软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(7):1512−1528 [doi: 10.3724/SP.J.1001.2013.04268]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种 UML 软件架构性能预测方法及其自动化研究*

李传煌, 王伟明, 施银燕

(浙江工商大学 信息与电子工程学院,浙江 杭州 310018)

通讯作者: 李传煌, E-mail: chuanhuang_li@zjgsu.edu.cn

摘 要: 软件性能需求作为软件质量需求的重要组成部分,已受到人们极大的重视.而只在软件开发周期后期才重点关注软件性能需求的传统软件开发方法,将给开发者带来高风险和高成本等后果.如果能在软件开发周期的早期对软件系统性能进行预测,可以提前发现软件系统架构存在的性能瓶颈,并找出可能的优化方案,对各种设计方案进行比较以得出最优的软件系统架构.研究了一种基于模型的 UML 软件架构性能预测方法:该方法选取软件架构设计中的 UML 用例图、活动图和构件图,并引入构造型和标记值,将它们扩展为 UML SPT 模型;进而,通过转换算法将 UML SPT 模型转换为排队网络模型,该算法可处理同时包含分支节点和汇合节点的 UML 模型活动图;最后,利用频域分析理论求解排队网络模型,以得出性能参数及性能瓶颈.同时介绍了 UML 软件架构性能自动化工具的设计方案,并给出了软件架构性能预测实例.

关键词: 性能预测;软件架构;UML;排队网络

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 李传煌,王伟明,施银燕.一种 UML 软件架构性能预测方法及其自动化研究.软件学报,2013,24(7):1512−1528. http://www.jos.org.cn/1000-9825/4268.htm

英文引用格式: Li CH, Wang WM, Shi YY. Performance prediction method for UML software architecture and its automation. Ruan Jian Xue Bao/Journal of Software, 2013,24(7):1512−1528 (in Chinese). http://www.jos.org.cn/1000-9825/4268.htm

Performance Prediction Method for UML Software Architecture and Its Automation

LI Chuan-Huang, WANG Wei-Ming, SHI Yin-Yan

(School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China)

Corresponding author: LI Chuan-Huang, E-mail: chuanhuang_li@zjgsu.edu.cn

Abstract: The requirement of software performance as an important part of the software quality requirements is very concerning. The traditional software development methods that focus on the software performance issues later in the development process will bring high risks and high costs. If the performance of software architecture can be predicted at the early phases of the development cycle, the performance bottlenecks can be found in advance, and the possible optimization also can be worked out. In this paper, a model-based UML software architectures performance prediction method is introduced. This method selects and uses case diagrams, activity diagrams and component diagrams, and extends them to UML SPT (schedulability, performance and time) model by introducing the stereotypes and tagged values. It then transforms these UML SPT models into queueing network model through an algorithm which can handle the activity diagram with both branch nodes and confluence nodes. At last, uses the analysis theory of frequency domain to solve queuing network model to derive the performance parameters and performance bottlenecks. At the same time, the design of an automatic performance analysis tool for UML software architecture is introduced, and an instance of performance prediction is given.

Key words: performance prediction; software architecture; UML; queueing network

* 基金项目: 国家重点基础研究发展计划(973)(2012CB315902); 国家自然科学基金(61102074, 61170215); 国家高技术研究发展计划(863)(SQ2009AA01XK1485130); 浙江省教育厅项目(Y201018208); 浙江省科技计划(2011C21049); 浙江省重点科技创新团队资助项目(2011R50010)

收稿时间: 2012-02-01; 定稿时间: 2012-05-29; jos 在线出版时间: 2012-07-27

CNKI 网络优先出版: 2012-07-27 11:03, http://www.cnki.net/kcms/detail/11.2560.TP.20120727.1103.003.html

软件设计产品除了应满足基本功能需求之外,还需满足质量需求.而作为软件质量重要组成部分的软件性能,得到了越来越多的关注.传统的软件开发方法通常只重视软件的功能性要求,往往在系统测试之后,才确定该设计是否能真正满足系统性能和可靠性等方面的需求.若不满足系统的性能需求,则重新设计软件系统,这将导致整个项目的成本急剧增加.如果在软件开发早期对软件模型的性能进行研究,将能降低软件开发的风险,减少开发成本,优化整个软件设计过程.如,对现有的软件架构设计方案进行定量的预测和评价,对各种设计决策进行比较从而选择更优的软件架构设计方案等.因此,在软件开发早期对系统的性能进行预测和评估,对构造满足用户性能需求的软件具有重要意义.

本文研究一种适合于 UML 软件架构系统的软件性能预测方法,该方法将系统 UML 模型扩展为 UML SPT (schedulability,performance and time)模型,通过转换算法,将 UML SPT 模型转换为排队网络模型;并依托于排队网络分析理论进行求解,以获得系统性能参数.本文的主要贡献表现在:提出一种 UML 软件架构性能自动化分析的设计方法,并提供自动化分析工具支持;研究包含分支节点和汇合节点的UML活动图转化为排队网络的方法,并在排队网络中定义该类节点的转化形式;提出一种由 *M*/*M*/1 排队系统构成的复杂开环排队网络性能参数的求解方法.

本文第 1 节介绍软件性能预测领域的相关工作.第 2 节详细叙述 UML 软件性能预测方法,主要包括软件性能预测总体方案、UML 模型的扩展、UML SPT 模型生成排队网络模型算法及排队网络模型求解方法.第 3 节介绍软件性能预测自动化工具设计方案.第 4 节给出软件性能预测实例.第 5 节为结论及下一步工作.

1 相关工作

通常,软件性能由体系结构或设计问题引起,而非低效的编码造成[1].早期的软件开发绝大部分在集成测试阶段或更晚时才引入性能问题,如今已出现一些将软件性能分析集成在软件生命周期早期的方法.它们涉及不同的说明语言和性能模型以及不同的自动化工具和性能评估环境,主要有:基于随机进程代数、随机 Petri 网方法、基于仿真的方法及基于排队网络方法[2].随机进程代数方法,如 TIPP(time processes and performability evaluation)[3],EMPA(extended Markovian process algebra)[4,5]及 PEPA(performance evaluation process algebra)[6],它们将行为与服从指数分布的随机变量相关联,并产生一个来自系统语义模型的马尔可夫链,除了指数分布的行为以外,它们也会考虑被动的和立即的行为;Petri 网是一种图形化、数学化建模工具,能够提供一个集成的建模、分析和控制环境,用来分析离散的并发系统.人们提出许多将 UML 规范和 Petri 网集成的方法,如文献[7]提出通过用例图及由协作图、状态图的组合产生通用随机 Petri 网(general stochastic Petri net,简称 GSPN),其主要思想是:将与每个协作图的对象相关联的状态图转化成 GSPN(其中,状态图中的状态和转移分别转化为 Petri 网中的库所和变迁),然后将不同的网组合成一个完整的模型.但文献[8]指出,这种转化存在重大的缺陷,即在组合不同的网时,每个状态图中的转移与其他构件是共享的;基于仿真的方法主要是针对实时系统而提出,文献[9]提出一种基于仿真模型的方法,使用扩展 UML 图来描述当前的需求和资源使用情况,这种扩展使用了构造型、标记值和约束.Arief 和 Speirs 在文献[10]中提出了另一种基于仿真模型的方法,他们描述了一种仿真框架,称为仿真模型语言,通过实现面向过程的仿真模型的系统 UML 规范,可以自动产生一个仿真的 Java 程序,SimML 允许用户利用用例图和顺序图及说明自动产生仿真模型所必须的信息;基于排队网络(queueing network,简称 QN)模型的方法是目前性能预测中应用较为广泛的一种分析模型,支持一些工具和算法,还可以扩展为扩展排队网络(extended QN,简称 EQN)[11]和分层排队网络(layered QN,简称 LQN)[12].

研究者们根据 UML 不同类型图的特点,选取其中的几类图,基于图转换提出了多种从 UML 描述到排队网络模型的方法.文献[13,14]提出从 UML 协作图、UML 活动图和 UML 部署图导出 LQN 性能模型的方法.文献[15]基于 UML 顺序图的软件性能测试方法进行研究,将顺序图转化为活动执行图,再分析活动执行图的性能,并提出相应的转化算法.文献[16]将 UML 用例图、活动图和部署图添加性能标签,转化为排队网络模型,再用目前成熟的性能分析方法求解排队网络性能,该方法受限于当前的排队网络性能分析方法,只能处理UML活动图中没有分支节点和汇合节点的情况.文献[17]类似于文献[16],也将 UML 用例图、活动图和部署图添加性能标

签,但将它转化为仿真模型,通过仿真的方法进行排队网络的性能求解.该方法考虑了 UML 活动图中分支节点和汇合节点的情况,但需要仿真软件的支持,算法复杂度较高.本文提出的 UML 生成排队网络模型算法也是一种基于图转换的方法,将 UML 用例图、活动图和构件图添加性能标签,转化为排队网络模型,同时考虑了 UML 活动图中有分支节点和汇合节点的情况.

