基于Petri网的软件架构演化波及效应分析

吴小兰,王忠群,刘 涛,王 勇
(安徽工程科技学院 计算机科学与工程系,安徽 芜湖 241000)

摘 要:Petri网是一种系统描述和分析的工具。在构造系统PSA模型的基础上,利用其可达标识图同样能分析软件架构演化中的构件删除、增加、修改以及合并与分解各种变化引起的波及效应,且算法直观、复杂度小;并为运用Petri网技术研究SA提供了一个新思路。

关键词:Petri网;演化;软件架构;可达标识图;波及效应
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2007)12-0099-04

Ripple-Effect Analysis of Software Architecture Evolution Based on Petri Net

WU Xiao-lan, WANG Zhong-qun, LIU Tao, WANG Yong
(Department of Computer Science and Engineering, Anhui University of Technology and Science, Wuhu 241000, China)

Abstract: Petri net is a useful tool for system description and analysis. Based on constructing software architecture model by Petri net (PSA), can calculate the relative quantity of component effect via the reachable marked graph of Petri net and spend less time, and every ripple-effect of SA evolution caused by component deletion, addition, modification, division and combination is described respectively in this thesis. Meanwhile it's a new thought to use Petri net technology to research SA.

Key words: Petri net; evolution; software architecture; reachable marked graph; ripple-effect

0 引 言

在软件工程实践中,随着软件系统规模和复杂性的增长,系统总体结构设计的重要性已远远超过了特定算法和数据结构的选择,良好的软件架构是保证系统成功的关键。软件架构目前已经成为软件工程的热点之一。

软件架构设计位于系统开发的前期。在这一层次上的设计主要包括以下内容:系统中构件的描述、构件之间的交互、指导构件交互的模式以及施加在模式上的约束等[1]。一个软件架构刻画了系统的3个方面[2]:

(1)软件总体结构:架构采用高层抽象的计算成分和它们之间的相互作用刻画了系统的结构(structure);

(2)构件交互方式的抽象:在架构设计中,构件之间的相互作用反映了系统设计人员对构件交互方式所做的抽象;

(3)全局特性:架构设计的一个主要作用是刻画系统的整体行为。因此,软件系统的架构通常关注的是系统的一级行为特征,如端对端的数据传输率和响应时间,系统的一部分对其他部分失效的适应程度,或者当系统的某一部分修改后,所造成的波动影响范围等。

着眼于有关系统架构全局特性方面的问题,文中在系统的基于Petri网[3]的SA模型基础上,利用Petri网的可达标识图给出了一种计算构件贡献大小的方法,同时还给出了架构演化中构件删除、增加、修改以及合并与分解各种变化引起的波及效应分析,与文献[4]相比,能达到相同目的,但比文献[4]更为简单、有效、方便,最后还进行了系统活性分析。

1 基于Petri网软件架构演化波及效应分析

在阐述文中的"基于Petri网的软件架构演化波及效应分析"时,会用到软件架构中的一些相关内容,因此以下先简单说明。

收稿日期:2007-02-14
基金项目:安徽省自然科学基金重点项目资助(2006KJ016A,2005KJ065)
作者简介:吴小兰(1982-),女,安徽宿松人,硕士研究生,研究方向为分布式计算、软件工程;王忠群,教授,硕士生导师,研究方向为软件工程、分布式计算、工作流技术。

1.1 软件架构及其定义

软件架构 SA(又常被称为软件体系结构,software architecture)已成为软件研究热点之一,它作为软件的蓝图,描述整个系统的结构和行为模型,为人们宏观把握软件的整体结构提供了一条有效途径。目前对 SA 的定义形式多样,为了研究方便,采用许多文献中公认的简单定义,即 SA 是组成系统的构件以及构件与构件之间交互作用关系(连接件)的高层抽象。

