计算机科学2002Vol. 29№. 11

软件体系结构:一个新的研究领域*)

Software Architecture:a New Research Domain

赵会群　孙　晶　王国仁　高　远

(北方工业大学计算机系　北京100041)　(东北大学计算机科学与工程系　沈阳110006)

Abstract　As a new research domain, software architecture is attracted. However the study on software architecture is still in beginning state, and there are many questions about the basic theory for subject construction and method for engineering problem solving need answer. This paper introduces result on the basic conception, the basic theory and the applying method that have archived, explores some problems which need should be solved and new method that can be used for problem solving.

Keywords　Software architecture, ADL, Software paradigm, Software case

计算机软件系统中软件成分越来越复杂,系统规模不断扩大,使得软件体系结构越来越庞杂,软件系统的质量和性能已经不再仅仅取决于软件实现算法和数据结构,软件系统体系结构在一定程度上决定系统的优劣,因此软件体系结构(Software Architecture, AS)研究已经逐渐地引起计算机界的重视。软件体系结构作为一个新兴的计算机学科,它的理论体系和解决问题的方法尚未形成。加强其基础理论和应用方法的研究,不论对学科发展,还是对软件生产都具有理论意义和现实意义。

本文综述软件体系结构的基本概念和研究现状,明确软件体系结构研究中需要进一步解决的问题,对解决问题的途径和方法提出可借鉴的思路和做法。

1　软件体系结构概述

本节从基本概念、典型结构风格等方面概述软件体系结构。

1.1　软件体系结构概念

由于软件体系结构(有时简称"体系结构"或"结构")是一个新兴的计算机学科,其理论基础尚未形成,因此公认的定义尚未出现。下面列举出一些有关软件体系结构的描述性定义。

S. Vestal 对四种软件体系结构描述语言(Software Architecture Language, ADL)分析后提出[1]:软件体系结构只关心软件系统的高层结构,而不是各模块的实现细节。

Robert Allen 和 David Garlan 从系统组件联系角度指出[2]:所谓的软件体系结构是问题解决方案的逻辑框架。软件体系结构设计包括系统结构的总体设计和控制、各计算单元功能分配、各单元间的高层交互等。

Mary Shaw 和 David Garlan 从软件工程角度指出[3]:软件体系结构是对构成系统各组件、组件交互、指导系统装配范例和范例约束的描述。这里的组件可以是客户/服务器中的客户、服务器,也可以是数据库,还可以是系统结构中的一个层面。从设计角度,组件间交互有时可能很简单,如只包括过程调用和共享变量访问;有时相当复杂,如客户/服务器结构中的各种交互协议、数据库访问协议、异步事件多波通信和管道流。

上述描述性定义从不同侧面对软件体系结构进行描述,但未对软件系统结构的属性和行为给出形式化描述。N. Medvidovic 和 R. N. Taylor 通过对不同的软件体系结构描述语言分析和对比,提出了一个带有普遍意义的软件体系结构框架以及框架中属性和特征的含义描述,框架如下[4]:

Architecture Modeling Features

Components/Connectors/ * 组件或连接器 * /

Interface/ * 组件或连接器接口 * /

Types/ * 组件或连接器的类型定义,它是可重用功能的抽象 * /

Semantics/ * 组件或连接器语义说明 * /

Constraints/ * 组件或连接器行为约束 * /

Evolution/ * 组件或连接器可修改的特性 * /

Non_functional properties/ * 组件或连接器除约束以外说明,如安全性。* /

Architectural configurations/ * 结构说明 * /

Understandability/ * 可理解性,即无需了解细节,就能理解系统特性。* /

Compositionality/ * 可装配性,即可从不同的层面上描述软件的结构。* /

Refinement and traceability/ * 精练和可处理性,描述更接近执行系统。* /

Heterogeneity/ * 可异构性,即支持对不同语言和平台组件的连接。* /

Scalability/ * 可升级,即支持不断更新的需求。* /

Evolution/ * 可进化,即支持对系统功能的添加、删除和替换。* /

Dynamism/ * 动态,即支持对系统功能的动态变化。* /

Constraints/ * 约束,即对结构的限制,它通常依赖局部约束。* /

Non_fuctional properties/ * 非功能说明,含义同组件。* /

N. Medvidovic 和 R. N. Taylor 的定义把软件体系结构看成由组件和连接器构成的集合,组件和连接器按配置说明可以构造一个动态的、可进化的、可升级的系统。

从上述定义中不难看出,软件体系结构是软件系统逻辑框架,是对软件系统的说明和描述。这种描述可分为三个层次[3]:

体系结构级(Architecture):是对构成系统各模块的设计和系统装配说明。系统组件是系统模块,它们可以通过多种途

*)基金项目:国家八六三高技术研究发展计划项目(项目号:863-511-946-003)资助。赵会群　博士,副教授,研究方向:软件体系结构。孙　晶　硕士,讲师,研究方向:软件工程。王国仁　教授,研究方向:分布对象技术。高　远　教授,博士生导师,研究方向:软件容错。

径进行交互;系统操作和组合机制是对各子系统(系统模块)的连接与装配。

代码级(Code):是对系统模块所采用的算法和数据结构设计和说明。系统组件是程序语言元素,如变量和数据类型定义等;操作是程序语言原语(primitives),如算术运算和数学函数等;组合机制是记录和过程等。

执行级(Executable):是对代码执行所需的内存分布、数据映射、堆栈调用和寄存器分配的设计和说明。系统组件是计算机硬件支持的位串;对位串的操作和组合机制通过机器码描述。

在上述三个层次中,软件体系结构是软件体系的最高层描述,它与代码级和执行级设计相互配合,刻画软件系统的不同层面。

1.2 几种典型的结构风格

软件体系结构设计的一个特点是使用成熟的体系结构风格。下面介绍一些较为公认的软件体系结构。在介绍具体体系结构风格之前,首先明确何谓软件体系结构风格。

所谓的软件体系结构风格是一个软件系统组织结构框架,以此框架结构构建的软件系统具有相同的结构级组件、连接器类型和结构约束规约[3]。

下面从组件与连接器类型、结构框架、风格基本特征、典型实例和优缺点等方面剖析典型的软件体系结构风格。

(1)管道-过滤器(Pipes-Filters)

组件与连接器类型:在管道-过滤器结构中,每一个组件具有输入集合和输出集合。它在输入端读取数据流,经处理后在输出端口流出。连接器是连接不同组件,在一个组件的输出端口与另一个组件的输入端口建立连接的通道。形象地把管道-过滤器结构中的组件称为过滤器,而连接器称为管道。

结构框架:管道-过滤器结构框架构成一种偏序拓扑结构,它限定结构中的过滤器按偏序关系连接。连接器是具有一定约束的管道,这种约束主要限制数据在管道中的延迟和管道传输数据的类型。

基本特征:管道-过滤器结构的基本特征主要表现在,过滤器具有独立性、多态性和兼容性。这里的独立性是指,组件与组件之间在逻辑上是独立的,一个组件的行为不受其它组件状态的干预;多态性是指它可以处理来自于输入端口的任何满足要求的数据流,而不关心它是上传流,还是下传流;另一个重要特征是兼容性,这种特征使得一个组件只表现其自身的功能,而不必关系它的前端和后续组件类型。

典型实例:一个典型的实例是 Unix shell[7]。Unix 通过提供参数符合连接组件(Unix processes),并通过实时机制实现连接(pipe)。另一个例子是广域网路由系统,在路由体系结构中,路由器(软件)作为过滤器接收从连接器(通信协议软件)传来的路由信息,并在处理后转发给其它路由器。

优缺点:管道-过滤器结构的优点正如在结构基本特征中总结的那样,管道-过滤器结构具有较好的组件独立性、多态性和兼容性。然而,管道-过滤器也有明显的缺点。其一是根据组件独立性要求,过滤器在输出端口必须提供完整的数据流,以便后续组件处理,这给组件的实现带来难度。其二是在管道传输数据流时往往需要对传输数据编码和解码,这也为实现增加了复杂性。

(2)数据抽象与面向对象(Data Abstraction-Object Oriented)