根据到达顾客形式的不同,排队网络模型可分为 3 种基本类型[18]:开环、闭环和混合网络,其典型代表分别是 Jackson 网络、Gordon-Newell 网络和 BCMP 网络,这些网络平衡状态条件下的解都满足乘积形式[19].这种乘积形式解的特殊性,使得排队网络模型的研究越来越广泛.研究者们提出一系列求解具有乘积形式解的排队网络的方法,比较著名的有平均值分析法、归一化常数[18].同时,也在不断探求排队网络满足乘积形式解的条件,并在特定场合下扩展这些条件,如文献[20]为转移状态与转移概率相关的模型提供了一种乘积形式解.乘积形式解的计算复杂度会随着顾客类别及服务中心的增多而急剧增加,于是,人们开始寻求其近似解法[18,21],在计算复杂度与结果准确性之间取得相对平衡.文献[18]中指出一些近似求解方法,如马尔可夫过程的分解、排队网络的分解、矩阵几何方法.文献[22]将 Pentium 处理器建模为排队网络,根据排队网络中服务中心之间的串联和并联情况等价为一个新的排队系统,通过不断的迭代等价,最终将整个排队网络等价为一个单队列单服务器模型,进而求出性能.但文献[22]中没有给出并联和串联的具体划分标准,只是作为一个例子给出了这种方法的实施步骤,缺乏科学性.本文提出的排队网络的性能求解方法也是一种类似等价求解的方法,是一种近似的求解方法.与上述方法不同,本文创新性地提出一种在各服务中心相互独立的条件下,利用信号与系统的频域分析来求解性能近似解的方法.

2 UML 软件性能预测方法研究

2.1 UML软件性能预测总体方案

本文提出的性能预测方法基于自动化分析工具来完成,具体方案如图 1 所示,共 8 个步骤:

- 步骤 1:用户指定系统所需满足的性能参数指标,这些性能指标将作为判断利用 UML 软件架构性能自动化工具进行系统性能预测得到的性能参数是否满足系统性能要求的标准.本文讨论的性能参数指标包括系统的平均系统顾客数、平均系统时间和平均等待时间.这些性能参数指标是评价一个系统运行状况的主要指标,其中,平均系统顾客数越少、平均系统时间越短、平均系统等待时间越短,系统的性能越好;
- 步骤 2:用户建立软件系统的 UML 模型.本文选择用例图、活动图和构件图作为性能预测的 UML 图形.用户可以选择 violetUML 软件建立系统的 UML 用例图、活动图和构件图;
- 步骤 3:用户建立软件系统的 UML SPT 模型.用户根据步骤 2 中建立的 UML 模型,加入构造型和标记值转为 UML SPT 模型,用于后期转为排队网络;
- 步骤 4:用户根据经验或测量给定标记值,这些标记值将成为排队网络参数;
- 步骤 5:将加标记值的 UML SPT 模型转为程序可以识别的 XML 格式文件;
- 步骤 6:解析得到的 XML 格式文件,利用 UML SPT 模型生成排队网络模型算法转化为排队网络.本文仅限于开环排队网络的转换;
- 步骤 7:分析步骤 6 获得的排队网络,求解排队网络的性能参数值;
- 步骤 8:将步骤 7 求得的性能参数值与步骤 1 中用户指定的性能参数指标进行比较,如果满足用户性能需求,则结束性能预测过程;否则,用户修改性能参数指标或者更改系统的软件设计,再次按照上述步骤依次执行,直到满足用户性能需求为止.

从图 1 中可以看出,步骤 5~步骤 7 是通过 UML 软件架构性能自动化工具完成的,无需用户参与.由此可知,用户不需要了解如何将 UML SPT 模型转化为 XML 格式文件,也不需要了解如何将这些文件转化为排队网络,更不需要了解如何求解排队网络获得性能参数值,只需建立软件系统的 UML 模型,并加入构造型和标记值,使之成为 UML SPT 模型,利用 UML 软件架构性能自动化工具即可得到系统的性能参数值,评定是否满足用户的

系统性能需求.

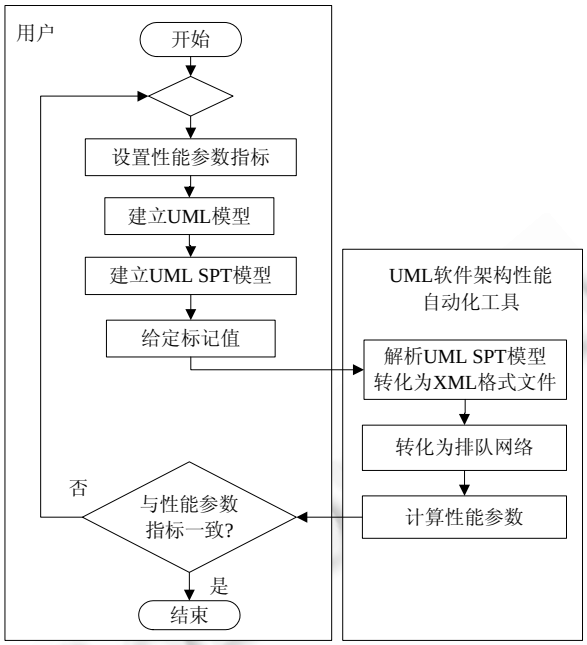

Fig.1 Procedure of performance prediction

图 1 软件性能预测的步骤

2.2 UML模型转化为UML SPT性能模型

在对软件系统进行性能预测时,选择 UML 用例图、活动图和构件图作为输入,在这 3 种 UML 图的基础上加入构造型和标记值,分别得到带标注的用例图、活动图和构件图.见表 1.

Table 1 The extension from UML model to UML SPT model

表 1 UML 模型转化为 UML SPT 模型的构造型和标记值

UML 模型	UML SPT 模型		
	构造型	标记值	含义
用例图	《PAopenuser》	*PAarrival*=[*PDFstring*,*value*] *PDFstring*={*exponential*,*uniform*, *formal*,*const*,…}	《PAopenuser》为操作者的构造型, *PAarrival* 为标记值,描述操作者到达系统的规律, 包括服从的分布和相应的分布参数
构件图	《PAresource》	*PArate* *PAschedpolicy*={*FIFO*,*PS*,…} *PAbfork*	《PAresource》为资源节点的构造型,*PArate*, *PAschedpolicy* 和 *PAbfork* 为标记值,分别描述资源 节点的服务速率、调度策略和占有的最大分支节点号
活动图	《PAaction》	*PAresource* *PAdemand*={*exponential*,*uniform*, *formal*,*const*,…}	《PAaction》为活动状态的构造型, *PAresource* 和 *Pademand* 为标记值, 分别描述活动状态请求的资源节点和请求方式
	《PAtransprob》	*PAprob*	《PAtransprob》为分支节点的构造型, *PAprob* 为标记值,描述分支概率
	《PAjoin》	*PAjname*={*join*1,*join*2,…,*joinM*}	《PAjoin》为汇合节点的构造型, *PAjname* 为标记值,描述汇合节点的名称
	《PAfork》	*PAfname*={*fork*1,*fork*2,…,*forkN*}	《PAfork》为分支节点的构造型, *PAfname* 为标记值,描述分支节点的名称

用例图从用户角度描述系统功能,并指出各功能的操作者.操作者使用系统特定功能后可能离开系统,也可能不离开系统.对于这两种情况,分别用构造型《PAopenuser》和《PAcloseduser》来描述操作者.本方案只考虑使用系统功能后离开系统的操作者.使用标记值 *PAarrival* 来描述《PAopenuser》操作者访问系统的情况.

PAarrival 描述操作者到达系统的规律,这是一个随机变量,包含两个元素:*PDFstring* 和 *value*,其中,*PDFstring* 代表所服从的分布,如指数分布.*value* 代表所服从分布的参数,如指数分布的参数λ.

构件图描述代码构件的物理结构及各构件之间的依赖关系.构件图中的每一个资源节点使用构造型《PAresource》描述,同时用标记值 *PArate* 描述资源节点的平均处理速率,用标记值 *PAschedpolicy* 描述资源节点的处理方式,如先进先出方式,用标记值 *PAbfork* 描述资源节点被分支节点衍生出来的活动状态请求的信息,若没有被分支节点衍生出来的活动状态所请求,则该标记值为 0.

