软件体系结构综述*

刘小娜, 常万军, 苗 恺
(河南机电高等专科学校, 河南 新乡 453002)

摘要: 随着计算机软件的快速发展, 软件的复杂性也越来越高, 软件体系结构就显得非常重要。文章介绍了软件体系结构的基本概念、常见结构及其分析评测方法, 最后给出了软件体系结构的未来发展方向。

关键词: 软件体系结构; 分析评测; 管道; 过滤器
中图分类号: TP31 文献标识码: A 文章编号: 1008-2093(2007) 05-0030-04

1 引言

随着计算机技术的快速发展, 计算机的应用也越来越广泛, 人们对各种信息的需求也在快速增长。同时, 对各类信息的处理以及对相关先进技术的应用要求也急剧增加, 而这些需求的实现几乎都离不开计算机软件。伴随信息量的大幅增长及具体实现的不断复杂化, 计算机软件也变得越来越复杂, 软件的体系结构及软件体系结构的评估测试也就显得非常重要。

软件体系结构逐渐成为软件工程领域的研究热点以及大型软件系统开发和产品线开发中的关键技术, 在软件的开发和管理中扮演着越来越重要的角色。对软件体系结构的研究日益成为软件应用开发及产品线研究中的一门重要学科。

软件体系结构是在软件系统的整体结构层次上关注软件系统的构建和组成, 主要任务是研究软件系统是如何构建的、由哪些子系统或部件构成、这些子系统或部件之间的关系是什么。

2 基本概念

当前对于软件体系结构没有统一明确的定义, 但是从结构模式角度来说, 软件体系结构是一个程序或系统的构件的组织结构, 是它们之间的关联关系及其支配系统设计和演变的原则和方针。通俗地说, 一个软件体系结构是由一组构件、连接件和它们之间的约束组成的, 同时还包括系统需求和结构元素之间的对应关系。

构件是相关对象的集合, 运行后实现某计算逻辑。它们或是结构相关(嵌套对象, 被嵌对象是嵌套对象的一部分) 或是逻辑相关(若干聚集对象完成某功能)。构件相对独立, 即不在指定的界面上与它们通信, 外面对象运行不会对它有任何影响。可以以独立单元嵌入到不同的体系结构(也就是不同应用项目) 中, 实现构件的重用。因此构件的定制和规范化十分重要。

连接件是构件的黏合剂, 它也是一组对象。它把不同的构件连接起来, 形成体系结构的一部分, 一般表现为框架式对象或转换式对象(调用远程构件资源)。

约束一般为对象连接时的规则, 或指明构件连接的势态和条件。例如, 上层构件可要求下层构件的服务, 反之不行; 两个对象不得递归地发消息; 代码复制迁移的一致性约束; 以及在什么条件下此种连接无效等等。

因此, 我们可以将软件体系结构抽象地描述如下: 软件体系结构= 构件+连接件+约束。

3 几种常见的软件体系结构及特点

1) 管道和过滤器

“管道—过滤器”模型的基本部件都有一套输入输出接口。每个部件从输入接口中读取数据, 经过处理, 将结果数据置于输出接口中, 这样的部件称为“过滤器”。这种模型的连接者将一个过滤器的输出传送到另一个过滤器的输入, 把这种连接者称为“管道”。

在管道和过滤器中每一组模块都有一组输入和输出。每一模块从它的输入端接收数据流, 在其内部经过处理后, 按照标准的顺序, 将结果数据流送到输出端, 以达到传递一组完整的计算结果的目的。

其结构特征是: ①过滤器是独立的实体。具体来说, 各过滤器之间不能共享状态; ②过滤器与其连接的上下游过滤器之间相互独立。

管道和过滤器的种类主要有三种: ①管线: 管道

*收稿日期: 2007-05-09
作者简介: 刘小娜(1981-), 女, 河南南阳人, 本科, 主要从事图书馆学研究

限制过滤器的拓扑结构只能是线性序列; ②受约束的管道: 受约束的管道限制在其上的流通的数据量; ③有名管道: 要求在两个过滤器之间流通的数据经过严格的定义。

采用该结构的优点为: ①它允许设计者将一个系统的输入输出行为理解为各个独立过滤器行为的一个简单组合; ②这种体系结构支持重用, 维护这种系统或增强其功能很容易, 修改其中的一些过滤器的时候并不影响其他的过滤器; ③它们很自然地支持并发执行。每一个过滤器作为一个单独的任务和单独的进程来实现, 并可以与其他的过滤器并发执行。