组件与连接器类型:在数据抽象与面向对象结构中,对象是结构组件,组件通过函数或过程调用(连接器)实现交互。

结构框架:对象具有可继承性,所以数据抽象与面向对象结构是一种可伸展的体系结构框架,即类可以派生子类,各子类只有一个父类。

基本特征:数据抽象-面向对象结构中,组件具有实现逻辑的独立性和对数据操作的封闭性,这些特性使得组件具有较好的可重用性。

典型实例:有很多采用面向对象技术的实例,如基于面向对象的高级语言 C++、J++的实现等。

优缺点:数据抽象-面向对象结构有许多显著的优点,如重用性、封装性等。除此之外数据抽象-面向对象结构还具有可扩充性,即对象标准接口不是一成不变的,用户完全可以根据自己的需要编写新的接口实现版本,以替换系统缺省的接口实现版本,而且一旦替换,就自动在客户的应用程序中生效,并不需要修改应用程序的其它内容。

数据抽象-面向对象结构并不是十全十美的体系结构,也存在不足。一个不足是当一个对象调用另一个对象时,调用方必须知道被调用方的接口。当被调用对象接口改变后,调用方对象接口也必须随之修改,在这一点上它不具备管道-过滤器结构的优点。

(3)事件驱动与事件激发(Event-Based,Implicit Invocation)

组件与连接器类型:在事件驱动与事件激发体系结构中,组件接口提供了一个过程和函数的集合,这点与数据抽象-面向对象结构相似,但组件之间的连接不同于对象调用机制。事件的激发不是通过在两个组件之间建立明确的调用连接,而是通过事件的广播机制把调用消息广播开来,其它组件通过消息注册来接收调用请求,并实现组件激发。

结构框架:事件驱动与事件激发结构是在面向对象技术基础上发展而来,所以在体系结构框架上保持了面向对象体系结构的特点。除此之外,事件驱动与事件的激发体系结构还具有可构造性,即系统结构可随着系统功能的需要增删组件。

基本特征:事件发生后,因事件的激发,系统无法确定哪些组件被激发,以及激发后的结果。也就是说,系统无法确定组件发生顺序,通常把这种特性称为非过程性。

典型实例:部件对象模型是事件驱动与事件激发体系结构的典型应用,如 COM/DCOM、Java Bean、CORBA 等。

优缺点:事件驱动与事件的激发体系结构的优点主要有:对软件重用的有效支持和对软件系统可进化的支持。

该体系结构有以下不足:组件激发结果的不可见性、资源和性能管理的复杂性和系统行为的不确定性。其激发结果的不可见是指,由于组件的独立性使得被激发组件的执行结果与主动激发组件逻辑无关,即被激发组件何时何地执行,执行的结果等都不受激发组件的控制。资源和性能管理复杂性是指,组件之间的数据交互是通过事件消息转发的,这就使得基于该体系结构系统必须建立消息缓冲机制,这给系统管理增加了复杂性,系统性能也不同程度地受到影响。

(4)客户机/服务器结构(Client/Server)

组件与连接器类型:典型的客户机/服务器结构由三部分组成,客户端组件、服务器端组件和中间层组件。客户端组件是用户前端部分,它常常使用图形用户界面,有时也把客户端组件称为服务器的驱动器。中间层组件是体系结构的连接器,一般由 API 和协议组成。服务器端组件是客户/服务器结构的核心,按服务类型可分为文件服务器、数据库服务器、对象

服务器。

结构框架：客户机/服务器结构中的三个部分可能会变得异常复杂，特别是当系统定义模糊时。在许多情况下，客户机与服务器共存于一台计算机，因而从表现形式上无法确切区分客户端组件和服务器端组件。在这种情况下，连接器应采用不同的通信方式。从体系结构角度，有时服务器端组件还可继续分解，形成多层结构，三层客户机/服务器结构就是多层结构的代表。

基本特征：客户机/服务器结构最鲜明的特征共享资源。它可以分配处理任务和集中的数据给客户机和服务器，使系统可以共享从数据到处理能力的每一种资源。

典型实例：客户机/服务器结构是应用增长较快的软件体系结构之一，典型的应用有管理信息系统(MIS)和在线交易系统(OLTP)。

优缺点：客户/服务器结构的优点主要有：有效地支持处理任务分布和系统负载平衡；支持数据资源共享，一个服务器为多个客户服务；支持可扩展的层次结构，即服务器端组件可分解成不同的层次，提供相独立的系统服务。