活动图描述了为满足用例要求所要进行的各类活动以及活动间的约束关系,故活动图与特定的用例图相关联.活动图中的汇合节点用构造型《PAjoin》描述,同时用标记值 *PAjname* 描述该汇合节点的名称,如 *join*1,*join*2,…汇合节点名称后序号越大,代表该汇合节点号越大;分支节点用构造型《PAfork》描述,同时用标记值 *PAfname* 描述该分支节点的名称,如 *fork*1,*fork*2,…分支节点名称后序号越大,分支节点号就越大.如果分支节点后有汇合节点,那么两者在一定条件下可以进行匹配,匹配的条件就是分支节点和汇合节点拥有相同的分支.如果有一个汇合节点与第 1 个分支节点匹配,那么第 1 个分支节点之前和与之匹配的汇合节点之后的活动状态所请求的资源节点的标记值 *PAbfork* 为 0;分支节点用构造型《PAtransprob》描述,同时用标记值 *PAprob* 描述分支概率;活动状态用构造型《PAaction》描述,同时用标记值 *PAresource* 描述活动所请求的节点资源,用标记值 *PAdemand* 描述以怎样的规律来请求节点资源,如指数分布.

2.3 UML SPT模型生成排队网络模型的算法

将 UML SPT 模型中的元素映射到排队网络模型中的元素,其对应关系为:UML SPT 模型中带标注的构件图的每一个资源节点映射为排队网络中的服务节点;UML SPT 模型中带标注的用例图的构造型《PAopenuser》和《PAcloseduser》分别映射为开环和闭环的排队网络,即该构造型指定排队网络的外部到达情况;UML SPT 模型中带标注的活动图中活动状态的转移情况映射为排队网络的拓扑,即指定顾客到达排队网络后如何通过整个排队网络的过程;UML SPT 模型中的标记值映射为排队网络模型中的参数,带标注的用例图中的标记值 *PAarrival* 指定到达系统的规律,带标注的活动图中的 *PAdemand* 指定请求的资源节点服从什么样的分布,带标注的构件图中的 *PArate* 指定服务的平均服务速率,*PAschedpolicy* 指定调度的策略.

一个带标注的 UML 活动图记为一个 $G=(A,T,F)$的有向图集合,其中,

- $A=\{a_0,a_1,a_2,\ldots\}$代表活动图 G 的活动状态的集合,a_0 为活动图 G 的初始节点和结束节点;
- $T=\{t_1,t_2,t_3,\ldots\}$代表活动图 G 的活动状态的转移集合,t_i 表示一对活动状态转移(a_i,a_j),$1\leqslant i,j\leqslant|A|$,即活动状态 a_i 完成后立即执行活动状态 a_j.由于当一个活动状态完成后可能会执行一个或多个活动状态,令 $p(t)\in(0,1)$为活动状态转移 t 发生的概率,即从活动状态 a_i 转移到活动状态 a_j 的概率,此概率在活动图中用标记值 *PAprob* 表示;
- $F=\{f_1,f_2,f_3,\ldots,f_N\}$代表活动图 G 的分支节点的集合,f_i 表示第 i 个分支节点,并为每个分支节点 f_i 分配唯一的标识$[1,2,\ldots,N]$,用属性 $id[\cdot]$表示分支节点 f_i 的标识,则有 $id[f_i]=i,1\leqslant i\leqslant N$.

将带标注的 UML 构件图记为一个 $R=\{r_1,r_2,r_3,\ldots,r_K\}$集合,$r_i\in R$ 为构件图 R 的资源节点,并为每个资源节点 r_i 分配唯一的标识$[1,2,\ldots,K]$.用属性 $id[\cdot]$表示资源节点 r_i 的标识$[1,2,\ldots,K]$,则有 $id[r_i]=i$,用属性 $bfork[\cdot]$来表示资源节点 r_i 被分支节点 f_i 衍生出的活动状态请求,$bfork[r_i]=id[f_j]=j$,若资源节点 r_i 没有被分支节点衍生出的活动状态 a_i 请求,则有 $bfork[r_i]=0$,$bfork[\cdot]$在带标注的构件图中用标记值 *PAfork* 表示.用 $res[a_i](1\leqslant i\leqslant|A|)$表示带标注的 UML 活动图中的每个活动状态 a_i 请求构件图中的资源节点 $r_j(1\leqslant j\leqslant K)$,则有 $res[a_i]=r_j,1\leqslant i\leqslant|A|,1\leqslant j\leqslant K$.

从带标注的 UML 用例图的构造型《PAopenuser》可知,转换出来的排队网络是一个开环的排队网络,标记值 *PAarrival* 描述到达该排队网络的一个外部输入情况,包括服从什么分布以及相应的参数.

带标注的 UML 活动图中请求带标注的构件图中资源节点的活动状态转化为排队网络中请求服务台的作业,由于不同的活动状态可能请求同一个资源节点,故需要区分请求同一个资源节点的不同活动状态.用 $res[a_i](1\leqslant i\leqslant|A|)$代表活动状态 a_i 请求的资源节点 r_j,用 $count[r_j](1\leqslant j\leqslant K)$代表资源节点 r_j 被请求的活动状态的总数,于是,资源节点 r_j 被请求的活动状态可用$[1,2,\ldots,count[r_j]]$标识,定义 $index[a_i]$保存活动状态 a_i 请求资源节点 r_j

时属于标识[1,2,…,$count[r_j]$]中的第几个.因此,UML 活动图中的活动状态 a_i 请求构件图中的资源节点 r_j 就转化为排队网络中第 $index[a_i]$类作业向服务台 $res[a_i]=r_j(1\leqslant i\leqslant|A|,1\leqslant j\leqslant K)$请求服务.定义路由矩阵 $P=P[i,r,j,s]=P(t)$ 表示第 i 个服务台的第 r 类顾客完成服务后进入第 j 个服务台的第 s 类的概率.

考虑到带标注的UML 活动图中包含分支节点,它将触发之后的活动状态同时发生,这些被同时触发的活动状态所请求的资源节点 r_j 在带标注的 UML 构件图中用标记值 *PAfork* 表示.在排队网络中,用 $bfork[r_i]$表示资源节点 r_j 被分支节点衍生的活动状态中 *PAfork* 的 id 号最大值.活动图中的分支节点意味着并发进行,并发后的资源节点相互独立,将并发的资源节点用虚框标识,它代表一个并发的过程,活动图中有几个并发的过程,就用几个虚框标识.

UML SPT 模型生成开环排队网络模型的算法如下所示:

1. Require: Activity diagram $G=(A,T,F)$, Component diagram R
2. for ($r_i=r_1$; $r_i\leqslant r_M$; r_{i++}) do
3. $count[r_i]=0$
4. for ($a_i=a_1$; $a_i\leqslant a_{|A|}$; a_{i++}) do
5. {
6. $count[res[a_i]]=count[res[a_i]]+1$
7. $index[a_i]=count[res[a_i]]$
8. }
9. $C=\max_{r\in R}\{count[res[a_i]]\}$ {Number of Classes in G}
10. $K=|R|$ {Number of Service Centers in G}
11. $M=|A|$ {Number of Actions in G}
12. $P=P[i,r,j,s]=new\ \ vector[K\times C\times K\times C]$
13. for all $t_i=(x,y)\in T$ such that $x\neq a_0$ and $y\neq a_0$ do
14. {
15. $i=id[res[x]]$
16. $r=index[x]$
17. $j=id[res[y]]$
18. $s=index[y]$
19. }
20. $P[i,r,j,s]=P(t)$
21. for (x=1;$x\leqslant\max\{bfork[r_i]\}$; x++) do
22. for all $r_i\in R$ do
23. if ($bfork[r_i]==x$)
24. $visualbox(r_i)$

2.4 排队网络模型及其性能参数求解方法

本节针对第 2.3 节中生成的开环排队网络模型探寻求解方法.设生成的排队网络为开环无反馈网络,各系统为 $M/M/1$ 排队系统,且相互独立,排队网络性能参数包括平均系统顾客数、平均系统时间和平均等待时间.根据 UML 模型中 UML 活动图中是否包含分支,将转换后的排队网络分为普通排队网络和含并发处理的排队网络,下面分别就这两种排队网络进行建模,并研究其性能参数的求解方法.

2.4.1 普通排队网络模型及性能参数求解方法

普通排队网络是指由不包含分支节点的 UML 活动图的 UML 模型转换之后的排队网络,是不含并发处理的排队网络.由相互独立的 $M/M/1$ 排队系统组成的开环无反馈的普通排队网络可以看成是一个或多个 $M/M/1$ 排队系统的串联或并联的组合.

考虑一般性,将普通开环排队网络模型建模,如图 2 所示,该排队网络由 5 个 $M/M/1$ 排队系统组成,外部到达系统的到达率为λ,第 T_i 个 $M/M/1$ 排队系统的服务速率为μ_i.从总体来看,这是一个 3 级串联型排队网络,其中,第 1 级和第 3 级分别为一个单独的 $M/M/1$ 排队系统 T_1 和 T_5;第 2 级是一个并联网络的组合,即串联中包含了并联,它以概率 h_1 进入 T_2,以概率 h_2 进入 T_4,在这个并联网络中,同时包含 T_2 与 T_3 串联.

