软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2011,22(Suppl.(2)):63−79
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# ASBS 的反射 Petri 网性能建模方法*

葛 亮[1], 张 斌[1+], 刘 莹[2], 李 飞[1,3]

[1](东北大学 信息科学与工程学院,辽宁 沈阳 110819)

[2](东北大学 软件学院,辽宁 沈阳 110819)

[3](东北大学 秦皇岛分校 计算中心,河北 秦皇岛 066004)

## ASBS Performance Modeling Approach using Reflective Petri Net

GE Liang[1], ZHANG Bin[1+], LIU Ying[2], LI Fei[1,3]

[1](College of Information Science and Engineering, Northeastern University, Shenyang 110819, China)

[2](College of Software, Northeastern University, Shenyang 110819, China)

[3](Computer Center, Northeastern University at Qinhuangdao, Qinhuangdao 066004, China)

+ Corresponding author: E-mail: zhangbin@ise.neu.edu.cn

**Ge L, Zhang B, Liu Y, Li F. ASBS performance modeling approach using reflective Petri net. *Journal of Software*, 2011,22(Suppl.(2)):63−79.** http://www.jos.org.cn/1000-9825/11027.htm

**Abstract**: With the widespread adoption of SOA in large-scaled distributed systems, Service-Based Software system has brought much attention in the field of software engineering. Capable of monitoring system status and dynamically adjusting accordingly, Adaptive Service-Based Software system enhances SBS with self-adaptive ability to satisfy the system requirements in aspects of QoS, etc. This paper defines the structural description of ASBS, proposes a performance evaluation model based on reflective Petri net, and introduces the constructing and analyzing approaches for the performance model, which also incorporates explorations in ASBS performance analysis.

**Key words**: ASBS; performance model; reflective Petri net

**摘 要**: 随着 SOA 技术在大规模分布式系统中的广泛应用,基于服务的软件系统(service-based software system,简称 SBS)已成为软件工程领域关注的热点.为了处理服务动态性给 SBS 带来的影响,基于服务的自适应软件系统(adaptive service-based software system,简称 ASBS)通过对系统的监测以及动态调整,为 SBS 扩展了自适应能力,从而保证系统 QoS 等方面的需求得到满足.从 ASBS 的性能角度出发,定义了 ASBS 的结构描述,并提出了基于反射 Petri 网的 ASBS 性能分析模型,介绍了性能模型的构建方法以及模型的分析方法,对 ASBS 性能分析进行了探索.

**关键词**: ASBS;性能建模;反射 Petri 网

随着大规模分布式应用的发展,感知环境以及本身状态并调整行为以适应变化的自适应系统已成为重要的研究方向之一[1].由于构建的灵活性,Web 服务通常是实现动态自适应工作流以及面向服务软件系统适合的技术选择[2].基于服务的软件系统(service-based software systems,简称 SBS)是一种基于 SOA 的软件系统,它能

* 基金项目: 国家自然科学基金(61073062); 核高基科技重大专项(2010ZX01045-001-008)
收稿时间: 2011-05-02; 定稿时间: 2011-11-07

© 中国科学院软件研究所 http://www.jos.org.cn

够通过服务的动态发现和动态绑定,使得系统可以有效地适应其运行时的环境变化[3].在动态变化的分布式环境中,高质量的 SBS 可以监测系统变化状态,分析不同 QoS 控制开销并调整服务配置,以满足不同的性能需求,这样的 SBS 被称为基于服务的自适应软件系统(adaptive SBS,简称 ASBS)[4].

相对于传统软件系统,ASBS 由于其结构以及实现方式等方面具有许多特性.不同于传统软件设计开发到部署应用的过程,SOA 根据现有的服务构建业务流程通过延迟绑定实现组合服务的部署,通过编排或编制的方式快速构建组合服务,具有松耦合、延迟绑定等特点.服务的动态性使得 ASBS 具有较强的适应能力,相对于传统的过程逻辑 SOA 具有一定灵活性(flexibility),使得组合服务可以动态地对流程中的服务进行调整,为组合服务的自适应提供了基础.对系统状态的监测以及对组合服务的调整共同实现了 ASBS 的自适应循环.ASBS 的生命周期不在部署时结束,而会根据变化情况通过监测,适应构成的反馈,循环进行持续的调整.

