第 34 卷 第 9 期 计 算 机 工 程 2008 年 5 月
Vol.34 No.9 Computer Engineering May 2008
·软件技术与数据库· 文章编号：1000—3428(2008)09—0048—02 文献标识码：A 中图分类号：TP311

基于构件运算的软件体系结构设计方法

张友生，李 雄
（湖南师范大学计算机应用技术研究中心，长沙 410081）

摘 要：软件体系结构对软件的稳定性、软件维护和软件演化等各方面的性能很重要。该文引入构件运算的方法描述体系结构，分析调用运算、协作运算和条件运算的相关性和构件的复杂度，综合体系结构的多方面因素，采取局部性能指标方案对体系结构的性能进行分析评估，利用最佳性能方法调整构件关系和体系结构性能指标，设计出最优性能的软件体系结构。

关键词：软件体系结构；构件运算；性能

Design Method of Software Architecture Based on Component Operation

ZHANG You-sheng, LI Xiong
(Research Center of Computer Application Technologies, Hunan Normal University, Changsha 410081)

【Abstract】 Software architecture is important in stability, maintenance and evolution of software. This paper introduces component operations to descript software architecture, analyzes the component relativity and component complexity, and synthesizes various factors by using the optimal performance method to adopt the relations of components and design the optimal performance of software architecture. This method can conduct developers to design software architecture in developing software, and the result shows that the software architecture by this design method is optimal for the system.

【Key words】 software architecture; component operation; performance

1 概述

如何描述和设计体系结构；如何验证一个体系结构是否符合期望的系统需求；如何开发出一个适合系统性能要求的最优体系结构，是基于构件和体系结构的软件开发方法所必须研究和解决的核心问题[1]。

当前对软件体系结构的描述均采用图形加文本的方式或形式化描述语言[2-3]，但对构件的动态行为、互操作和演化特征方面的描述明显不足，无法系统科学地描述软件的可靠性、功能性和演化等特征。因此，有学者用代数理论对构件的属性和行为特征进行抽象，研究构件调用运算、协作运算和条件运算及这 3 种运算之间的关系[4]。生成一个满足软件需求的体系结构的过程即为体系结构设计。体系结构设计的本质在于：将系统分解成相应的组成成分(如构件、连接件)，并将这些成分重新组装成一个系统。体系结构设计有 2 大类方法：过程驱动方法和检查列表驱动方法。过程驱动方法包括：(1)面向对象方法，与OOA/OOD相似，但更侧重接口与交互；(2)“4+1”模型方法[5]；(3)基于场景的迭代方法。基于过程驱动的体系结构设计方法适用范围广、易于裁剪、具备动态特点、通用性与实践性强。而检查列表驱动法的基本思想是枚举设计空间，并考虑设计维的相关性，以此来选择体系结构的风格。显然，该方法是静态的，适用于特定领域，并可以实现量化体系结构设计空间。这 2 类设计方法都有各自的优点和不足，不适合通用的高层抽象级体系结构设计。因此，开发一个体系结构性能的评估方案和设计方法对指导体系结构的设计具有重大意义。本文设计了一个最优性能的软件体系结构来解决上述问题。

2 构件运算及相关概念

2.1 构件运算及体系结构的抽象模型

构件是指可重用的用于构造其他软件系统的软件单元，可以是被封装的对象、一些功能模块、软件系统模块等。构件的抽象模型及其相关概念定义如下：

定义 1 软件系统的构件 $C=(O, A, X, P)$，其中，O 是组成构件的所有对象的集合；A 是构件属性的集合；X 是构件动作的集合；P 是构件端口的集合。