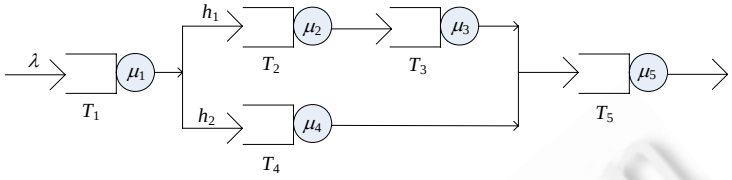

Fig.2 The general open-loop queueing network

图 2 简单形式的普通开环排队网络模型

对于复杂形式的普通排队网络,只是增加了串联的级数和并联的级数以及串联和并联嵌套的层数.

- 平均系统顾客数 $E\{N\}$

平均系统顾客数 $E\{N\}$是指系统处于平稳状态时在系统中的顾客总数.Bruke 在文献[23]中证明离开 Poisson 到达,指数服务系统的分布与顾客到达此系统的分布相同,都是参数为λ的 Poisson 分布.也就是说,图 2 所示的普通开环排队网络模型的平均系统顾客数相当于 5 个独立 $M/M/1$ 排队系统的平均系统顾客数累加的和.故,平均系统顾客数为

$$E\{N\}=\sum_{i=1}^{5}E\{N_i\} \tag{1}$$

其中,$E\{N_i\}$是第 T_i 个 $M/M/1$ 排队系统的平均系统顾客数,而

$$E\{N_i\}=\rho_i/(1-\rho_i) \tag{2}$$

其中,$\rho_i=\lambda_i/\mu_i$.

- 平均系统时间 $E\{S\}$

平均系统时间 $E\{S\}$是指系统处于平稳状态时顾客从到达到离开系统所花的时间,根据 little 定理,排队系统的平均系统顾客数 n、平均到达率λ和平均系统时间 s 满足 $n=\lambda s$.由于本文所求的排队网络只有一个外部到达,故该到达率λ就是平均到达率λ,故平均系统时间 $E\{S\}$为

$$E\{S\}=E\{N\}/\lambda \tag{3}$$

- 平均等待时间 $E\{W\}$

平均等待时间 $E\{W\}$是指系统处于平稳状态时顾客从到达到开始接受服务所需时间,而系统处于平稳状态时顾客从开始接受服务到离开系统时所需时间称为平均服务时间 $E\{T\}$,它们与平均系统时间 $E\{S\}$存在如下关系:

$$E\{W\}=E\{S\}-E\{T\} \tag{4}$$

设第 T_i 个 $M/M/1$ 排队系统的服务时间为随机变量τ_i,则该随机变量τ_i服从参数为μ_i的指数分布.故,第 T_i 个 $M/M/1$ 排队系统服务时间的概率密度函数 $f_{\tau_i}(x)$ 为

$$f_{\tau_i}(x)=\mu_i e^{-\mu_i x} \tag{5}$$

若 M 个独立的 $M/M/1$ 排队系统的平均服务时间的概率密度函数为 $f_i(x)$,由系统的并联和串联特性可知,这 M 个 $M/M/1$ 排队系统串联的平均服务时间的概率密度函数为 $f_1(x)*f_2(x)*\ldots*f_M(x)$,其中,*代表卷积,这 M 个 $M/M/1$ 排队系统并联的平均服务时间的概率密度函数为 $f_1(x)+f_2(x)+\ldots+f_M(x)$.

结合串联特性和并联特性,图 2 所示的普通开环排队网络的平均服务时间的概率密度函数 $f_\tau(x)$为

$$f_\tau(x)=f_{\tau_1}(x)*(h_1 f_{\tau_2}(x)*f_{\tau_3}(x)+h_2 f_{\tau_4}(x))*f_{\tau_5}(x) \tag{6}$$

对式(6)进行 s 域分析,先求出第 T_i 个 $M/M/1$ 排队系统服务时间的概率密度函数的 Laplace 变换,依据 Laplace

变换的时域卷积定理,可将时域卷积转换为 s 域的乘积,最后通过 Laplace 逆变换求出普通开环排队网络的服务时间的概率密度函数,由此求得概率密度函数为

$$f_\tau(x)=\mu_1\mu_5\left\{\frac{h_1\mu_2\mu_3\mu_4+h_2\mu_2\mu_3\mu_4-h_1\mu_1\mu_2\mu_3-h_2\mu_1\mu_2\mu_4-h_2\mu_1\mu_3\mu_4+h_2\mu_4\mu_1^2}{(\mu_5-\mu_1)(\mu_4-\mu_1)(\mu_3-\mu_1)(\mu_2-\mu_1)}\mathrm{e}^{-\mu_1 x}+\right.$$
$$\frac{h_1\mu_2\mu_3}{(\mu_5-\mu_2)(\mu_3-\mu_2)(\mu_1-\mu_2)}\mathrm{e}^{-\mu_2 x}+$$
$$\frac{h_1\mu_2\mu_3}{(\mu_5-\mu_3)(\mu_2-\mu_3)(\mu_1-\mu_3)}\mathrm{e}^{-\mu_3 x}+ \tag{7}$$
$$\frac{h_2\mu_4}{(\mu_5-\mu_4)(\mu_1-\mu_4)}\mathrm{e}^{-\mu_4 x}+$$
$$\left.\frac{h_1\mu_2\mu_3\mu_4-h_2\mu_2\mu_4\mu_5+h_2\mu_2\mu_3\mu_4-h_2\mu_3\mu_4\mu_5-h_1\mu_2\mu_3\mu_5+h_2\mu_4\mu_5^2}{(\mu_4-\mu_5)(\mu_3-\mu_5)(\mu_2-\mu_5)(\mu_1-\mu_5)}\mathrm{e}^{-\mu_5 x}\right\}$$

根据数学期望的定义,平均服务时间 $E\{T\}$为

$$E\{T\}=\int_0^\infty xf_\tau(x)\mathrm{d}x \tag{8}$$

将公式(7)代入公式(8),得到平均服务时间 $E\{T\}$为

$$E\{T\}=\int_0^\infty xf_\tau(x)\mathrm{d}x=\mu_1\mu_5\left\{\frac{h_1\mu_2\mu_3\mu_4+h_2\mu_2\mu_3\mu_4-h_1\mu_1\mu_2\mu_3-h_2\mu_1\mu_2\mu_4-h_2\mu_1\mu_3\mu_4+h_2\mu_4\mu_1^2}{(\mu_5-\mu_1)(\mu_4-\mu_1)(\mu_3-\mu_1)(\mu_2-\mu_1)\mu_1^2}+\right.$$
$$\frac{h_1\mu_3}{(\mu_5-\mu_2)(\mu_3-\mu_2)(\mu_1-\mu_2)\mu_2}+$$
$$\frac{h_1\mu_2}{(\mu_5-\mu_3)(\mu_2-\mu_3)(\mu_1-\mu_3)\mu_3}+ \tag{9}$$
$$\frac{h_2}{(\mu_5-\mu_4)(\mu_1-\mu_4)\mu_4}+$$
$$\left.\frac{h_1\mu_2\mu_3\mu_4-h_2\mu_2\mu_4\mu_5+h_2\mu_2\mu_3\mu_4-h_2\mu_3\mu_4\mu_5-h_1\mu_2\mu_3\mu_5+h_2\mu_4\mu_5^2}{(\mu_4-\mu_5)(\mu_3-\mu_5)(\mu_2-\mu_5)(\mu_1-\mu_5)\mu_5^2}\right\}$$

将公式(3)、公式(9)代入公式(4),可求出图 2 所示的普通开环排队网络的平均等待时间.

2.4.2 含并发处理的排队网络模型及性能参数求解方法

含并发处理的排队网络是指由有分支节点的 UML 活动图的 UML 模型转换之后的排队网络.并发处理部分用虚框表示,即虚框内代表 UML 活动图中分支节点和汇合节点之间的并发行为,其中的每一条并发分支都代表 UML 活动图中由分支节点引出的活动路径,并且必须执行,是完整系统功能不可分割的一部分,但是每条并发分支可以同时执行.

图 3 为含并发处理的排队网络模型,它由 7 个相互独立的 $M/M/1$ 排队系统组成,外部到达该网络的到达率为λ.

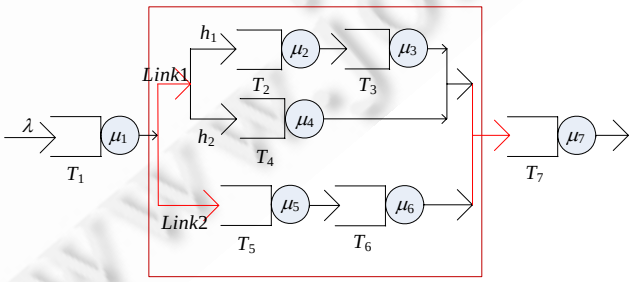

Fig.3 The open-loop queueing network including parallel processing

图 3 含并发处理的开环排队网络模型

从图 3 可以看出,虚框内有两条独立的链路 *Link*1 和 *Link*2,即有两个并发分支,顾客从 T_1 离开后,可同时得到 *Link*1 及 *Link*2 的服务.每一条链路可以看成是一个普通的串联排队网络模型,如:*Link*1 串联网络为只有 1 级并联网络的组合,顾客以概率 h_1 进入 T_2,以概率 h_2 进入 T_4,而在这个并联网络中,又包含了 T_2 与 T_3 串联;*Link*2 串联网络由 2 级简单串联网络组成,分别为 *M*/*M*/1 排队系统的 T_5 和 T_6.当顾客到达时,这两条链路 *Link*1 和 *Link*2 都必须执行,但执行无先后顺序关系,可以看成该顾客的两个相同的副本同时通过链路.这也是和概率分支的区别,概率分支只以一定的概率进入某一分支,一旦执行某一分支,其他所有分支将不会执行.