系统性能相关的约束是驱动系统适应动作的主要因素,而在设计期对 ASBS 整体的性能分析是 ASBS 设计中的关键问题之一.传统组合服务性能分析方法可以在 SBS 系统设计阶段由资源的描述对系统的运行性能进行预测,而为了提高系统性能,ASBS 为 SBS 加入了自适应的能力,因此,对系统性能的分析需要把这种适应动作融入系统性能分析中,即将 ASBS 作为一个整体进行分析.从传统性能分析角度考虑,对 ASBS 整体的性能分析可以实现 ASBS 部署运行前系统整体性能的预测;从 ASBS 设计角度考虑,资源以及自适应策略是影响系统整体性能的重要因素,对 ASBS 整体的性能分析,一方面,有助于在给定适应策略下选择更适合的服务以提高系统性能,另一方面,有利于对适应策略进行分析以便针对具体资源情况选择更合适的自适应规则.相对于系统的业务功能,自适应动作为系统添加了复杂的执行逻辑,使得 ASBS 的自适应能力在提高系统性能的同时,也为系统的性能分析带来了一定的难度,并为 ASBS 系统性能分析提出了新的需求.为此,在 ASBS 系统性能分析中需要一种灵活的模型,解决传统的基于模型的性能分析方法将适应行为与业务逻辑硬性混编带来的问题,并可以有针对性地解决 ASBS 系统性能的复杂需求.

针对这种情况,本文从 ASBS 结构角度出发,对 ASBS 进行了详细定义,结合 ASBS 的结构提出了 ASBS 性能模型的反射 Petri 网模型,给出了模型框架、具体的模型构建方法以及应用 ASBS 反射 Petri 网性能模型进行系统性能分析的基本思路.

本文第 1 节介绍 ASBS,并针对 ASBS 的结构给出相关的描述.第 2 节介绍反射 Petri 网模型,并结合 ASBS 给出反射 Petri 网框架.第 3 节给出 ASBS 性能模型的构建方法以及基于模型的分析方法的介绍.第 4 节通过比较自适应组合服务方面的相关工作分析,介绍本文的出发点.第 5 节总结本文的内容,并对未来工作进行展望.

## 1 基于服务的自适应软件系统(ASBS)

参考 Yau 等人的研究[4,5],给出 ASBS 的概念视图如图 1 所示,ASBS 的适应反馈回路(closed-loop)主要通过监测器(monitor)以及适应器(adapter)两个模块实现,它们共同作用于连接 ASBS 的两个层次.监测器负责将包括组合服务执行状况以及资源状态两部分内容的系统状态进行收集并传递到适应器进行分析处理.监测器对组合服务执行状态和资源状态采用“push(推)”和“pull(拉)”两种不同的监测策略,由于前者在一定程度上是可控的,因此,当组合服务执行状态出现变化时,主动将当前状态通知监测器,而在后者情况下,对于不可控的环境资源监测器需要主动获取状态信息.监测器收集的系统状态随后被传递到适应器,适应器遵循 ASBS 的适应策略对系统状态进行分析,根据相应的触发式规则生成具体的适应指令,调整服务的执行甚至改变组合服务的结构.

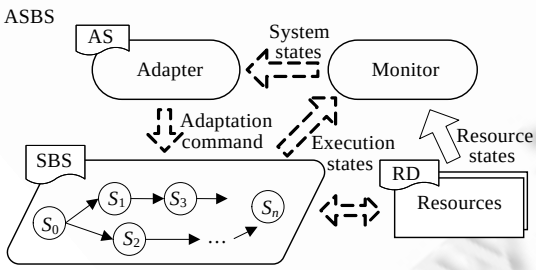


Fig.1 Conceptual view of ASBS

图 1 ASBS 概念视图

根据 ASBS 概念视图的描述,接下来给出 ASBS 的详细描述.

© 中国科学院软件研究所 http://www.jos.org.cn

### 1.1 ASBS的描述

ASBS 定义为一个四元组⟨*SBS*,*RD*,*AS*,*M*0⟩,其中,SBS(service based software)为基于服务的软件系统,描述了系统的业务逻辑;RD(resource description)对服务空间中所有的资源提供描述;AS(adaptation strategy)描述了系统的自适应策略;*M*0 描述了系统的初始配置.

SBS:以 BPEL 形式对业务逻辑进行描述,使用抽象服务描述业务操作,将系统的流程建模为一系列服务的组合.在 SBS 中,定义集合 *BService* 为 SBS 中所有业务操作构成的集合,它可以通过抽象服务构成的服务集描述 $BService=\{s_1,s_2,...,s_n\}$,其中,具体抽象服务 $s_i,i\in[1,n]$ 描述了业务流程中的第 $i$ 个操作.在本文的后续内容中,为方便介绍,使用业务流程中的操作描述抽象服务.