定义 1:Garlan&Shaw 模型[5]:SA = {components, connectors, constrains}

其中:构件(component)可以是一组代码,如程序的模块,也可以是一个独立的程序,如数据库的 SQL 服务器;连接件(connector)表示构件之间的相互作用,它可以是过程调用、管道、远程过程调用等;一个软件架构除了构件和连接件外,还包括构件和连接件如何结合在一起的某些约束(constrains)和配置关系(如图 1 所示)。

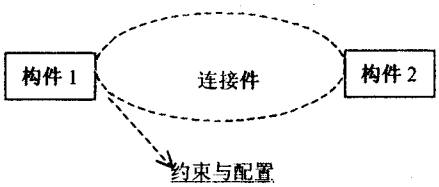

图 1 软件架构示意图

构件作为一个封装的实体,仅通过其接口与外部环境交互,而构件的接口由一组端口组成,每个端口表示了构件和外部环境的交互点。连接件作为建模软件架构的主要实体,同样也有接口。连接件的接口由一组角色组成,连接的每个角色定义了该连接表示的交互的参与者[6]。

1.2 基于 Petri 网的 SA 模型 PSA

Petri 网利用网络图形描述对象之间的输入输出关系,能很好地反映系统的静、动态特性,Petri 网的动态特性通过托肯(token)的移动和传播进行控制[7]。

以文献[4]中 SA 模型为例,假设一个系统的 SA 由 5 个构件和 6 个连接件构成,如图 2 所示,其中构件 Component1 通过连接件(单箭头)Connector1, Connector3, Connector5 和 Connector6 分别与构件 Component2, Component3, Component4 和 Component5 发生交互关系,其他构件之间的关系亦然。该 SA 模型能在宏观层面上为建立基于 Petri 网的 SA 模型提供足够的信息。针对该 SA 模型,建立系统的基于 Petri 网的 SA 模型 PSA。

根据定义 1 中 SA 的说明,提出一个新模型 PSA = {Sp, Tp; Fp},建模原则如下:Sp = {components};Tp = {connectors};Fp = {constrains}。

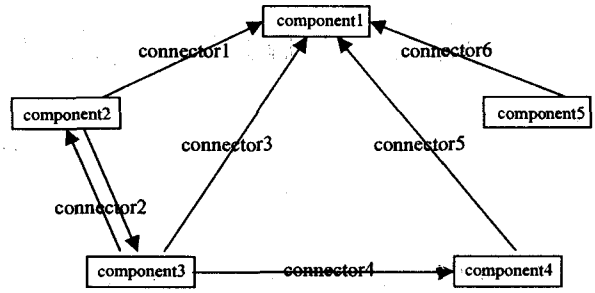

图 2 系统 SA 模型

其中库所用圆圈(○)表示,用粗杠(|)表示变迁,用从 x 到 y 的箭头表示流关系中的(x,y)。则图 2 转变为基于 Petri 网的 SA 模型 PSA(Petri - Software Architecture),如图 3 所示。其中,库所 S_i 对应构件 C_i,t_i 代表连接件,其初始标 M 可根据要求再给出。

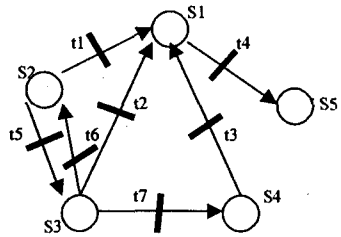

图 3 系统 PSA 模型

该模型的变化借助 Petri 网中的点火规则[8]:

假设有向网 $N = (S, T; F)$,$N_0 = \{0,1,2,\cdots\}$,$N_1 = \{1,2,3\cdots\}$,ω 表示无穷:$\omega = \omega + 1 = \omega - 1 = \omega + \omega$。$K$ 是 $S \to N \cup \{\omega\}$ 称为 N 的容量函数;M 是 $S \to N_0$ 称为 N 的一个标识的条件是:$\forall s \in S: M(s) \leqslant K(s)$;$W$ 是 $F \to N_1$ 称为 N 上的权函数,对$(x,y) \in F$,$W(x,y) = W((x,y))$ 称为(x,y) 上的权:若 $t \in T$ 是一个变迁元素,t 在标识 M 下有发生权的条件是:

$$s \in {}^{\cdot}t: M(s) \geqslant W(s,t) \wedge s \in t^{\cdot}: M(s) + W(t,s) \leqslant K(s)$$