复杂形式的含并发处理的排队网络模型,只是虚框内各个分支是复杂形式的普通排队网络模型;或是虚框内的分支数有多条;或是有多个独立的虚框;或是各个分支又嵌套了另一个虚框.

接下来将求解含并发处理的开环排队网络模型的性能参数,主要包括平均系统顾客数、平均系统时间和平均等待时间,另外还包括单个 *M*/*M*/1 排队系统的利用率.

- 平均系统顾客数 $E\{N\}$

对于如图 3 所示的含并发处理的开环排队网络模型来说,由于各个 *M*/*M*/1 排队系统之间的相互独立性,可认为其平均系统顾客数 $E\{N\}$包括虚框内部的排队网络的平均系统顾客数 $E_{IV}\{N\}$和虚框外部的排队网络的平均系统顾客数 $E_{OV}\{N\}$,即平均系统顾客数为

$$E\{N\}=E_{IV}\{N\}+E_{OV}\{N\} \tag{10}$$

虚框外部的排队网络是普通的排队网络模型,由公式(1)可知,其平均系统顾客数为

$$E_{OV}\{N\}=E_1\{N\}+E_7\{N\} \tag{11}$$

其中,$E_1\{N\}=\rho_1/(1-\rho_1)$,$E_7\{N\}=\rho_7/(1-\rho_7)$,$\rho_1=\lambda/\mu_1$,$\rho_7=\lambda/\mu_7$.

分析图 3 所示虚框内部两条链路 *Link*1 和 *Link*2,由于顾客到达后两条链路都必须执行,且两条链路并发执行,故虚框内部平均系统顾客数等于两链路中平均系统顾客数中最大的一个,即虚框内部的平均系统顾客数为

$$E_{IV}\{N\}=\max\{E_{Link1}\{N\},E_{Link2}\{N\}\} \tag{12}$$

而 *Link*1 和 *Link*2 链路是普通排队网络,故可以用普通排队网络模型的平均系统顾客数的求解方法求解.

复杂形式的含并发处理的开环排队网络模型的平均系统顾客数的求解方法与简单形式的含并发处理的开环排队网络模型的求解方法类似.

- 平均系统时间 $E\{S\}$

含并发处理的排队网络模型的平均系统时间的求解方法与普通排队网络模型的求解方法一样,用公式(3)求解,其中,$E\{N\}$是上述公式(10)求出的平均系统顾客数,λ为平均到达率.

- 平均等待时间 $E\{W\}$

平均等待时间的求解方法是先将其等价为普通的排队网络模型,然后按照普通排队网络模型的求解方法进行求解.因此,求解含并发处理的排队网络模型的平均等待时间的关键是如何将其转换为普通的排队网络模型,即如何去掉虚框.

由并发概念可知,图 3 所示中顾客在虚框内部的平均等待时间为顾客在 *Link*1 和 *Link*2 中等待时间的最大值,即虚框内部的平均等待时间为

$$E_{IV}\{W\}=\max\{E_{IV1}\{W\},E_{IV2}\{W\}\} \tag{13}$$

其中,$E_{IV1}\{W\}$,$E_{IV2}\{W\}$为虚框内链路 *Link*1 和 *Link*2 的平均等待时间,其求解方法用普通排队网络模型的平均等待时间的求解方法求解.

复杂形式的含并发处理的开环排队网络模型的平均系统顾客数的求解方法与简单形式的含并发处理的开环排队网络模型的求解方法类似,先将虚框内的排队网络从内而外等效为普通的排队网络模型,再利用普通排队网络模型的平均等待时间的求解方法进行求解.而这里的等效是指使用虚框内最大平均等待时间的链路加以替代.

3 UML 软件架构性能自动化分析工具

UML 软件架构性能自动化分析工具是在软件开发早期对软件系统架构性能进行自动化分析的工具,它可以自动地将软件系统的 UML 活动图、用例图和构件图转化为排队网络模型,同时计算出软件系统的性能指标,提供给用户并找出软件系统的性能瓶颈.

UML 软件架构性能自动化分析工具为用户提供一个操作界面,采用面向对象语言 C++来实现.图 4 所示为工具的实现总体方案,工具的输入为由 Violet UML[24]绘制的 XML 格式的带标注的 UML 活动图、构件图和用例图;输出为排队网络模型图片和软件系统的性能指标参数值,包括平均系统顾客数、平均系统时间、平均等待时间和单个排队网络节点的利用率.工具利用 Graphivz[25]绘制图形,利用 OpenCV[26]显示图形,利用 Matlab[27]计算排队网络的性能参数,故其应用环境应提供对 Graphivz,OpenCV 和 Matlab 的支持.

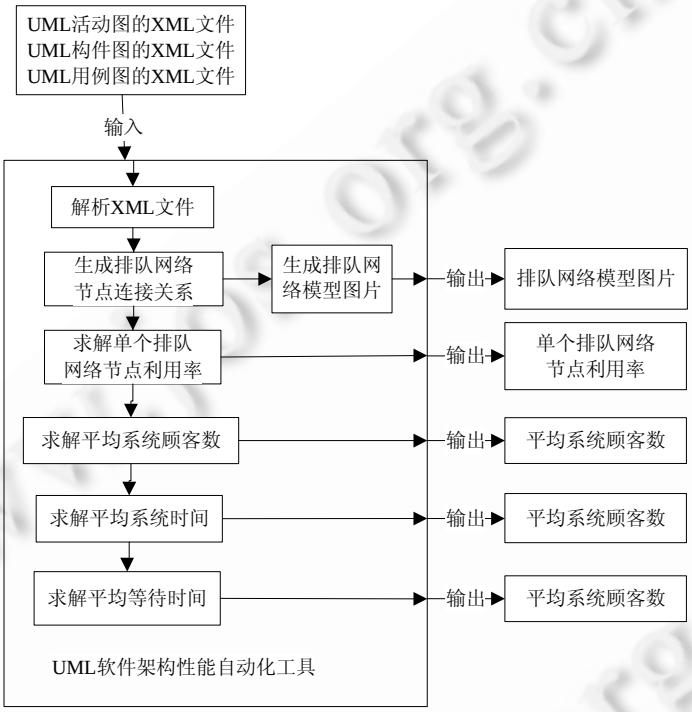

Fig.4 The implementation scheme for the automatic analysis tool

图 4 UML 软件架构性能自动化分析工具的实现总体方案

UML 软件架构性能自动化分析工具类图如图 5 所示.

XMLParseUseCaseDiagram 类:UML 用例图文件解析类,主要包括 UML 用例图文件名和用例节点信息,其中,UML 用例图文件名用 string 类型的 m_useCaseDiagFileName 存储,用例节点信息用结构体 TXMLUseCaseNode 存储.该类的主要作用是提取 XML 格式的带标注的 UML 用例图中的有用信息,将其存储到用例节点信息结构体中,提供与该用例相关的活动图、转换后的排队网络的类型和外部到达排队网络的到达分布及参数.

XMLParseComponentDiagram 类:UML 构件图文件解析类,主要包括 UML 构件图文件名和构件资源节点信息,其中,UML 构件图文件名用 string 类型的 m_compDiagFileName 存储,构件资源节点信息用结构体 TXMLCompNode 存储.该类的主要作用是提取 XML 格式的带标注的 UML 构件图中的有用信息,将其存储到构件资源节点信息结构体中,为后续生成排队网络模型和计算性能指标作准备.

XMLParseActivityDiagram 类:UML 活动图文件解析类,主要包括 UML 活动图文件名、活动图节点信息和

节点转移信息,其中,UML 活动图文件名用 string 类型的 m_activityDiagFileName 存储,活动图节点信息包括活动状态节点信息、开始结束节点信息、同步条节点信息和决策节点信息,分别用结构体 TXMLActivityNode, TXMLStartEndNode,TXMLSynBarNode 和 TXMLDecisionNode 存储,节点转移信息用结构体 TnodeTransfer 存储.该类的主要作用是提取 XML 格式的带标注的 UML 活动图中的有用信息,将其存储到活动图节点信息相应的结构体中,提取 XML 格式的带标注的 UML 活动图中节点之间的连接关系,将其存储到节点转移信息结构体中,为后续生成排队网络模型和计算性能指标作准备.