RD:资源描述是系统性能模型构建的基础,资源模型从功能以及非功能两方面属性对资源进行描述.在 ASBS 中,服务通过服务空间组织与管理,不同于开放世界(open-world)系统,ASBS 中服务的数量是有限的并且是有组织管理的,在 ASBS 中并不考虑服务的发现问题.资源功能描述建立了资源以及 SBS 流程的业务操作之间的匹配关系,这部分内容并不是本文关注的重点,因此将不作详细讨论.

定义集合 *SResource* 描述 ASBS 中所有可用资源构成的集合,$SResource=\{res_1,res_2,...,res_n\}$,其中 $res_i$ 描述了资源集中的第 $i$ 个资源.同时,为了方便后续 ASBS 性能模型描述,本文对 ASBS 规定如下约束:任意不同业务操作所匹配的资源集合相互斥.

AS:自适应策略是系统适应行为的依据,针对系统运行过程中状态变化情况的不同,依据触发式规则产生具体的适应调整指令,基础层则根据指令进行调整,完成系统的自适应.本文对 ASBS 的自适应策略定义了一种简单的描述语言,其详细内容将在接下来的小节中具体介绍.

*M*0:[*SBS*→*RD*],系统初始配置定义了 ASBS 系统首次运行情况下业务逻辑与资源之间的调用关系,在系统实例执行的最初节点载入系统初始配置,确定调用关系并执行服务请求处理,在系统随后的运行过程中,系统可以通过自适应调整对初始配置的调用关系进行调整.

可以从两个层次理解 ASBS 的结构,RD,SBS,*M*0 共同构成 ASBS 的基础层以及 AS 构成的系统控制层.ASBS 的基础层关注于系统业务需求的实现,以基于 SOA 的方式依据现有服务资源,设计业务逻辑,构建组合服务作为系统实现.同时,SOA 松耦合、延迟绑定等特点有利于系统运行时的自适应.ASBS 的控制层关注于系统的非功能需求,通过自适应行为,针对运行过程中不同的状态动态调整系统,保证服务质量、可靠性等非功能需求的满足.

ASBS 中的 SBS 采用 BPEL 进行描述,其内容并不是本文的关注点,接下来我们将详细介绍关于 AS 以及 RD 的描述.

### 1.2 自适应策略AS

自适应策略针对 ASBS 的基础层建立的一个复杂的、可替换的操作模式集.集合中的操作模式由作用于基础层的基本操作组合而成,在一定的情境下被触发并对基础层 ASBS 进行调整.在自适应策略中,ASBS 设计者制定具体的情境触发规则以及各情境下所触发操作集的具体内容,描述了系统运行过程中垂直于业务流程行为的自适应行为.通过定义复杂的情景可以为自适应策略补充描述能力,在本文中为精简内容考虑情境仅包含业务操作的执行状况以及资源的可用情况.

为了对 ASBS 的自适应策略进行描述,本文提出一种简单而标准的自适应策略描述语言,其语法参考 Hoare 的 CSP,考虑控制结构并在其基础上加入了一些特殊的描述语义,方便其转化为相应的反射 Petri 网模型.描述语言采用基于判定-命令(guarded-commands)的形式,定义了原始类型以及原语,同时,为了分析方便,定义了模式(pattern)以及判定(guards).适应策略描述语言的语法 BNF 见表 1.适应策略根据系统当前状态进行判定识别(guard evaluating),当判定条件得到满足时,将触发具体的适应动作.适应策略描述语言中的原始类型包括:整数(NAT)、布尔值(BOOL)、业务操作(BusinessOperation)、资源(ServiceResource)以及集合(set)和笛卡尔积(Cartesian product).通过笛卡尔积可以引入新的类型,如业务操作与具体资源之间的调用关系可以定义为

© 中国科学院软件研究所 http://www.jos.org.cn

(ServiceBND:BusinessOperation×ServiceResource),其中,“×”为笛卡尔积.原语包括:Matchto(),CurrentBND(), Ondemand()以及 Available(),分别表示获得与指定业务操作所匹配的所有资源的集合、获得指定的业务操作所调用的资源、指定的业务操作是否被请求以及指定的资源是否可用.第 1 条操作根据 ASBS 中的描述获得,后 3 条操作则可以由反射 PN 实现,详细内容将在本文后续部分加以详细介绍.