定义 2 连接件 $L=<R,G>$，其中，R 是连接件中角色进程所组成的集合，通常有 2 个元素；G 是胶水(Glue)的集合，该集合中只有一个元素。

在软件系统中，构件不是独立存在的，构件之间具有互操作性。下面把构件间的互操作关系抽象为代数运算[4]。

定义 3 设构件C_A和构件C_B是软件系统SA中的 2 个不同构件，若C_A使用了C_B的服务，就称C_A和C_B进行了一次调用运算，记作C_A+C_B。

定义 4 设构件C_A和构件C_B是软件系统SA中的 2 个不同构件，若C_A和C_B互相协作完成同一功能C_O，就称C_A和C_B进行了一次协作运算，记作$C_A\times C_B$。

定义 5 设构件C_A和构件C_B是软件系统SA中 2 个不同的构件，若C_A的执行是以C_B的执行作为前提，即C_B的输出是C_A的输入，就称C_A和C_B进行了一次条件运算，记作 $C_A\oplus C_B$。

作者简介：张友生(1969－)，男，副教授、博士，主研方向：软件体系结构，Web 工程；李 雄，硕士

收稿日期：2007-06-15 **E-mail：**hitech@hunnu.edu.cn

基于构件运算的构件关系如图 1 所示。

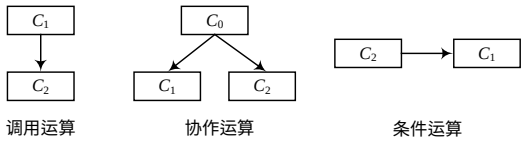

图 1　构件关系

2.2　构件相关性度量模型

定义 6　构件C_A和构件C_B的条件相关性(condition coupling)度量$CdCpl_{A,B}$定义为C_A在访问C_B的条件路径上满足的条件数，路径越长，条件数越多，体系结构复杂性越高，其衡量标准为

$$f(Cond)=n\cdot\alpha_k$$

其中，n 为条件构件数；α_k 为构件条件相关性系数。

定义 7 构件C_A和构件C_B的调用相关性(transfer coupling)度量$TsCpl_{A,B}$定义为C_A到C_B在它们的调用路径上的长度，路径越长，调用深度越大，体系结构复杂性越高，其衡量标准为

$$f(Tran)=n\cdot\beta_k$$

其中，n 为调用构件数；β_k 为构件调用相关性系数。

定义 8 构件C_A的协作相关性(combine coupling)度量$CmCpl_A$，定义为完成C_A功能所需协作构件的个数，协作构件数越多，体系结构的复杂性越高，其衡量标准为

$$f(Comb)=n\cdot\gamma_k$$

其中，n 为协作构件数；γ_k 为构件联合相关性系数。

定义 9 构件 C_i 复杂性度量定义为

$$f(C_i)=n\cdot perf_{Ci}(S_j(C_i))$$

其中，$perf$ 表示构件的某种性能属性(如时间响应、通信延时)；S_j (j=1,2,⋯,n)为构件 C_i 提供的 n(n≥1)种服务。构件 C_i 的服务数越多，构件的复杂性越高，构件在某方面的性能就越低。

定义 10 软件体系结构的性能度量定义为

$$f(SA)=\sum_{i=1}^{n}(f_{C_i}(Cond)+f_{C_i}(Tran)+f_{C_i}(Comb))+\sum_{i=1}^{n}\lambda_h\cdot f(C_i)$$

其中，f(SA)表示软件体系结构 SA 在总体性能上的复杂度；λ_h 表示构件在某性能上的参数；n 表示体系结构中构件的个数。f(SA)的值越大，说明设计出的体系结构不是满足系统某方面性能最优的体系结构；反之，说明该体系结构是最优的。

由于连接件在整个系统中只负责信息的传递，并不具体地实现系统某些功能，因此忽略连接件的复杂性及其性能。

3　最优性能体系结构设计通用方法

体系结构为软件系统提供了一个结构、行为和属性的高级抽象，因此，设计出满足系统性能需求的体系结构是必要的。满足系统需求的高性能体系结构设计的通用方法如下：

(1)设计体系结构关键用例模型。系统开发者必须从客户需求出发来建立满足系统性能要求的软件体系结构。体系结构对系统的非功能属性(如系统性能、可扩展性、可靠性、安全性、互操作性)的影响尤为深刻，这就要求在进行系统低级别的设计之前，仔细分析系统需求，并在此基础上进行体系结构设计，找出对整个系统影响较大的关键用例。

