系统具体的演化要求，得到演化基于构架级的表示，例如是构件的增加还是构件替换等等。接下来的工作是由系统本身或外部环境中的构架动态演化支持部分完成。首先是定位到相关的构架元素，如新老构件或连接器。接下来就是构架的重组，构件的装配。最后是演化过程中一致性的检查和控制变化部分的运行。

动态演化的理想模型：该模型描述了从系统接收外部变化开始到最终系统实现变化的整个过程，如图 4。

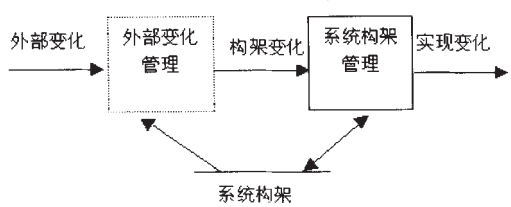

图 4　构架动态演化的模型

系统构架以脚本的形式存在。通过对外部变化进行解释，将其转化为系统构架的变化（如：构件的增、删、替换等）。这部分由系统设计师根据系统的需求情况得到。再由系统构架管理器将其转化到相应的实现变化。系统构架管理器还负责对提出的构架变化进行一致性验证及对构架脚本进行更新维护。

3.2 C2 风格在构架动态演化中的应用

在基于构架的动态演化的研究中,笔者在考察了许多构架风格的基础上选择了 C2 风格。C2 风格非常满足前面讨论的构架动态演化的相关要求。

(1)在 C2 风格中,构件不能直接引用,所有的构件是通过连接器来相互通讯的,这减少了构件之间的依赖性,分离了构件的计算部分和通讯部分。

(2)C2 风格中的拓扑约束有利于构架的动态变化。在 C2 风格中,每个构件有一个顶部接口和一个底部接口,这极大地简化了构件的增加、删除、替换、重新组合。

(3)C2 风格中的连接器以独立的实体存在,构件之间的通讯通过它以消息的传递来实现。

(4)C2 风格中的连接器也有顶部接口和底部接口,但接口数量是不确定的,由连接在它上面的构件和连接器的数量决定,能够实现运行时的动态绑定。

此外,C2 风格还提供了一套 C2 风格应用程序的实现类框架,如图 5。这个类框架中提供了对应 C2 风格中构件(component),连接器(connector),构架(architecture)的三个抽象类,并实现了构件-连接器,连接器-连接器互连的基本方法以及消息的发送、接收和传递机制。应用系统中的构件和连接器可以通过继承、重载上面的类进行复用和做相应的改变。其中方法 Component 的 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 框架的这些方法可以描述和实现系统构架的动态变化。

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

例如前面分析的动态演化操作可以通过类框架中所述的方法来实现:创建构件——start()、addComponent(component);删除构件——removeComponent(component);释放构件——finish();增加连接器——addConnector(connector);删除连接器——removeConnector(connector);构件的装卸,改变构架的拓扑结构——weld(connector,component),weld(component,connector)等。

在实验室的研究中,笔者以 C2 风格的演化框架为基础,利用 Java 的动态载入技术,实现了一个 C2 构架的组装和演化图形环境,该环境可以实现 C2 构架的文本描述和图形表示,并通过该描述来组装系统,在系统的运行过程中可以通过增加/删除构件、增加/删除连接器、改变构件与连接器的连接关系来实现系统运行期的演化。尽管只是一个示例性的系统,也已经表明 C2 风格的构架对软件系统动态演化的良好适应性和支持能力。

4 结束语

C2 构架风格的特性使之能够满足软件动态演化诸多方面的要求,其核心优势在于消息传递通讯机制和独立的连接器机制,这使得构架演化的实际应用也较易实现。目前在实践上的缺陷在于目标系统与构架演化环境紧密结合,笔者希望在进一步的研究中将演化的支持环境独立出来,对于任何一个正在运行的构架在侧面进行控制,同时加入对多种构件标准的支持,这样,构架的动态演化理论将具有更为广阔的应用前景。

(收稿日期:2001 年 5 月)

参考文献

1.Don Batory,Lou Coglianese,Mark Goodwin et al.Creating Reference Architectures:An Example from Avionics.ACM 0-89791-739-1/95/0004,1995

2.Chris F,Kemerer,Sandra Slaughter.An Empirical Approach to Studying Software Evolution[C].IEEE transactions on software engineering,1999,25(4)

3.Peyman Oreizy et al.Architecture-based Runtime Software Evolution[C].In:Proceedings of the International Conference on Software Engineering,1998

4.Peyman Oreizy.Decentralized Software Evolution[C].In:Proceedings of the In-ternational Conference on the Principles of Software Evolution(WPSE 1),Kyoto,Japan,1998

5.M shaw et al.Abstractions for software architecture and tools to support them[J].IEEE Transactions on Software Engineering,1995-04

6.齐治昌,谭庆平,宁洪.软件工程[M].北京:高等教育出版社,1997

(上接 74 页)

3.Ruts I,Rousseeuw P.Computing depth contours of bivariate point clouds[J].Computational Statistics and Data Analysis,1996,23:153~168

4.Ramaswamy S,Mahayana S,Silberschatz A.On the discovery of Interesting patterns in association rules[C].In:Proc of the 24th VLDB conference,New York,USA,1998:368~379

5.王丽珍.一种基于语义贴近度的抽象归纳法[J].计算机学报,2000,2(0):1114~1121

6.Ng R,Han J.Efficient and effective clustering methods for spatial data mining[C].In:Proc 20th VLDB,1994:144~155

7.Ester M,Kriegel H P,Sander J et al.A density-based algorithm for discovering clusters in large spatial databases with noise[C].In:Proc KDD,1996:226~231

8.Zhang T,Ramakrishnan R,Livny M.BIRCH:An efficient data clustering method for very large databases[C].In:Proc ACM SIGMOD,1996:103~114

9.Knorr M,Ng R.Finding intentional knowledge of distance-based outliers[C].In:Proceedings of the 25th VLDB conference,Edinburgh,Scottand,1999:211~221

10.陈世权,郭嗣琼.模糊预测[M].贵阳:贵州科技出版社,1994