**Table 1** BNF of adaptive strategy description language

**表 1** 自适应策略描述语言 BNF 描述

| **Gramma BNF for ASBS Adaption Strategy**. |
|---|
| ***Element***::=***BusinessOperation***\|***ServiceResource***\|***ServiceBND*** |
| ***BusinessOperation***::=variable\|constant |
| ***ServiceResource::=***variable\|constant\|CurrentBND(***BusinessOperation***) |
| ***ServiceBND::=***⟨***BusinessOperation***,***ServiceResource***⟩ |
| ***Expression***::=digit\|variable |
| ***Predicate***::=***Expression RelOp Expression***\|Ondemand(***BusinessOperation***)\|Available(***ServiceResource***) |
| ***RelOp***::=⟨\|⟩\|=\|!= |
| ***LogOp***::=exists\|foreach |
| ***BoolOp***::=and\|or |
| ***SetOp***::=∩\|∪\|\ |
| ***Set:***:={}\|***ServiceResourceSet***\|***Set SetOp Set*** |
| ***ServiceResourceSet:***:={}\|{***ServiceResourceList*** }\|***Pattern***\|Matchto(***BusinessOperation***) |
| ***ServiceResourceList::=ServiceResource***\|***ServiceResource List***,***ServiceResource*** |
| ***Pattern***::={variable In Expr\|***Guard***} |
| ***Guard***::=***Predicate***\|***LogOp*** variable ***InExpr, Predicate***\|not(***Guard***)\|***Guard BoolOp Guard*** |
| ***InExpr***::=$\varepsilon$\|in ***ServiceResourceSet*** |

在自适应策略描述中可以通过常量(constant)显式地使用基础层中的元素,如使用具体业务操作名表述 SBS 中的业务操作.同时,可以使用变量(variable)隐式地表述基础层中的元素,并通过赋值操作为变量设置实例,如赋值变量 *rsn=CurrentBND*(*BOp_name*).一个 *pattern* 的简单例子可以描述成:{*p* in *Matchto*(*BOp_name*)|*Available*(*p*)},其中 *p* 是类型为 *ServiceResource* 的变量,这个 *pattern* 描述了所有满足当前状态可用的匹配名为 *bnsop_name* 的服务资源.一个简单的 *guard* 的例子,如:exists *p* in *Matchto*(*BOp_name*),*CurrentBND*(*BOp_name*) and *Available*(*p*),这个 *guard* 描述了在名为 *bnsop_name* 操作被请求时刻与其匹配的可用资源.另外,在策略描述语言中,使用“*”描述重复,“?”描述输入,“@”描述替换.一种特殊的遍历形式可以描述为*(*e*1 in *E*1,*e*2 in E2,…, *en* in *En* in *E*)[command],变量 *e*1,*e*2,…,*en* 分别对应了集合 *E*1…*En* 中的元素,该语句含义为对集合 *E*1…*En* 中的所有元素进行遍历执行指令 command.

适应动作由一系列对系统进行调整的基本指令组成,ASBS 对系统的操作主要反映为对业务操作与服务资源之间调用关系的调整,在本文中基本的指令包括:创建调用关系 *CreateBND*(),解除调用关系 *DestroyBND*(),载入系统配置 *LoadCfg*(),更新系统配置 *UpdateCfg*(),清除指定业务操作调用关系丢弃返回结果 *FlushBOperation*(),实现调整动作 *Shiftdown*().通过创建以及解除业务操作与服务资源的调用关系可以根据系统运行中遇到的变化情况动态地调整系统运行实例,从而提高系统运行实例的性能以及可靠性等.而对系统配置的调整则体现了系统针对变化情况的演化,以使得后续系统运行实例更高效地执行.特殊的清空操作是实现系统实例调整的关键,该操作丢弃了当前执行中的调用并重新启动操作执行.系统运行实例在执行最初阶段载入系统配置并启动,而自适应策略通过对系统变化情况分析后对系统配置的更新操作 *updateCfg*(),实现系统的演化,由此,相对于原始配置,后续运行实例将适应新的环境.适应动作基本指令的详细描述见表 2.