UMLTransToQN 类:UML 生成排队网络模型类,它关联 XMLParseUseCaseDiagram 类、XMLParseComponentDiagram 类和 XMLParseActivityDiagram 类,包括 UML 用例图文件名、UML 构件图文件名、UML 活动图文件名和节点转移信息,其中,UML 用例图文件名、UML 构件图文件名和 UML 活动图文件名分别用 string 类型的 m_useCaseDiagFileName,m_compDiagFileName 和 m_activityDiagFileName 存储,节点转移信息用结构体 Tnode Transfer 存储.该类的主要作用是用 UML SPT 模型生成排队网络模型算法将 XMLParseUseCaseDiagram 类、XMLParseComponentDiagram 类和 XMLParseActivityDiagram 类中生成的信息生成排队网络节点之间的连接关系,然后将排队网络节点之间的连接关系按照 dot 语言的格式写入 dot 文件,利用该文件通过软件 Graphivz 绘制排队网络模型图片,最后利用 OpenCV 显示生成的排队网络模型图.

CompPerformanceParam 类:性能指标求解类,它是 UMLTransToQN 类的泛化,包括平均系统顾客数、平均系统时间、平均等待时间、平均服务时间及其 *s* 域表达式,其中,平均系统顾客数、平均系统时间、平均等待时间和平均服务时间分别用 float 类型的 m_AveSysCustomNum,m_AveSysTime,m_AveWaitTime 和 m_AveServTime 存储,平均服务时间的 *s* 域表达式用 string 类型的 m_equation 存储.该类的主要作用是首先计算各个排队网络节点的访问率,然后计算单个排队网络节点的利用率、系统的平均顾客数、平均系统时间、平均服务时间和平均等待时间.

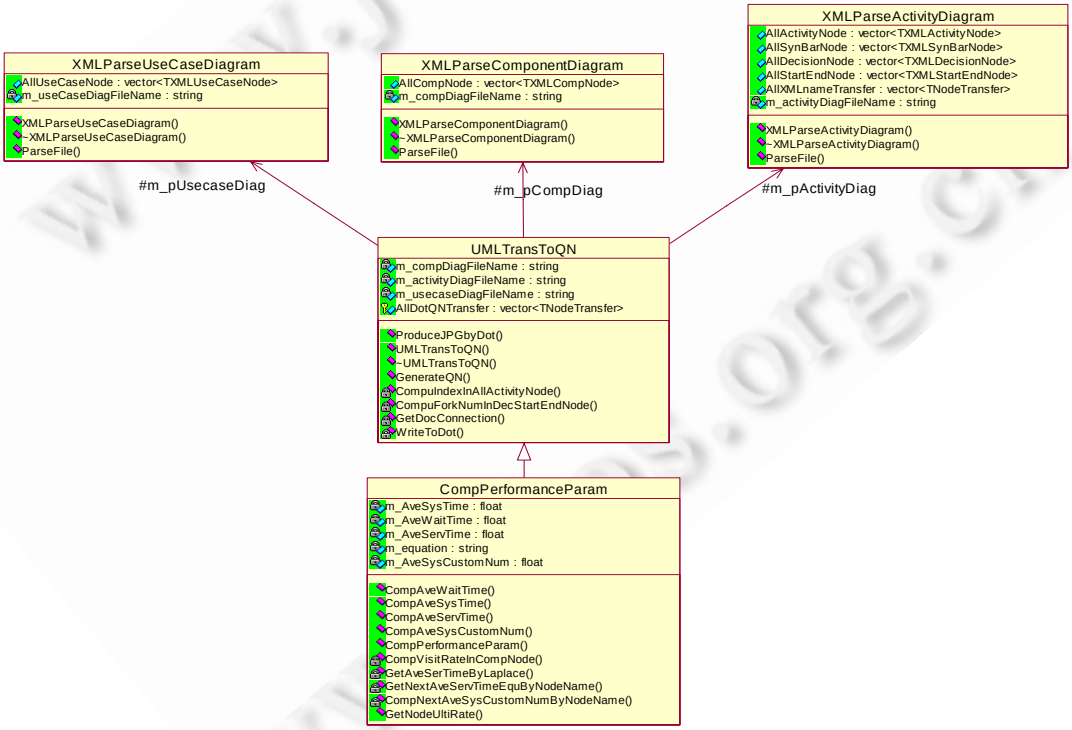

Fig.5 The class diagram for the automatic analysis tool

图 5 UML 软件架构性能自动化分析工具类图

4 软件架构性能预测实例分析

4.1 实例概述及性能需求

以我们参与的“863”计划重大专项:可重构柔性实验网组网设备工程化实施(SQ2009AA01XK1485130)、可重构柔性实验网中的可重构路由交换平台管理子系统为例,该子系统是实验网中可重构综合管理系统与可重构路由设备之间的桥梁,它将可重构综合管理系统下发的嵌入式逻辑承载网构建粗颗粒命令转换为可重构路由设备的加载及配置命令,同时将收集的可重构路由设备的信息发送给可重构综合管理系统.

可重构路由交换平台管理子系统除了满足上述功能上的需求之外,还需满足性能需求.为此,制定了可重构路由交换平台管理子系统的预期性能指标,在可重构综合管理系统下发构建命令满足参数为 220 的 Poisson 分布时,其平均系统顾客数不大于 10,平均系统时间不大于 0.5,平均等待时间不大于 0.1.

4.2 初始软件架构设计及性能预测

根据子系统应用场合,将可重构路由设备具体化为两台可重构路由器,并根据前期评估,确定可重构路由设备与子系统之间以 telnet 方式进行通信,子系统与可重构综合管理系统之间以 Web 服务方式进行交互.

根据可重构路由交换平台管理子系统的功能描述,对其进行架构设计,并依据性能指标要求,利用 violetUML 软件画出其带标注的 UML 用例图、构件图和活动图分别如图 6~图 8 所示.

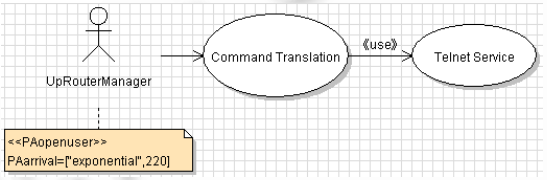

Fig.6 The use case diagram for the example system

图 6 可重构路由交换平台管理子系统的带标注 UML 用例图

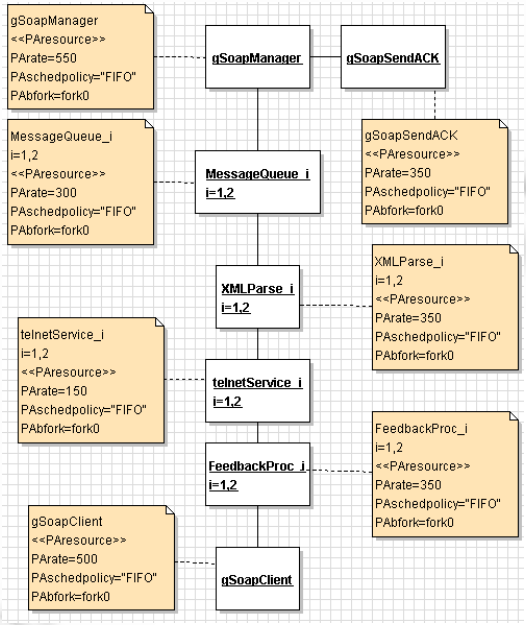

Fig.7 UML component diagram for the example system

图 7 可重构路由交换平台管理子系统的带标注 UML 构件图

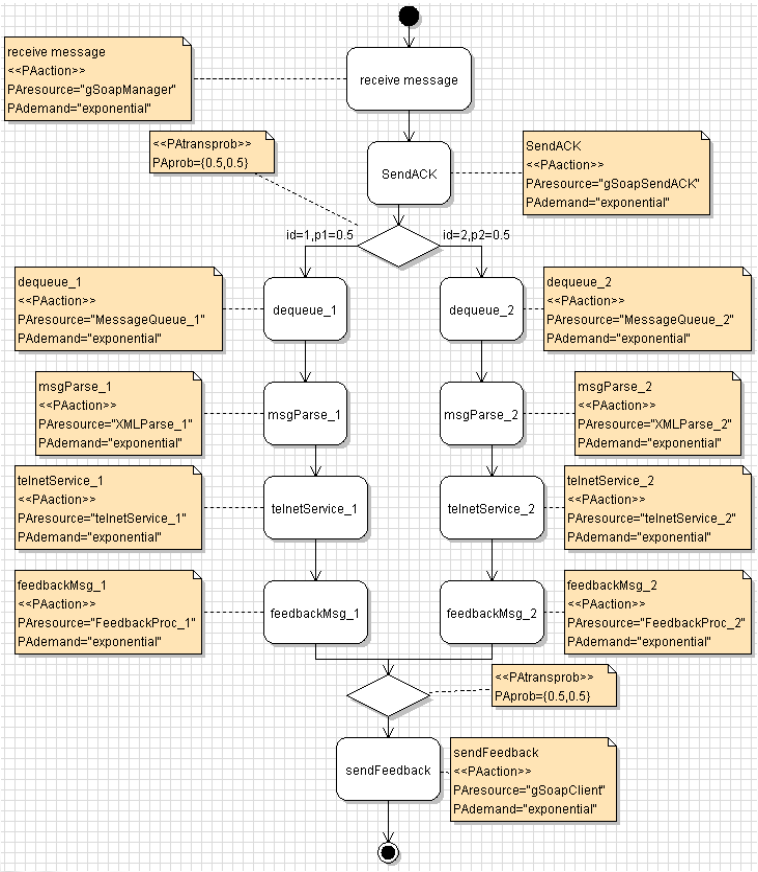

Fig.8 UML activity diagram for the example system

图 8 可重构路由交换平台管理子系统的带标注 UML 活动图

应用 UML 软件架构性能自动化工具对可重构路由交换平台管理系统进行性能预测,生成的排队网络如图 9 所示.