把 t 发生后 M 的后继标识记作 M',则 M' 和 M 的关系如下:

$$M'(s) = \begin{cases} M(s) - W(s,t) & \text{若 } s \in {}^{\cdot}t - t^{\cdot} \\ M(s) + W(t,s) & \text{若 } s \in t^{\cdot} - {}^{\cdot}t \\ M(s) - W(s,t) + W(t,s) & \text{若 } s \in {}^{\cdot}t \cap t^{\cdot} \\ M(s) & \text{若 } s \notin {}^{\cdot}t \cup t^{\cdot} \end{cases}$$

(1)${}^{\cdot}t = \{y \mid (y,t) \in F\}$ 称为 t 的前集(pre - set)或者输入集。

(2)$t^{\cdot} = \{z \mid (t,z) \in F\}$ 称为 t 的后集(post - set)或者输出集。

PSA 模型按照上述规则,对使能的变迁执行点火,可以反映出被模拟系统的动态特性[8]。结合图 4 来说

明所提出的新模型 PSA 是可行的。假定在图 4 中库所 S_2 中有一个 token,初始标 M = (01000),则根据 PSA 模型可得可达标识图见图 5。

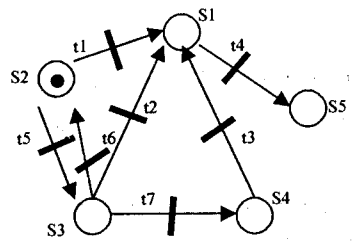

图 4　系统 Petri 网

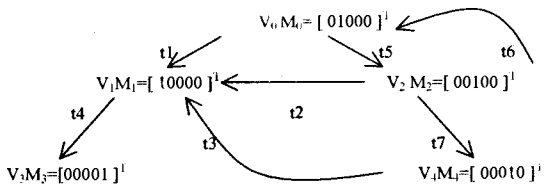

图 5　系统 PSA 的可达标识图 RMG

发现用 PSA 模型建立可达标识图 RMG 所耗的时间复杂度是 $O(n*n)$,其中 n 是构件的个数(下文出现的 n 都表示相同意思),虽与利用系统建立可达矩阵的时间复杂度 $O(n*n)$ 相同,但是所提出的基于 PSA 模型的可达标识图更加直观,易于计算,并且用它来分析软件架构的性能时,发现它所耗废的时间复杂度远远小于重建可达矩阵。

1.3　PSA 模型的演化波及效应分析与系统活性分析

1)构件贡献大小及波及效应分析。

通过系统 PSA 的可达标识图 RMG 可以很容易界定某一构件变化所影响的其他构件,即构件对 SA 影响的大小(称为贡献或者贡献大小),比文献[4]中的通过系统可达矩阵计算更加直观、更加容易计算,而且构件变化后不一定需要每次重新计算,而可达矩阵方法需要每次系统更新后重新计算其可达矩阵。

通过系统可达标识图,可以对各个构件对 SA 影响的大小进行确定和排序。具体办法是,求取可达标识图中各个构件对应标识的不同的后继标识个数,其数值代表相应构件所影响的其他构件的个数。若记 $\sigma(M_j)$ 为标识 M_j 的不同的后继标识个数,则 $\sigma(M_j)$ 越大,构件 M_j 对 SA 的影响(贡献)越大。以图 5 的可达标识图为例,可达标识 M_0 表示 token 处在库所 S_2 中,则 $\sigma(M_0)=5$ 即为库所 S_2 代表的构件 C_2 的贡献大小,依此类推,标识 $\sigma(M_1)=1$ 为构件 C_1 的贡献大小,$\sigma(M_2)=5$ 为构件 C_3 的贡献大小,$\sigma(M_3)=0$ 为构件 C_5 的贡献大小,$\sigma(M_4)=2$ 为构件 C_4 的贡献的大小,即构件 C_1, C_2, C_3, C_4, C_5 的贡献大小分别 1,5,5,2,0,所以各个构件对 SA 的贡献大小排列为 $C_2 \geqslant C_3 > C_4 > C_1 > C_5$(结果与文献[4]一致)。