**Table 2** Basic operation of adaptation

**表 2** 适应动作基本指令

| *CreateBND*(*ServiceBND*) | *DestroyBND*(*ServiceBND*) |
|---|---|
| 在基础层中添加指定的业务操作与资源之间的调用关系 | 删除基础层中指定的业务操作与资源之间的调用关系 |
| *LoadCfg*() | *UpdateCfg*(*Set*(*ServiceBND*)) |
| 读取系统当前的配置,作用于每次系统实例执行最初阶段,以在基础层建立初始的业务操作与资源之间的调用关系 | 根据所描述的业务操作与资源的调用关系更新系统配置,从而后续实例将根据新的配置运行 |
| *FlushBOperation*(*BusinessOperation*) | *Shiftdown*() |
| 清除指定业务操作的当前执行,用舍弃当前调用资源的返回结果,并重新执行业务操作 | 将适应策略最终实现为基础层相应的调整 |

© 中国科学院软件研究所 http://www.jos.org.cn

本文所提出的自适应策略描述语言仅从 ASBS 自适应策略的最基本描述展开讨论,所涉及的原语相对有限,然而,该描述语言具有良好的扩展性,当面临复杂自适应策略描述需求的情况下,可以方便地引入新的类型以及原语.在为自适应策略描述语言增加内容后,相应的需要在性能模型框架以及性能分析方法中补充对应的内容,从而保证性能分析方法的可用性.

### 1.3 资源描述RD模型

服务的 WSDL 以 XML 形式给出了服务功能相关的描述,包含了服务操作、触发机制、消息传递以及服务所在环境的端口信息等一系列的服务功能相关内容的定义.然而,随着对组合服务 QoS 关注的日益提升,WSDL 显示出了其在服务性能方面描述能力的不足.在构建组合服务以及组合服务运行自适应时,仅有服务的功能描述已经不足以支持系统更加高效的执行,在服务的描述中加入 QoS 相关的属性成为设计高效并具有适应能力组合服务的一个前提.Ambrogio 等人提出的 Q-WSDL[6]针对传统 WSDL 在服务 QoS 方面描述能力的不足,对其进行了轻量级的扩展,以便于非功能相关属性的描述.本文中 ASBS 构建组合服务时同样选择 Q-WSDL 对服务的性能进行描述.相对于其他服务 QoS 方面的研究,Q-WSDL 仅关注于从 WSDL 扩展角度对服务性能的描述,而并不在 WSDL 之上引入新的描述语言,这种轻量级的扩展无需改变服务 WSDL 初始的描述,不需要复杂的专业知识,同时具有较强的描述能力,易于应用.

Q-WSDL 参考 UML SPT Profile 对 QoS 相关属性的描述以及其关联到 UML 模型的方法,引入了性能相关的描述类.由于 Q-WSDL 的目的在于对服务给出详细的 QoS 描述,而本文中仅关注性能相关属性,由此,为了方便环境建模以及后续应用,本文对 Q-WSDL 进行了微小的缩减,仅使用其中一部分内容,具体如下:Network,服务所处环境的描述,包括描述网段用户传输速率 BitRate,描述端到端 IP 包传输速率 Delay,端到端时延变化 Jitter 以及传输失败丢包率 PacketLoss;Availability,服务可用性的描述,包括修复时间 TimeToRepair(TTR),失效间隔 Time-Between-Failure,期待可用性 ExpAvailability;Reliability,连续提供服务能力的描述,包括失效间隔 TimeBetweenFailure(TBF)以及期待失效 ExpFailures;OperationLatency,操作所需时间的描述,包括服务器端执行操作所需时间 ServiceTime 以及完成操作的端到端时间 TurnAround;OperationDemand,描述服务操作请求密度,包括请求到达速率 ArrivalRate.

### 1.4 ASBS详例

本节通过一个 ASBS 的详例来辅助说明研究内容,这个详例在 Grassi[7]等人以及 Palacin[8]等人的研究中都有过详细的讨论,其流程如图 2(a)所示,服务 *S*1 需要调用服务 *S*2 完成相应的工作,而在服务空间中,服务 *S*2 可以选择 *C*2,*C*3,*C*4 实现部署.假设 *C*2 位处在通用服务器上,对访问限制相对宽松,可以通过 Internet 访问并且速度较快;而 *C*3,*C*4 位处在专属服务器上,提供访问受限制的服务,需要通过较慢的私有网络(VPN)访问.通过 Internet 访问通用网络服务可以得到较快的访问速度,但在有的情况下难以保证其连接以及服务器的稳定性,通过 VPN 访问私有网络中的服务,虽然访问速度不及 Internet,但可以提供更高的连接以及服务器的稳定性.