采用该结构有如下缺点: ①管道和过滤器常常导致批处理方式; ②在维护两个分离但相关的数据流时, 有一定的困难; ③可能对数据传输标准有一定的要求, 这就导致了为每一个过滤器进行数据语法分析和编码的额外工作, 同时, 在编写过滤器时增加了复杂性, 降低了性能。

2) 面向对象

面向对象的程序设计其实是一种观念, 主要是完成对象的封装、继承、多态等特性。在这种结构中, 数据的表示和与之相连的属性的基本构造都被封装在一个抽象的数据类型或对象中。这种结构的部件就是对象, 或者说是抽象数据类型的实例。对象之间通过函数和过程调用发生相互作用。这里所说的对象就是 C++, JAVA 中的类, 都是封装了特定功能的属性和事件。

其基本特征是: ①对象维护本身的完整性; ②信息隐藏: 对象的结构和方法的实现对其他对象是不可见的。

采用这种结构的优点为: ①对象的隐藏并不影响它的使用者, 更改它的实现, 不会对系统造成任何危害; ②将它们所操纵的数据与访问这些数据的操作绑在一起, 是设计者易于将问题分解成相关对象之间的相互作用。

其缺点为: ①为了实现某个对象与其他对象相互作用, 必须知道对象的标志。这与管道形成了鲜明的对比; ②会产生连锁反应。比如 A 调 B, C 调 B, 那么 A 对 B 的调用可能会对 C 造成影响。

3) 分层结构

分层结构采取层次化的组织方法, 每一层向上一层提供服务, 并利用下一层的服务。比较常见的有 TCP/IP 协议, OSI 七层结构等。图 1 展示了层与层之间的关系。

在 OSI 形成的过程中, 把通信协议有关的不同任务交给不同的层来完成, 使每一层的函数既得到良好的定义又与其他层分开。服务的定义是从底层向顶层形成的。图 1 表示了相邻两层之间的关系, N 层通过服务访问点向 N+1 层提供服务, 但同时又利用了 N-1 层的服务。

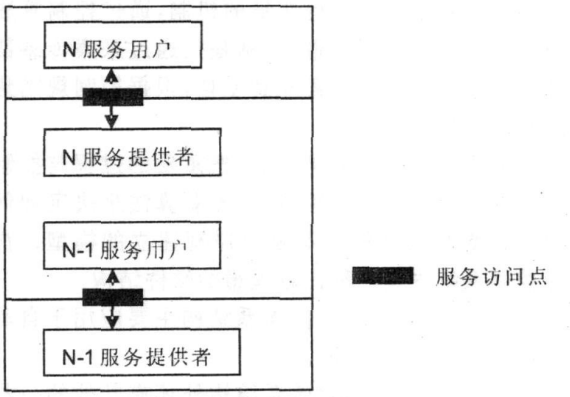

图 1 层与层之间的关系

采用这种结构的优点为: ①支持基于抽象程度递增的系统设计, 这使得设计者可以把一个复杂的系统按递增的步骤分解开; ②跨层影响小, 因为每一层只和相邻的层打交道, 因此功能的改变至多影响上下层; ③支持复用, 只要提供的服务接口定义不变, 同一层的不同实现可以交换使用。这对整个系统并无影响。

其缺点为: 并不是每一个系统都可以很容易地划分为分层的模式, 甚至即使一个系统的逻辑结构是层次化的, 出于对系统性能的考虑, 不得不把一些低级的功能或高级的功能耦合起来。

4) 基于事件的隐式调用

一个软件体系结构的部件可以声明一个或多个事件, 而不是直接的调用过程。系统中的其他成分可以通过将一个过程连接到一个事件上来表示对该事件感兴趣。当事件激发时, 系统本身就会调用所有的已注册的与该事件有关的过程。

例如: 我们用 JAVA 编写的一个程序, 当为某变量申请内存失败时, 系统就会抛出一个内存异常, 系统就会产生一个异常事件, 系统通过调用注册的相应的中断程序、接管程序, 进行相应的处理。

采用这种结构的优点为: ①事件声明者不用知道哪些部件会被事件影响, 因此软件部件不能确定处理的先后顺序, 甚至不能确定事件会导致哪些处理过程; ②对软件复用的大力支持, 任何软件部件, 只要声明了数据和处理过程的连接关系, 就可以加入系统; ③系统容易升级, 一个软件部件可以被另一个部件替代, 而不影响接口。

其缺点为: ①软件部件放弃了对计算的控制, 而是完全由系统控制完成。当一个部件声明一个事件时它不知道别的部件会如何处理, 即使知道也无法调用其他部件来完成某种功能; ②存在数据传递问题: 有时数据可以通过事件来传递, 但大多时候系统本身需要一个存储库来实现, 对系统的功能和资源管理有