(2)建立体系结构性能目标。为第(1)步中找出的关键用例定义体系结构性能目标。性能目标能够提供一个定量的标准，以便对开发中的系统进行定量分析。这个目标可以通过多个途径加以描述，如构件性能、连接件性能、体系结构性能。

(3)选取构件、连接件建立体系结构性能模型。从构件库中选取能够完成系统功能且具有较好性能的构件和连接件，所选取的构件都为原子构件。并建立软件体系结构的性能模型，以供下一步分析评估。

(4)进行体系结构性能评估。在软件开发的各个阶段对体系结构性能目标中各部分的性能进行评估。在此，可以通过体系结构复杂度函数 f(SA)中的各分量进行分析评估。

(5)进行结果分析及优化体系结构。这一步的主要目的是分析评估结果，通过优化算法对软件体系结构进行优化设计。把这些评估值和性能目标进行比较，如果能够满足目标就进行系统的开发，否则回到第(3)步，重复以上步骤，直到满足体系结构性能目标。如果都无法满足，那么就需要改进体系结构或者改变体系结构性能目标。

第(5)步的优化设计是整个方法的核心，它使设计出的体系结构满足系统性能目标最优。

体系结构优化设计算法如下：

输入 带性能目标参数的软件体系结构，构件复杂性度量函数为 $f(C_i)$。

输出 满足性能目标的最优软件体系结构。

(1)把系统中对应的构件的服务数和系数及对应的相关性参数赋值给相应的变量。

(2)通过广度优先搜索(BFS)算法求出从初始构件起各个构件所包含的调用运算、协作运算及条件运算的个数，分别计算其相关性度量 $f(Tran)$, $f(Comb)$, $f(Cond)$。

(3)通过软件体系结构复杂度函数 f(SA)评估体系结构的性能目标。

(4)体系结构设计优化过程如下：

1)从初始构件开始通过广度优先搜索算法逐层搜索每个构件及其关系，并把已检测到所包含构件个数的体系结构复杂度定义为 $f(SA(k))$ (k=1,2,⋯,n)。

$$f(SA(k))=\sum_{i=1}^{k}(f_{C_i}(Cond)+f_{C_i}(Tran)+f_{C_i}(Comb))+\sum_{i=1}^{k}\lambda_h\cdot f(C_i) \tag{1}$$

2)合并直接关联的调用构件或条件构件，或者合并协作构件，并计算其体系结构复杂度：

$$f(SA(k-1))=\sum_{i=1}^{k-1}(f_{C_i}(Cond)+f_{C_i}(Tran)+f_{C_i}(Comb))+\sum_{i=1}^{k-1}\lambda_h\cdot f(C_i) \tag{2}$$

3)如果 $f(SA(k-1))<f(SA(k))$，则合并构件，继续往下搜索；否则不合并构件，返回 1)。

4)搜索完所有构件，并进行优化使得 $f(SA)$ 为最小。

(5)输出满足性能目标的最优软件体系结构，结束。

该算法通过是否合并相关构件来优化体系结构。图 2 为某系统软件体系结构。

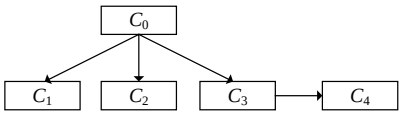

图 2　某系统软件的体系结构

在该体系结构中，C_3以C_4为条件，C_1, C_2, C_3 协作完成C_0的功能，假设其对应的参数都为已知，则通过式(1)可以计算其复杂度 $f(SA(5))$。按照上述算法，采用BFS算法从C_0

（下转第 57 页）