下面对用可达标识图分析 SA 演化中基本活动下的构件波及效应,如构件的删除、增加、修改以及构件合并与分解 5 种活动,具体分析如下:

(1) 删除 SA 中一个构件 C_j。

在可达标识图中,如果是死节点(即标记不能使任何变迁发生的叶节点),则可以直接删除该节点标识所对应的构件及其连接件,除了影响系统 SA 的结构外,不会对其它构件造成影响。例如,图 5 中死叶节点 $[00001]^T$ 标识了 token 在库所 S_5 中不能使得任何变迁发生,即对应的构件 C_5 及 t_4 可以直接删除,不会影响其他构件,具体表现为软件功能的删减,而且删除后只要更改标识的向量维数即可得到新系统的可达标识图分析新系统特性,所耗的时间复杂度只是 $O(n)$,与重新计算可达矩阵相比,时间复杂度更少,并且简单,有效;如果不是死节点,则删除系统构件 C_j 对 SA 的结构和整个软件的功能及性能等会发生影响,其影响的大小就为 C_j 对应的 $\sigma(M_j)$,如删除构件 C_3(其 σ 值为 5)比删除构件 C_4(其 σ 值为 2)的影响要大。

(2) 新增一个构件 C_j。

在系统中新增加一个构件时,只需新增标识维数,把新增的使能标识添加到系统可达标识图中,就能得到系统的可达标识图,重新分析新系统中各构件的贡献大小,复杂度才为 $O(2*n)$,不及重新计算系统可达矩阵。

(3) 修改某个构件 C_j。

修改构件可是构件自身结构的调整和功能的增减,构件自身结构的调整不影响系统的 RMG(PSA),因而其构件贡献大小不变;对于构件功能的增减,可能会导致 SA 在结构上或者语义上的变化,但是这种变化实施构件变更的人员事先已能把握,因而新 SA 与其对应的可达标识图也能被确定。

(4) 合并 SA 中一簇构件。

对一簇构件进行合并是经常发生的,如功能相近的构件的合并、小构件组装成更大的构件、遗留构件的进一步包装(符合新的标准)等都可引起构件的合并[4]。

在构件的合并过程中,要完成两项工作:

a. 依据系统的 SA 简化模型指导构件的有效合并;

b. 被合并的构件簇确定的前提下,新的 SA 可达标识图的快速求取。

例如,在图 2 中,构件 2 和构件 3 交互操作关系强(双向交互关系),则可以考虑合并构件 2 和构件 3。现假设合并构件 2 和构件 3 为一簇,新系统的可达标识图可按照如下方法来快速获取:

对图5中各个标识 M_0、M_1、M_2、M_3、M_4(列向量)中第2个和第3个元素求"逻辑或"运算,则分别得 $M_0 = [0100]^T$、$M_1 = [1000]^T$、$M_2 = [0100]^T$、$M_3 = [0001]^T$、$M_4 = [0010]^T$(求得新的 M_i 的四个元素分别对应原系统的S_1、S_{23}、S_4、S_5库所)。其中 $M_0 = M_2$,则图5中节点 V_0、V_2 可以合并为一个节点 V_{02},二者之间的变迁如 t_5、t_6 省略,二者向同一外部构件的变迁如 t_1、t_2 可以合并为 t_{12},则可以快速得到新系统的可达标识图,如图6所示。其结果与先合并构件得到系统SA模型再求其基于Petri网的RMG(PSA)一致,时间复杂度为 $O(3*n)$。