客户机/服务器结构的缺点主要有：从所支持的业务角度，客户机和服务器组件的功能不易区分，尤其是在层次或多层体系结构设计时更是如此。另一个不足是从两层结构到三层结构的迁移是较困难的，往往需要重新进行系统分析、设计。

以上简要介绍了四种典型的软件体系结构风格，较典型的结构风格还有知识库结构(Repositories)、虚拟机结构(Virtual Machine)等，这里不再一一介绍。实际上各种软件体系结构并不是独立存在的，在一个系统中往往有多种体系结构共存，形成复合的体系结构。

2 软件体系结构研究状况

上面介绍了一些典型的软件体系结构风格，它们可以作为软件系统体系结构设计的参考。软件体系结构作为一个新的计算机研究领域，已经引起学术界和工程界的广泛关注。近期研究成果表明，以下四个方向研究有一定的进展[5,6]。

(1)软件体系结构描述语言(Architecture Description Language，ADL)

目前有关软件体系结构的研究，主要集中在软件体系结构描述语言方面。该项研究的目标是提供一种形式化的软件体系结构描述工具，使得软件系统开发人员能够清晰地描述软件体系结构特性，方便开发人员交流。虽然有关 ADL 的研究时间不很长，但各类风格的 ADL 已有十几种，较著名的 ADLs 有：C2[8]、Wright[9]、Rapide[10]、UniCon[11]、PEADL[12]等。它们采用不同的风格和手段建立基于组件的软件体系结构视图。C2采用基于消息的风格支持对可进化的、动态的分布式组件系统的描述；Wright 提供了对结构级组件间交互和说明能力；Rapide 具有系统仿真能力，并提供了相应的支持工具；UniCon 通过组件间的交互协议生成组件连接代码；PEADL 以进程代数为理论基础，从数学理论角度分析软件体系结构行为，充分体现软件系统结构的可构造性和动态特征，使用 PEADL 可以建立软件体系代数模型与 Semi-Markov 过程的对应关系，并利用随机过程理论分析软件体系结构的系统性能。下面给出软件体系结构描述语言 PEADL 的描述实例。

例1 三层 C/S 应用系统体系结构的 PEADL 描述

```
Architecture A-T-R_Architecture return x/open is
Type
  AP_component:Application_Programe;
  TM_component:Transaction_Manager;
  RMs_component:array(Integer of Resource_Manager);
  TX_connector:TX;
  XA_connector:XA;
  AR_connector:AR;
End type
Configuration
Instence
    ap:AP_commponent; tm:TM_component; rms:RMs_component;
    tx:TX_connector;xa:XA_connector;ar:AR_connector;
  Behavior
    ap to tx;tx to tm;tm to xa;xa to rmap to ar;ar to rm;
  Constraint
    two-phase commit protocol//两段提交协议//
    Coordination Constraint//保证两段提交协议的事物调整规则//
End Configuration;
End A-T-R_Architecture.
```

(2)软件体系结构案例研究

该项研究是通过对软件系统分析，提取软件体系结构典型特征和风格，从而明确构成软件体系结构的框架和内涵。目前公认的软件体系结构如上节所述的 Pipe_Filters、Event-Based Implicit Invocation、Layered Systems 等。

(3)软件体系结构范例研究

该项研究是对典型软件体系结构进行进一步提炼，形成软件体系结构范例。这种范例可以经实例化后生成具体的软件系统。目前有关该方向的研究成果不多。

(4)软件体系结构基础理论研究

软件体系结构基础理论研究，主要集中在形式化的体系结构建模研究。通过形式化建模可以明确区分软件体系结构概念，如结构化的连接、层次化描述和结构风格等。一个理想化的结果是通过软件体系结构形式化描述，提供有效的系统体系结构层分析手段，在系统设计阶段分析系统体系结构的优劣。目前有关形式化建模的主要手段是基于谓词逻辑的形式化描述，而有关系统体系结构的评价方法的研究尚未见到。

3 需要进一步研究的问题

早期的(1960s～1980s)软件系统设计重点在代码级和执行级，体系结构级设计不被重视。虽然有一些体系结构级设计的概念和方法，但这些概念和方法仅局限于一些简单的图表和文本描述，缺少系统的、一致的语法和语义定义。随着软件系统规模的不断扩大，软件体系结构的设计显得越来越重要。由于软件体系结构研究刚刚起步，现有的研究成果不能满足工程需要。总体说来，有以下几个方面需要进一步研究。

