第33卷 第10期 计 算 机 工 程 2007年5月
Vol.33 No.10 Computer Engineering May 2007
·软件技术与数据库· 文章编号：1000—3428(2007)10—0038—03 文献标识码：A 中图分类号：TP311.133.1

分布构件化软件性能预测的UML模型标注方法

王克波，贾 焰，韩伟红，王志英，马永柱
(国防科技大学计算机学院，长沙 410073)

摘 要：构件能够独立部署，经过组装构成应用。将软件模型转换成性能模型的过程中，传统的功能模型不能提供足够的信息进行转换。文章提出了分布构件化软件CCPE Profile以及一种基于此的标注方法，使得分布构件化软件的非功能信息蕴含于软件模型之中，为软件模型向性能模型的自动转换提供了基础。

关键词：性能；分布式构件；UML

UML Annotation Approach for Distributed Component Based Software Performance Prediction

WANG Kebo, JIA Yan, HAN Weihong, WANG Zhiying, MA Yongzhu
(School of Computer Science, National University of Defense Technology, Changsha 410073)

【Abstract】 Distributed components are independent deployable units which can be composed to applications. The software model cannot supply enough information to transform into a performance model. CCPE Profile and a UML model annotation approach based on CCPE for distributed component based applications are proposed, so as to integrate non-functional information into functional model. The approach can be used as a basis of tool-based transformation of software model to performance model for distributed component based software application.

【Key words】 Performance; Distributed component; UML

软件的性能属性有规律可循，其性能属性往往隐含于软件的静态结构与动态结构之中[1]，比如软件体系结构或模型。因此可以通过研究软件模型，进而推演出软件性能属性的相关信息。基于软件模型的性能预测可以分为3个阶段：软件模型阶段，性能模型阶段和模型评估阶段。一段时间以来，软件建模与软件性能预测技术各自独立发展。软件模型描述软件的结构、行为，而软件性能预测技术使用QN[2]或Petri Net对软件建模。

使用自动化工具将软件模型转换为性能模型能大幅减少工作量。为了能够由软件模型自动生成性能模型，软件模型需要携带足够的信息，使得性能模型生成工具能够生成性能模型。软件模型中需要的信息[3]通常包括：软件的行为描述，软件行为的资源需求，资源描述，工作负载描述。

软件模型与其性能模型在软件的行为描述、软件运行资源描述上有一定的重叠，在工作上有重复，并且在软件开发过程中对软件模型的修改都需要修正性能模型，以反映变化，而性能预测人员需要同时掌握两种建模语言；传统建立性能模型的方式需要专家知识，难度大并且易出错。在软件模型的精化或修正过程中，需要不断地对性能模型进行修正和验证，以保证性能模型能够正确反映软件的性能属性，增大了工作量。

基于软件模型的性能预测技术特别是使用工具进行自动化转换的方法在软件模型和性能模型之间建立了桥梁，减少了维护软件模型与性能模型之间一致性的工作量，简化了性能模型的开发，降低了进行基于软件模型性能预测的门槛。

1 相关研究

随着近年来软件建模技术的进步，各个标准化组织的建模语言不约而同地向非功能属性建模领域发展，有代表性的是OMG的UML Performance Profile和ITU-T的URN。非功能属性建模语言为软件模型生成性能模型提供了坚实基础。

基于模型的性能预测研究一个重要的部分是性能模型。目前的研究工作尝试了多种性能模型[4]，如QN、PN[5]和仿真模型。QN及其派生的LQN[6]得到了较多应用。

PASA[7]方法针对一般的软件应用，能够从软件模型自动地生成性能模型，作为SPE方法以及为了实践SPE过程而引入的SPEED工具[8]的补充。

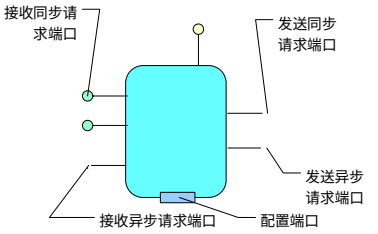
CBSPE[9]方法试图将SPE方法应用到构件领域，它将其分为两个层次：一个层次是构件层，由构件开发者开发；另一个层次是构建于其上的应用层，由组装人员完成。CBSPE采用QN进行模型评估，一定程度上为基于构件应用的性能预测提出了解决方案，但没有特别针对分布构件化应用中的问题。UML-Ψ[10]是由UML直接生成性能模型，并对性能模型进行仿真分析的工具。UML-Ψ的应用领域是一般软件，没有针对分布构件化应用。