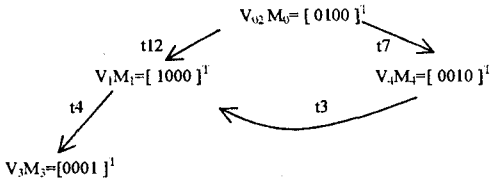

图6 合并后的RMG

(5)拆分SA中构件 C_j 成若干个构件 $C_{j1}, C_{j2}, \cdots, C_{jn}$。其过程与合并互为逆过程,也能快速求取新系统的可达标识图,从而分析新系统的各项特性。这里不再重述。

2)系统活性分析。

建立基于Petri网的SA模型除了可以定量分析构件贡献大小即演化时的波及效应外,还可以来分析所建立的系统的活性(如系统是否存在死锁)。若一个Petri网中不存在死锁,则Petri网具有活性,一个Petri网死锁时,它的标识是死标识[8]。因此,可达标识图中若有终止结点,则Petri网可能死锁。例如,图5中 $[00001]^T$ 是终止结点,所以图4中Petri网可能死锁。

除此之外,建立基于Petri网的SA模型还可以进行系统故障诊断与检测,还可以利用Petri网技术设计自校正能力的SA模型,相关理论还有待进一步探讨。

总之,利用文中提出的PSA模型建立的可达标识图RMG,可以用来完成软件架构的分析,与系统可达矩阵相比,所需的时间复杂度低,并且直观明了,易于计算。

2 结束语

运用基于PSA模型的可达标识图RMG来分析软件架构性能,一方面能给系统架构师提供系统信息参考,如构件贡献大小可以帮助合适地改变系统架构,同时也可适当地更改原有系统可达标识图快速分析变更后新系统的特性;另一方面,首次用Petri技术分析系统架构性能为软件架构研究提供了全新的一个思路。进一步的工作是利用Petri网辅助完成软件架构其他方面的设计,如辅助完成系统运行时的故障诊断与自校正设计。

参考文献:

[1] Shaw M, Garlan D. Software Architecture: Perspectives on an Emerging Discipline[M]. Upper Saddle River: Prentice Hall, 1996.

[2] 张世琨,王立福,杨芙清.基于层次消息总线的软件体系结构风格[J].中国科学(E辑),2002,32(3):393-400.

[3] Petri C A. Kommunikation mit automaten[D]. Bonn, West Germany: University of Bonn, 1962.

[4] 王映辉,张世琨,刘 瑜,等.基于可达矩阵的软件体系结构演化波及效应分析[J].软件学报,2004,15(8):1107-1115.

[5] Garlan D, Shaw M. An introduction to software architecture [R]. [s.l.]: Carnegie Mellon University, 1994.

[6] 冯 冲,江 贺,冯静芳.软件体系结构理论与实践[M].北京:人民邮电出版社,2004.

[7] Murata T. Petri nets: properties, analysis and applications[J]. Proceeding of IEEE, 1989, 77(4): 541-582.

[8] 袁崇义.Petri网原理与应用[M].北京:电子工业出版社,2005.

(上接第98页)

参考文献:

[1] Barillot C, Benali H. Federating Distributed and Heterogeneous Information Sources in Neuroimaging: The NeuroBase Project [EB/OL]. 2005. http://www.irisa.fr/visages/demo/Neurobase/Download/IRISAResearchReport1712.pdf.

[2] Josifovski V, Risch T. Query Decomposition for a Distributed Object-Oriented Mediator System[J]. Distributed and Parallel Databases, 2002, 11(3): 307-336.

[3] 袁晓洁.基于Mediation的异构数据集成系统HDIS设计与实现[J].计算机工程与应用,2006(10):162-165.

[4] Schneider R. Work Queue based multi-threading[EB/OL]. 2005. http://www.codeproject.com/csharp/workqueuethreading.asp.

[5] Microsoft. MSDN Library for Visual studio. Net 2005[DB/CD]. Microsoft corporation, 2005.

[6] 房 俊.一个基于中介机制的动态数据集成框架[J].计算机科学,2003(增刊):124-126.