(1)软件体系结构基础理论研究

基础理论研究可分为三个方面考虑。

①形式化建模方法研究：不同的建模方法反映的结构特征也有所不同，所以具有普遍意义的建模方法将是建模理论研究的主要内容。形式化的建模方法应对体系结构中各成员的类型、结构框架构成与约束等结构特性建模。

②软件体系结构分析方法研究：形式化的建模方法仅给出了软件体系结构基本特征描述，这种描述应该有利于对软件体系的优劣的评价和系统行为分析。不同类型的分析方法和手段是该领域研究的重点。

③软件体系结构行为检测研究：软件体系结构是软件系统的高层视图，系统行为取决于构成系统各组件和连接器的行为。组件与组件、组件与连接器行为一致性检测是体系结构行为检测的重点。

(2)软件体系结构描述语言研究

软件体系结构描述语言是与建模技术息息相关的描述技术。目前软件体系结构描述语言已有十几种，但各自的风格和描述手段不尽相同，建立一种具有普遍意义的体系结构描述语言将是一件有意义的工作。具体讲这种语言应具有下列品质：

①在结构描述与程序实现建立紧密联系，指导软件开发。

②提供灵活的描述机制，根据描述对象的需要增删语言元素。

③强烈地依赖形式化建模技术，具有简明、方便数学处理等特点。

④使用广泛认同的符号体系作为描述，任何陌生的符号体系都可能是不受欢迎的。

(3)软件体系结构案例研究

随着软件技术的不断发展，新的软件技术将不断出现，因而对新型软件体系结构的挖掘是一项永无止境的工作。在上面已经介绍了 Pipe-Filters 等五种典型的软件体系结构，实际上多种结构风格共存也是一种普遍现象。在一个复杂的结构中区分不同结构风格和特征，规范系统设计与挖掘新结构具有同样意义。

(4)软件体系结构范例研究

软件体系结构范例往往与具体应用领域有关，但体系结构范例应具备的基本特征和基础框架是一个具有普遍意义的研究内容。所以对体系结构范例抽象和描述，以及如何实例化是该项研究的重点。除此之外，对一些具体领域结构范例的研究同样是不可忽视的研究问题。

(5)工具的开发

软件体系结构研究的主要内容是体系结构建模和对所建模型进行分析，无论是建模还是分析都需要工具的支持。工具开发与建模技术和描述语言紧密相关，把工具开发研究作为进一步研究问题列出，其主要原因是该项研究的不足和此项工作的重要性。

4 研究方法

针对上述问题，本节提出解决问题的方法和技术路线，以供读者参考。

(1)软件体系结构基础理论研究　采用进程代数(process algebra)[14]或 Petri 网[15]理论和技术作为软件体系结构建模基础理论，建立软件体系结构的进程代数模型或 petri 网模型。利用进程代数或 Petri 网的建模能力，建立软件体系结构的代数或网络理论体系，并在此基础上对软件体系结构行为进行分析。进程代数与 Petri 网是平行计算机系统建模理论和实用技术，业已证明二者是建模能力相互等价、功能强大的建模理论和技术。把进程代数与 Petri 网引入到软件体系结构建模中来，需要进一步解决的问题是扩展二者对软件系统的建模特性。

(2)软件体系结构描述语言研究　随着软件体系结构研究的深入，对 ADL 的要求也越来越高，能够对软件体系结构分析、检测和评价提供强有力支持的 ADL 将成为新的需求。另一个需要考虑的问题是提出的 ADL 与相关标准的兼容性。有两个标准必须考虑，其一是 ISO1988/1998颁布的计算机系统建模语言 LOTOS/E-LOTOS[16]；其二是统一建模语言 UML[17]。

从对软件体系结构检测支持角度，ADL 是软件体系结构，是软件系统的高层结构视图，它不关心各组件的实现细节，所以构成体系结构的各组件行为一致性是软件体系结构的关键，对软件体系结构组件与组件、组件与连接器行为一致性进行研究，并采用仿真方法对行为一致性检测是一种研究途径。