2 构件通信模式性能模型

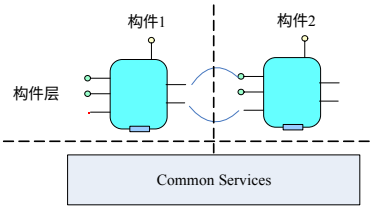
由于基于分布构件的软件在构件开发和构件组装上的分离，目前的构件技术往往以功能属性为基础。对应用组装一方来说，构件往往作为“黑盒”提供，组装人员能够从外部看到构件的功能规格，而无法了解进行性能预测需要的信息。以构件为粒度成为对构件化软件进行性能预测的自然选择。

基金项目：国家“863”计划基金资助项目(2004AA112020，2003AA115210，2003AA111020，2003AA115410)
作者简介：王克波(1975－)，男，博士生，主研方向：分布式计算；贾 焰，教授；韩伟红，副教授；王志英，教授；马永柱，助教
收稿日期：2006-05-23 **E-mail：**kebowang@gmail.com

构件是可部署的、独立的、只能通过一组接口访问的软件单元[11]。一般化的分布构件模型，如图 1 所示，其访问接口按照请求的方向分类，可以分为请求入端口和请求出端口；按照通信模式分类，可以分为同步通信端口和异步通信端口。在组装过程中，出端口与入端口相连，同步通信端口与异步通信端口不能相连。在分布构件化软件中，同步调用和异步调用都是远程调用。



(a)



(b)

图 1 构件模型

针对同步通信和异步通信两种形式，同步调用需要等待服务器端完成计算并返回结果；而异步通信则不必等到服务器端返回结果。显然，这两种通信模式对性能的影响大不相同，需要区分对待。

构件模型的同步调用是一次 RPC，客户端需要等待服务器端返回结果。同步通信方式的性能建模采用排队网络符号，如图 2 所示。为了一定程度上简化预测过程，本方法将一次同步调用服务器端简化为单个队列。

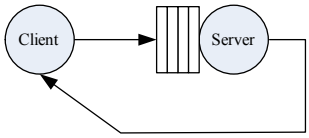


图 2 同步通信性能模型

异步通信[12]模式有多种方式实现客户端与服务器端耦合的解除。在目前流行的构件平台(CCM, EJB)中，异步通信由多次同步通信完成。在一次异步通信中，消息的生产者首先使用同步通信方式发送异步消息到消息队列，消息队列将异步消息保存后向客户端返回。

当消息的消费者在线时，消息队列将异步消息以同步通信的方式发送给消费者。以图 2 所示的同步通信模型为基础，则异步通信模型如图 3 所示。

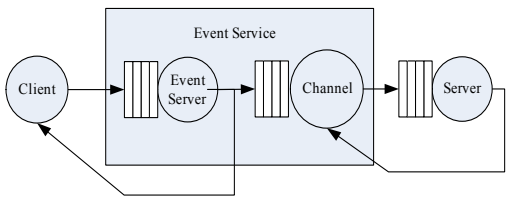


图 3 异步通信性能模型

通用服务层的性能相关信息也会影响构件组装得到应用的性能。异步通信模式依赖于通用服务层的异步消息服务。我们将异步消息服务抽象为 NotiServ。

3 CCPE Profile

分布构件化软件的性能模型可以分为 3 个层次，分别是通用服务层、构件层和组装层(即应用层)。通用服务器层由一组常用的分布式服务组成，这些服务影响构件的通信流程，因而也会影响构件进行性能合成的过程。通常这些服务有通告服务、事务服务等。在构件层，每个构件都需要得到其性能属性，应用层用来进行性能合成。应用层的性能由构件性能属性、通用服务决定的通信模式合成决定。