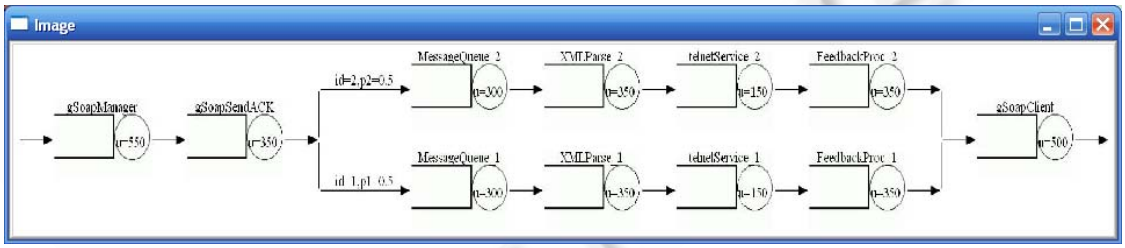

Fig.9 The queueing network model generated by the tool for the example system

图 9 可重构路由交换平台管理子系统生成的排队网络模型

根据排队网络模型的性能参数求解方法计算系统的性能指标,计算结果如下:平均系统顾客数为 11.635 9,平均系统时间为 0.052 809 5,平均等待时间为 0.008 090 54,其平均系统顾客数和平均系统时间不满足步骤 1 所设定的期望性能指标.

系统性能未达标,分析其原因可能有如下几点:设计的系统软件架构本身就无法满足系统的性能需求;在标注系统 UML 构件图时,所给出的各个构件资源的平均服务率的经验值不够准确;在提出系统性能需求时,制定

的预期性能指标过高.针对这 3 方面原因,首先考虑系统架构自身是否存在不合理性,因为这是软件设计的核心,如果能通过改进系统的软件架构来满足性能需求,比降低系统性能指标或是修改一些经验值更可靠;其次考虑原因二,查看经验值是否没有估计准确,但该方法受制于架构设计者个人的软件设计经验;最后,重新分析是否提出了不合理的性能需求,如是否未考虑到硬件资源的局限等.

4.3 软件架构改进及性能重新评估

对可重构路由交换平台管理子系统的软件架构进行重新设计,采用 gSoapSendACK 与 MessageQueue 服务并发处理的机制.

改进后的系统的带标注的 UML 用例图无变化,主要变化在 UML 活动图上,如图 10 所示.其中,粗黑色横条为同步条,用构造型《PAfork》描述,其标记值 *PAfname* 表示同步条的类型是分支同步条或汇合同步条,分别用 *fork* 和 *join* 表示,数字代表该类型同步条属于第几个同步条.从中可以看出,改进的系统设计采用了并发的思想,将活动状态节点 SendACK 描述的发送收到可重构命令的确认消息给可重构综合管理系统和可重构命令消息的处理(包括活动状态节点 dequeue 描述的根据配置的可重构路由设备 id 号分发到消息队列;活动状态节点 msgParse 描述的可重构消息解析为可重构路由设备的配置命令;活动状态节点 telentService 描述的利用 telnet 服务将可重构路由设备的配置命令发送给可重构路由设备;活动状态节点 feedBackMsg 描述的收集和分析可重构路由设备的反馈信息)以及活动状态节点 SendFeedback 描述的反馈可重构命令的执行结果给可重构综合管理系统这两个环节放在同步条之间,说明它们同时进行,提高了效率.

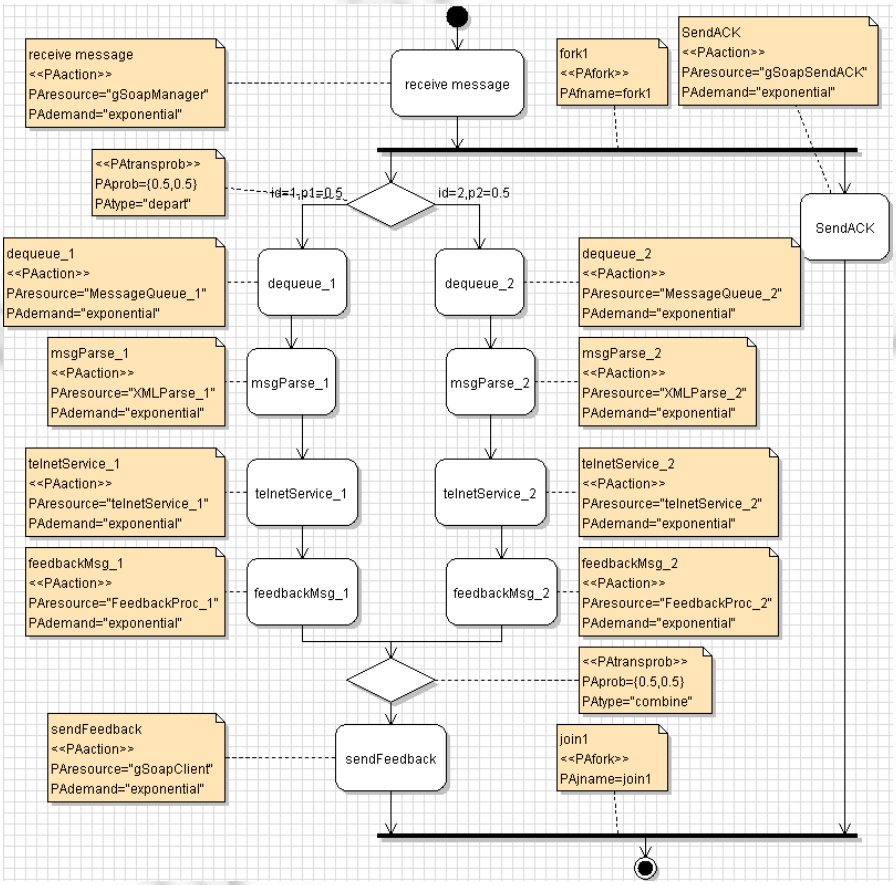

Fig.10 The modified UML activity diagram for the example system

图 10 改进的可重构路由交换平台管理子系统的带标注 UML 活动图

改进的可重构路由交换平台管理系统的带标注的 UML 构件图与图 7 所示的可重构路由交换平台管理子系统的带标注 UML 构件图基本一致,但由于图 10 所示的改进的可重构路由交换平台管理子系统的带标注 UML 活动图有同步条节点,故其构件图中每个构件资源的标记值 *PAbfork* 发生了变化,它描述的是构件资源被 UML 活动图中的活动状态节点请求时该活动状态节点的位置.如果在同步条之间,则为相应的同步条的名称;否则为 *fork*0.因此,图 7 所示的同步条节点之间的活动状态所请求的构件资源的标记值 *PAfork* 的值为 *fork*1,如图 11 所示.

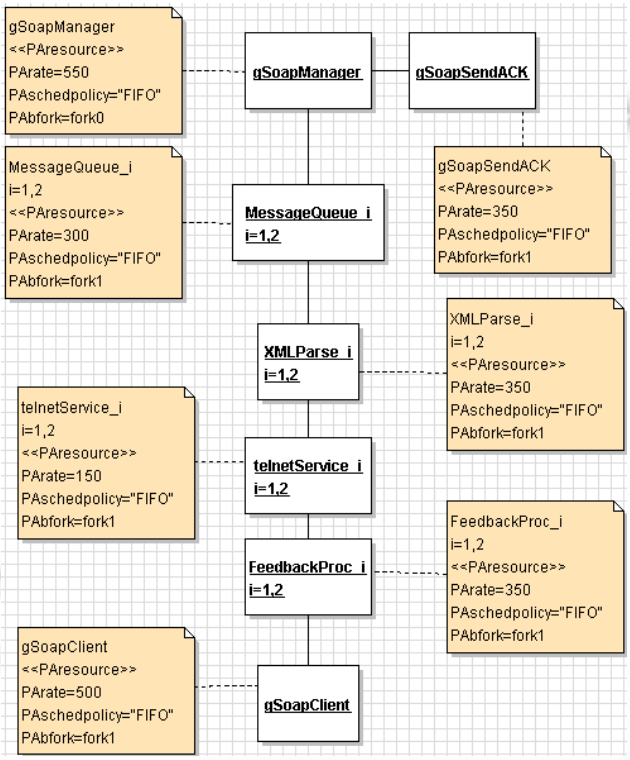

Fig.11 The modified UML component diagram for the example system

图 11 改进的可重构路由交换平台管理子系统的带标注 UML 构件图

此时,再利用 UML 软件架构性能自动化工具再次进行性能评估,得到的排队网络模型如图 12 所示.