从对软件体系结构评价支持角度，进程代数和 Petri 网都提供了系统性能分析能力，所以基于进程代数与 Petri 网的 ADL 是值得考虑的技术路线。

(3)软件体系结构案例与范例研究　对新型软件体系结构的挖掘是一项永无止境的工作，但在研究过程中需要对新型软件体系结构及时总结和归类，分析比较新结构的特点。软件体系结构范例与具体应用领域有关，所以从研究方法上，可以与案例研究相结合，建立案例与范例库。

(4)软件体系结构建模工具开发　从工程角度，软件体系结构建模理论和评价方法应对工程具有指导意义。然而，实际工程中问题的状态空间较大，相互关系复杂，一些问题无法通过手工来解决，开发建模辅助工具是当务之急。建模工具包可以把 ADL 作为描述语言，并通过建立等价类的方法对状态空间进行压缩处理，简化计算过程；通过建立与 Matlab 等工具的接口，实现软件体系结构的性能分析任务。

结束语　以上综述介绍了软件体系结构的基础概念、研究现状和进一步需要解决的问题；结合研究经历和相关研究成果，提出进一步研究需要注意的问题和解决问题的方法。由于软件体系结构是一个新的研究领域，新思想和新观点不断出现，因此本文旨在供该领域研究人员参考。

参 考 文 献

1 Vestal S. A Cursory Overview and Comparison of Four Architecture Description Languages:[Technical Report]. Honeywell Technology Center,1993,2

2 Allen R, Garlan D. Formalizing Architectural Connection. In: Proc. of the 16th Intl. Conf. on SW Engineering, Sirrebti Italy. May 1994. 71～80

3 Shaw M, Garlan D. Software architecture: perspectives on an emerging discipline. Prentice-Hall International, Inc. 1996

4 Medvidovic N, Taylor R N. A Classification and Comparison Framework for Software Architecture Description Languages. IEEE,2000,26(1)

5 Garlan D, Allen R, Ockerbloom J. Architectural mismatch, or, why it's hard to build system out of existing parts. In: Proc. of the 17th Intl. Conf. on Software Engineering. Seattle, Washington, April 1995

6 Garlan D. Proceedings of First International Workshop on Architectures for Software Systems:[CMU Technical Report CMU-CS-95-151]. April 1995. Summary reprinted in ACM Software Engineering Notes, July 1995

7 Bach M J. The Design of the UNIX Operating system, chap. 5, Software Series. Prentice Hall,1986. 111～119

8 Medvidovic N, Oreizy P, Robbins J E, Taylor R N. Using Object-Oriented Typing to Support Architectural Design in the C2 Style. In: Proc. of ACM SIGSOFT'96: Fourth Symposium on the Foundations of software Engineering (FSE4), San Francisco, CA, Oct. 1996. 24～32

9 Allen R. A Formal Approach to Software Architecture: [Ph. D. Thesis]. Carnegie Mellon University, CMU Technical Reprot CMU-CS-97-144, May 1997

10 Luckham D C, Vera J. An ERvent-Based Architecture Definition Language. IEEE Transactions on Software Engineering, 1995, 21(9):717～734

11 Shaw M, et al. Abstractions for Software Architecture and Tools to Support Them. IEEE Transactions on Software Engineering, 1995, 21(4):314～335

12 Zhao Hui-Qun, Gao Yuan. PEADL: A Software Architecture Description Language for Performance Analysis. ICYCS2001, Hang Zhou, 11, 2001. 24～26

13 Rumbaugh J, Jacobson I, Booch G. The Unified Modeling Language Reference Manual. Addison Wesley, Reading, Mass, 1999

14 Hillston J, Ribaudo M. Stochastic process algebra: a new approach to performance modeling. Modeling and Simulating of Advanced Computer Systems. Gordon Breach, 1998. 235～255

15 林闯. 随机 Petri 网和系统性能评价. 北京：清华大学出版社，2000

16 ISO/IEC. LOTOS-A formal description technique based on the temporal ordering of observational behaviour. International Standard 8807, International Organization for standardization-Information processing system-Open system Interconnection, Geneve, Sept. 1988

17 Rumbaugh J, Jacobson I, Booch G. The Unified Modeling Language Reference Manual. Addison Wesley, Reading, Mass, 1999