一定的影响。

5) 过程控制

过程是指某些过程变量的机制, 通过控制算法来决定如何操纵过程变量, 控制是指通过采集传感器而建立起连续更新的过程控制变量, 根据控制规则建立起相应的控制模型。

采用这种结构的优点为: 把重要的过程与控制规则分离开。它支持复用, 通过控制算法来决定如何操纵过程变量的过程并不影响控制建立的模型。在工业控制系统中是一种比较理想的软件结构。

其缺点为: 这种软件体系结构主要应用于自动化控制中, 应用范围受限。

当然, 除上述几种体系结构外还有其他的一些体系结构, 比如虚拟主机、数据仓库等模型, 在此不再一一赘述。

4 软件体系结构分析与评价方法

体系结构的描述是对其进行分析和评价的基础, 对体系结构的不同认识反映在看待体系结构的不同的观点上, 形成了不同的体系结构描述。已经提出的观点包括: 多视角观点、ADL(architecture description language) 观点、体系结构风格观点、设计模式观点。

软件体系结构分析评价分为①定性分析: 采用基于 check list, questionnaire 和场景的分析评价技术; ②定量分析: 采用基于度量指标、模拟、原型系统和数学模型等技术。在关注具体的质量特性时, 在上述技术基础上, 还需要特定的技术和方法支持。例如体系结构可靠性分析中的可靠性增长模型, 安全性分析中的失效模式和效果分析(failure modes and effects analysis, FMEA), 以及性能分析中排队理论和调度理论的应用等。

具有代表性的一些方法

1) SAAM (Software Architecture Analysis Method) 方法

SAAM 方法是第一个被广泛接受的体系结构分析评价方法, 适用于可修改性、可拓展性以及功能覆盖等质量属性。SAAM 方法有着以下缺陷: 没有提供体系结构质量属性的清晰的度量; 评估过程依赖专家经验等, 只适合对体系结构的粗糙评价。

2) ATAM (Architecture Trade－Off Analysis Method) 方法

在SAAM 的基础上, SEI 于 2000 年提出 ATAM 方法, 针对性能、实用性(availability)、安全性和可修改性, 在系统开发之前, 对这些质量属性进行评价和折中。ATAM 方法是被验证有效和广泛使用的一种方法, 但对质量属性并没有进行深入分析, 缺少定量的数据来支持分析的结果。

3) ALPSM (Architecture Level Prediction Software Maintenance) 方法

ALPSM 方法是 Bengtsson 和 Bosch 提出的在体系结构层次预测系统可维护性的一种方法。ALPSM 方法结合设计经验和历史数据对可维护行框架进行验证, 并且有效地引入变更, 预测系统的可维护性。方法的缺陷是具有不确定性。

4) ALMA (Architecture Level Modifiability Analysis) 方法

由于软件 50％~70％的成本都用于软件的演进, 因此可修改性的设计和分析对于减少软件的成本具有重要的意义, Bengtsson 等人 2004 年提出基于预测的软件体系可修改性的分析方法。ALMA 方法引入了定量的度量指标, 支持从风险评估、成本预测、体系结构选择等多个角度评估体系结构的可修改性, 并提供了场景构建的停止准则。ALMA 方法的缺点是缺少对结果准确性的判断和风险评估完整性的判断。

5) ALRRA (Architecture Level Reliability Risk Analysis) 方法

ALRRA 方法是 Yacoub2002 年提出的一种软件体系结构层次的可靠性风险分析方法。方法基于如下假设: 构件执行的频率越高, 失效的可能性越大。ALRRA 方法以定量的动态指标(动态复杂度度量、动态耦合度量、启发式风险要素) 做基础, 通过 Object Time tool 来模拟运行。支持 FMFA 分析和 CDG 的计算, 分析和识别体系结构中风险最大最关键的构件和连接子, 并且 ALRRA 可以自动化执行。但是 ALRRA 方法在以下方面需要改进, 度量指标参数(场景执行的概率、平均执行时间) 的不确定性对评价结果的影响, 在大型系统分析中的应用等。方法因为采用了模拟和场景结合的技术, 成本比单独采用场景技术的方法高。

6) SAE4＋1View (Software Architecture Evaluation with the 4＋1 View Model of Architecture) 方法