为了在软件模型中体现分布构件化软件的 3 个层次，引入了 CCPE Profile。

CCPE 方法采用 UML 中的活动图描述软件的行为。OMG 的 SPT 规范的 PA Profile 能够描述 Scenario、资源和工作负载，能够被用来标注软件模型。根据 CCPE 方法的构件同/异步通信性能模型，需要在软件模型中对同/异步通信进行区分。使用 UML 建模语言，在序列图中通过使用单/双向箭头表示请求的流动，可以区别同/异步通信方式；而在 UML 活动途中却无法体现。一种方式是将所有的异步通信按照消息队列的行为模型，在软件模型中全部展开成等价的同步通信模型。这种方式代价较大，对于软件模型的维护、精化不利。而根据 CCPE 的资源需求预测模型，每台服务器节点的处理能力信息也需要能够在软件模型中体现出来。显然，SPT 不能单独完成这些任务，因此，CCPE 方法对 UML 进行扩展，引入了 UML Profile for CCPE，使得构件之间的同步/异步通信方式能够在软件模型的活动图中体现出来；根据 CCPE 的资源需求预测模型计算每个资源的相对处理能力，并在软件模型中标注。

UML Profile 由一组预定义的 Stereotype 和 Tagged Value 组成。

CCPE Profile 描述的内容包括：构件间通信模式，节点描述，分布式服务资源需求描述。

下面对 UML 进行扩展，引入 UML Profile for CCPE，简称 CCPE Profile。领域模型视图如图 4 所示。

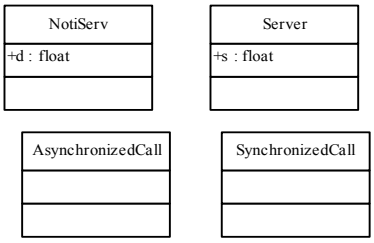


图 4 CCPE Profile

<<Server>>用来对活动图的 Swim Lane 进行标注，它表示一个服务器节点。<<Server>>包含一个 Taggedvalue，类型是 float，其值在 0 和 1 之间，用来描述该服务器节点的相对处理能力。

<<NotiServ>>用于对活动图的 Swim Lane 进行标注，它表示通告服务在该节点上运行，<<NotiServ>>包含一个 TaggedValue，类型是 float，其值在 0 和 1 之间，用来表示进行一次异步通信的服务需求。服务需求的值是按照参照节点得到的，需要根据实际节点进行转换。

<<SynchronizedCall>>标注在 UML 活动图的 Transition 上，表示这次调用是一次同步通信；

<<AsynchronizedCall>>标注在 UML 活动图的 Transition 上，表示这次调用是一次异步通信；

CCPE Profile 可以与 OMG 现有的 PA Profile 结合使用。

4 Case Study

在本节中，提出了一种基于 PA Profile 和 CCPE Profile 的标注 UML 模型的方法，即基于活动图的标注模型，采用 CCPE Profile 标注的软件模型如图 5 所示。

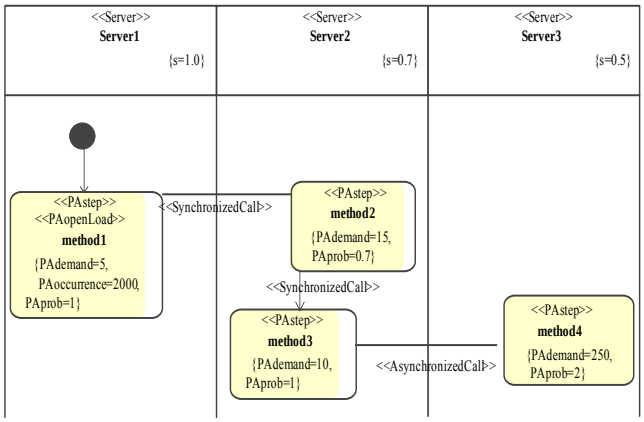


图 5 采用 CCPE Profile 标注的软件模型

基于 PA Profile 和 CCPE Profile 的标注模型基于 UML 的活动图，UML 活动图结合 PA Profile 和 CCPE Profile，不但包含了软件性能的资源需求参数，以及软件的行为模型，还通过 CCPE Profile 将部署及服务器节点处理能力等信息也在活动图中表达出来。使得性能人员在对软件进行性能分析的时候可以直观地看到所有性能相关信息。