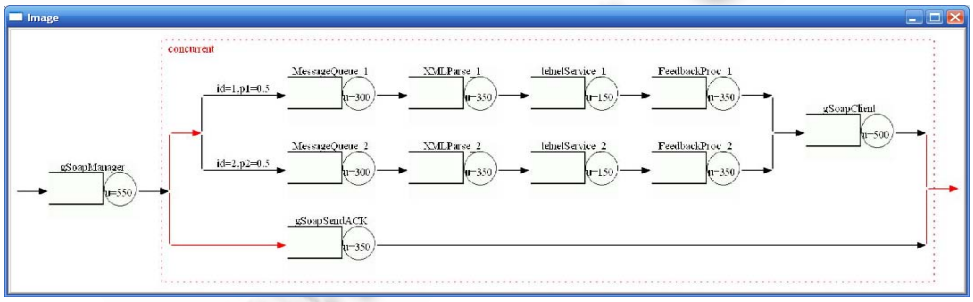

Fig.12 The queueing network model generated by the tool for the modified example system

图 12 改进的可重构路由交换平台管理子系统生成的排队网络模型

根据排队网络模型的性能参数求解方法计算系统的性能指标,其计算结果为:平均系统顾客数为 9.94,平均

系统时间为 0.045 2,平均等待时间为 0.006 1,其性能满足预期的性能指标.也就是说,系统改进后,其性能满足需求.

5 结论及下一步工作

本文提出一种基于模型的UML软件架构性能预测方法,可以在软件开发早期对软件系统的性能进行预测,不仅能对现有设计方案进行比较从而得出最优的解决方案,而且还能预测系统的性能瓶颈,同时提供了工具支持.但该方法和工具具有一定的局限性,仍需在以后的工作中作进一步的研究和完善,如:本文提出的 UML SPT 模型生成排队网络模型算法只适用于外部输入到达后离开的系统,还需要对外部输入到达系统后不再离开系统的情况作进一步的研究;本文提出的排队网络模型性能参数求解方法只适用于开环排队网络模型的性能参数求解,对于闭环和混合类型的排队网络模型还需作进一步的研究.

References:

[1] Denaro G, Polin A, Emmerich W. Early performance testing of distributed software applications. In: Proc. of the WOSP 2004. 2004. 94−103. [doi: 10.1145/974044.974059]

[2] Balsamo S, Di Marco A, Inverardi P, Simeoni M. Model-Based performance prediction in software development: A survey. IEEE Trans. on Software Engineering, 2004,30(5):295−310. [doi: 10.1109/TSE.2004.9]

[3] Herzog U, Klehmet U, Mertsiotakis V, Siegle M. Compositional performance modelling with the TIPPtool. Performance Evaluation, 2000,39:5−35. [doi: 10.1016/S0166-5316(99)00056-5]

[4] Bernardo M, Gorrieri R. A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. Theoretical Computer Science, 1998,202(1):1−54. [doi: 10.1016/S0304-3975(97)00127-8]

[5] Bernardo M, Ciancarini P, Donatiello L. AEMPA: A process algebraic description language for the performance analysis of software architectures. In: Proc. of the WOSP 2000. 2000. 1−11. [doi: 10.1145/350391.350394]

[6] Gilmore S, Hillston J. The PEPA workbench: A tool to support a process algebra-based approach to performance modelling. Lecture Notes in Computer Science, 1994,794:353−368. [doi: 10.1007/3-540-58021-2_20]

[7] Menasce DA, Gomaa H. On a language based method for software performance engineering of client/server systems. In: Proc. of the WOSP'98. 1998. 63−69. [doi: 10.1145/287318.287331]

[8] King P, Pooley R. Using UML to derive stochastic Petri net models. In: Proc. of the 15th Annual UK Performance Eng. Workshop. 1999. 45−56.

[9] Gomaa H, Menascé DA. Design and performance modeling of component interconnection patterns for distributed software architectures. In: Proc. of the WOSP 2000. 2000. 117−126. [doi: 10.1145/350391.350418]

[10] Buhr RJA, Casselman RS. Use CASE Maps for Object-Oriented Systems. Prentice Hall, 1996.

[11] Zhang Y, Huang T, Wei J, Chen NJ. Architectural level performance modeling of component system based on container middleware. Ruan Jian Xue Bao/Journal of Software, 2006,17(6):1328−1337 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/20060608.htm [doi: 10.1360/jos171328]

[12] Franks G, Hubbard A, Majumdar S, Neilson J, Petriu DC, Rolia J, Woodside M. A toolset for performance engineering and software design of client-server systems. Performance Evaluation, 1995,24(1-2):117−136. [doi: 10.1016/0166-5316(95)96868-S]

[13] Petriu DC, Wang X. From UML descriptions of high-level software architectures to LQN performance models. Lecture Notes in Computer Science, 2000,1779:217−221. [doi: 10.1007/3-540-45104-8_4]

[14] Petriu DC, Shen H. Applying the UML performance profile: Graph grammar-based derivation of LQN models from UML specifications. Lecture Notes in Computer Science, 2002,2324:183−204. [doi: 10.1007/3-540-46029-2_10]

[15] Huang YL. Research on performance testing based on UML sequence diagram [MS. Thesis]. Beijing: Beijing University of Chemical Technology, 2008. 31−47 (in Chinese with English abstract).

[16] Balsamo S, Marzolla M. Performance evaluation of UML software architectures with multiclass queueing network models. In: Proc. of the WOSP 2005. 2005. 37−42. [doi: 10.1145/1071021.1071025]

[17] Mitrani I, Chakka R. Spectral expansion solution for a class of Markov models: Application and comparison with the atrix-geometric method. Performance Evaluation, 1995,23(3):241−260. [doi: 10.1016/0166-5316(94)00025-F]

[18] Lin C. Performance Evaluation of Computer Network and System. Beijing: Tsinghua University Press, 2001 (in Chinese).

[19] Bolch G, Greiner S, Meer H, Trivedi KS. Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications. Wiley-Blackwell, 2006.

[20] Balsamo S, Marin A. Product-Form solutions for models with joint-state dependent transition rates. Lecture Notes in Computer Science, 2010,6148:87−101. [doi: 10.1007/978-3-642-13568-2_7]

[21] Ji G, Wang L, Wang ZY, Performance analysis of the non-linear asynchronous pipeline based on queuing networks. Computer Engineering & Science, 2011,33(2):65−69 (in Chinese with English abstract).

[22] Bhaskar V, Lavanya G. Equivalent single-queue-single-server mmodel for a pentium processor. Applied Mathematical Modelling, 2010,34(9):2531−2545. [doi: 10.1016/j.apm.2009.11.018]

[23] Burke PJ. The output of a queuing system. Operations Research, 1956,4(6):699−704. [doi: 10.1287/opre.4.6.699]

[24] Violet UML editor. http://alexdp.free.fr/violetumleditor/page.php

[25] Graphivz. http://www.graphviz.org

[26] OpenCV. http://opencv.willowgarage.com/wiki

[27] Matlab. http://www.mathworks.cn/index.html

附中文参考文献:

[11] 张勇,黄涛,魏峻,陈宁江.基于容器中间件的组件系统体系结构性能评价.软件学报,2006,17(6):1328−1337. http://www.jos.org.cn/1000-9825/20060608.htm [doi: 10.1360/jos171328]

[15] 黄玉麟.基于 UML 顺序图的软件性能测试方法研究[硕士学位论文].北京:北京化工大学,2008.31−47.

[18] 林闯.计算机网络和计算机系统的性能评价.北京:清华大学出版社,2001.74.

[21] 晋钢,王蕾,王志英.基于排队网络的异步非线性流水线性能分析.计算机工程与科学,2011,33(2):65−69.

李传煌(1980－),男,江西九江人,博士生,讲师,主要研究领域为系统性能预测及分析模型,网络演算及应用,新一代网络体系结构.

E-mail: chuanhuang_li@zjgsu.edu.cn

王伟明(1964－),男,博士,教授,主要研究领域为新一代网络体系结构,开放可编程网络.

E-mail: wmwang@zjgsu.edu.cn

施银燕(1986－),女,硕士生,主要研究领域为排队网络及应用.

E-mail: shiyinyan@pop.zjgsu.edu.cn