SAE 4＋1View 方法是 Heeseok 和 Keunhyuk 于 2002 年提出的一个多视角的体系结构评价方法。SAE 4＋1View 的缺陷在于虽然适用于多个质量属性, 但对质量属性的分析没有提供详细的支持, 缺少清晰的度量。并且对体系结构的功能需求和质量需求的关联缺少系统的方法支持, 依赖于经验, 具有很多不确定性。

7) SACAM (Software Architecture Comparison Analysis Method) 方法

SACAM 方法是 SEI 于 2003 年提出的体系结构比较分析方法, 方法从组织商业目标中提出准则, 一个准则明确地表达支持组织商业目标的体系结构的

一个需求,可以演化为质量属性场景(quality attribute scenario)。SACAM 方法根据组织商业目标,通过比较不同体系结构对系统的满足程度,提供了体系结构选择的基本原理。但 SACAM 方法中,SA 的描述需要是一致的、可比较的,SA 描述方式的多样性给 SACAM 的应用带来了一定的困难;另外 SA 描述者的经验以及质量属性场景的表示粒度等也使这种方法具有一定的不确定性。

8) Rapide 方法

Rapide 语言是一种可执行的面向对象的 ADL,允许在系统开发的早期阶段对体系结构进行模拟和行为分析,采用基于事件的方法对分布式系统的体系结构建模并构造原型系统。Rapid 方法基于模拟和原型系统,适用于对分布式系统体系结构的形式化分析,是一种比较严格和精确的分析方法。但是对体系结构的描述要求用特定的语言描述,为评价开发一个详细的模拟工具或者原型的成本也较高。

9) ESPA－SAPE 方法和 QNM－SAPE 方法

ESPA－SAPE 根据体系结构的特点,对随机进程代数 PEPA 进行扩展,建立体系结构的 ESPA 模型,求解具有 Markov 或者 semi2Markov 特性的转换矩阵,进而求解稳定状态概率分析,计算系统的性能指标。

QNM－SAPE 方法对体系结构采用化学抽象机(chemical abstract machine ,CHAM)进行形式化描述,基于队列网络模型(queuing network model ,QNM),导出体系结构性能评价模型,用于多个候选体系结构的选择。

上述软件体系结构的分析评测方法都是一些比较经典的分析方法,这些方法各有其自身的特点及其使用范围,在不同的领域发挥了极为重要的作用,赢得了广泛好评。但是没有哪种方法是完美的,不同的方法都在一定程度上存在缺陷,针对这些缺陷又有一些新的评测分析方法被提出,并在一定程度上是可行的。比如在文献[1]中作者就提出了一种将 Object－Z 和时态逻辑的结合来建立一种结构化的描述,从而可以对大部分的软件体系结构进行评测;在文献[2]中作者则针对管线模型提出了一种新的评测方法,从而解决了以往没有考虑过滤器失效的问题,并通过实例验证了这种方法的可行性及其发展潜能。

5 结束语

计算机在各行各业的广泛使用使计算机软件的复杂性大幅度提高,随着复杂性的不断提高,很多问题接踵而至,软件体系结构的选择也就相当重要,同时对软件的分析评测就变得非常必要。所以,软件体系结构需要不断地优化、改进,对应的各种分析评测标准也要不断地完善。软件体系结构及其分析评测标准将会越来越受到广大工程人员的重视和研究,在未来的软件开发和管理中会扮演极为重要的角色。

(责任编辑　吕春红)

参考文献:

[1] 刘霞,李明树,王青,周津慧. 软件体系结构分析与评价方法评述[J]. 计算机研究与发展,2005,42(7):1247-1254.

[2] 麦中凡,戴彩霞. 软件体系结构的概念[J]. 计算机工程与应用,2001,(11):92-95.

[3] 杨志明. 几种常见软件体系结构模型的分析[J]. 计算机工程与设计,2004,(8):1326-1328.

[4] 李小龙,毛文林. 管道和过滤器模式的软件体系结构及其设计[J]. 计算机工程与应用,2003,(35):114-115,182.

[5] 张广泉,郑建丹,舒明. 基于时态逻辑语言 XYZ/E 的软件体系结构研究(I)-基本组件和连接件的描述[J]. 重庆师范学院学报(自然科学版),2001,(18):1-4.

The Summarize of Software Architecture

LIU Xiao-na, et al

(Henan Mechanical and Electrical Engineering College, Xinxiang 453002, China)

Abstract: The widespread application of the computer has caused fast development of the software. The software's complexity becomes more and more high. So the software architecture becomes very important. This paper introduces basic concept、common structure and the analysis method of the software architecture. Finally it shows the future of the software architecture.

Key words: software architecture; analyze; conduit; filter