在进行了性能标注的软件模型中，应该包括部署信息、节点信息、软件的行为模型以及每个软件行为的资源需求参数。在基于 PA Profile 和 CCPE Profile 的标注模型中，利用 UML 活动图描述并发的 Swim Lane 来表述构件的分布情形，Swim Lane 表示发生在同一个服务器节点上的软件行为，即使用 Swim Lane 表示分布构件的部署关系。

每个被当作服务器节点的 Swim Lane 都必须使用 Stereotype 类型<<Server>>来标注，并为其 TaggedValue 赋值表示该服务器节点的处理能力。这种方式即没有违背 UML 本身的语义，同时巧妙地将部署信息与节点信息在 UML 活动图中表达出来。软件的性能模型与软件行为的资源需求参数，则正好利用 UML 活动图描述软件的行为，而 SPT 中的 Stereotype 类型<<PAStep>>则描述了软件的一个基本行为及其资源需求参数。

5 总结

在基于分布构件的软件中，为了使得软件模型能够使用工具自动地转换为性能模型，必须提供一种将性能相关信息标注在软件模型中的方法。现有的标注方法不适用于分布化、构件化应用的特征，本文提出了扩展 CCPE Profile 以及基于 CCPE Profile 的软件模型标注方法，以满足分布构件化软件模型转换成性能模型的需要。本方法使性能预测过程与软件开发过程集成起来，以减小性能预测周期和代价。

参考文献

1 Clements P C, Nothrop L N. Software Architecture: An Executive Overview[M]//Component-based Software Engineering: Selected Papers from the Software Engineering Institute. IEEE Computer Society Press, 1996: 55-68.

2 林　闯. 计算机网络和计算机系统的性能评价[M]. 北京: 清华大学出版社, 2001.

3 Williams L G, Smith C U. Information Requirements for Software Performance Engineering[C]//Proceedings of the 8th International Conference on Modeling Techniques and Tools for Computer Performance. 1995.

4 Balsamo S, Marco A D, Inverardi P, et al. Software Performance Modeling: State of the Art and Perspectives[R]. Dipartimento di Informatica, Universita Ca' Foscari di Venezia, 2003.

5 Bernardi S, Donatelli S, Merseguer J. From UML Sequence Diagrams and Statecharts to Analyzable Petrinet Models[C]// Proc. of Workshop on Software and Performance. 2002: 35-45.

6 Rolia J A, Sevcik K C. The Method of Layers[J]. IEEE Transaction on Software Engineering, 1995, 21(8): 689-700.

7 Williams L G, Smith C U. PASA: A Method for the Performance Assessment of Software Architectures[C]//Proceedings of Workshop on Software and Performance. 2002: 179-188.

8 Smith C U, Williams L G. Performance Engineering Evaluation of Object-oriented Systems Speed[C]//Proc. of the 9th International Conference on Computer Performance Evaluation Modeling Techniques and Tools. 1997: 135-154.

9 Kahkipuro P. UML-based Performance Modeling Framework for Component-based Distributed Systems[C]//Proc. of UML'99. 1999.

10 Balsamo S, Marzolla M. A Simulation-based Approach to Software Performance Modeling[C]//Proc. of the 9th European Software Engineering Conference Held Jointly with the 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering. 2003: 363-366.

11 Szyperski C. Component Software: Beyond Object-oriented Programming[M]. ACM Press, Addison-Wesley Publishing Co., 1998.

12 张志伟. 面向对象异步通信中间件的研究与实现[D]. 长沙: 国防科技大学, 2004.

(上接第37页)

参考文献

1 周培德. 计算几何－－算法分析与设计[M]. 北京: 清华大学出版社, 2005-04.

2 司连法, 王文静. 快速 Dijkstra 最短路径优化算法的实现[J]. 测绘通报, 2005, (8): 15.

3 康志谕, 王明生. 城市交通网中最短路径算法及其实现[J]. 国防交通工程与技术, 2005, (1): 57.

4 王晓东, 陈国龙, 林柏钢. 网络最短路问题的改进算法[J]. 小型微型计算机系统, 2002, 23(9).

5 李宁宁, 刘玉树. 改进的 Dijkstra 算法在 GIS 路径规划中的应用[J]. 计算机与现代化, 2004, (9): 12.

6 孟正大, 王小忠. 机器人无碰撞路径规划方法研究及实现[J]. 华中科技大学学报, 2004, 32(增刊).