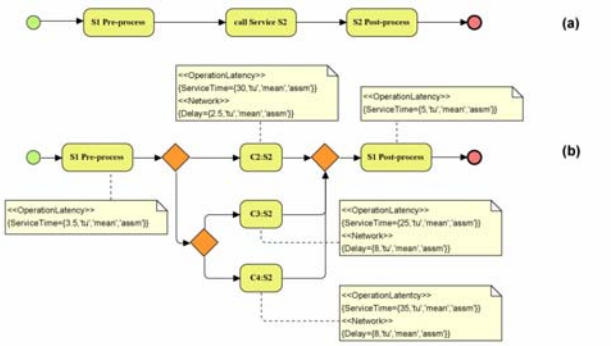


Fig.2 (a) Business workflow (b) Business workflow with resource descripiton

图 2 (a) 业务流程 (b) 结合资源的业务流程

© 中国科学院软件研究所 http://www.jos.org.cn

在这个 ASBS 详例中,服务 *C*2,*C*3,*C*4 共同构建了服务 *S*2 的候选服务集,为了提供更好的系统性能,需要根据当前情况选择最佳提供者实现服务 *S*2,而系统执行情况的变化也会导致 ASBS 根据自适应策略进行调整.根据描述,通过 Internet 访问通用服务器可以带来更快的速度,然而,由于在 Internet 情况下稳定性的不足,专用服务器上,*C*3,*C*4 可以作为备选服务在通用服务无法访问时替换 *C*2,以保证 ASBS 实例的运行.同样,*C*3,*C*4 在性能上的差异也将对服务的选择产生影响.结合资源匹配集以及资源描述,详例的流程可以如图 2(b)所示来描述.

在流程以及资源描述的基础上,下面将讨论具体的自适应策略,从而完整该 ASBS 详例.如前述介绍,ASBS 的自适应规则可以从 ASBS 系统运行实例以及 ASBS 系统两个方面进行描述.ASBS 系统运行实例自适应规则根据运行时刻系统状态对运行实例进行调整,其调整动作仅针对该运行实例具有挥发性,并不影响 ASBS 系统后续的运行实例,是 ASBS 系统自愈(self-healing)能力的一种体现.而 ASBS 系统的自适应规则根据系统状态对系统配置进行调整,由此影响系统未来的运行实例,可以看作系统的自重置(self-reconfiguration)能力的一种体现,另一方面也体现了系统的学习演化能力.

对于图 2 描述的 ASBS 详例,在运行时刻,Internet 网络拥堵以及服务器负载过重都可能导致部署在通用服务器上的 *C*2 无法访问,这种情况下 ASBS 将动态调整选择可用的资源(*C*3,*C*4)临时替换 *C*2,从而保证 ASBS 实例的运行.在选择服务进行替换时仍需对所选择的资源检查其可用性,为了描述简单,这里对可用资源并不继续进行分析,仅进行随机选择(如“@”所描述).图 3(a)对上述规则流程进行了描述,规则的伪代码描述如下:

Rule 1:*[*Ondemand*(*S*2) and not *Available*(*C*2)→[

*Available*(*C*3)→[*DestroyBND*(*S*2,*C*2);*CreateBND*(*S*2,*C*3);*FlushBOperation*(*S*2);]

@*Available*(*C*4)→[*DestroyBND*(*S*2,*C*2);*CreateBND*(*S*2,*C*4);*FlushBOperation*(*S*2);];]

在上述规则中,*C*2 失效情况下选择服务对其替换是一种临时的行为,仅针对当次服务调用有效,在后续的对服务 *S*2 请求到达时仍将首选尝试访问服务 *C*2.对于 *C*2 失效的情况,ASBS 系统同样可以进行分析,根据系统自适应规则调整系统配置,使得后续系统运行实例适应新的情况(*C*2 失效).这里参考 Palacin 的研究,定义一种连续累计规则记录 *C*2 的失效情况,当累计到一定数量时,调整 ASBS 的配置,选择可用的 *C*3/*C*4 替换初始配置中的 *C*2,后续的实例将根据新的配置运行.在这种调整规则情况下,可以同时定义一种恢复规则,在 *C*2 正常工作时再次调整 ASBS 恢复原配置.图 3(b)简述了上述规则的流程,具体规则采用伪代码描述如下:

Rule 2:*Var C*2*FailureCount*,*Threshold:NAT*;

*[*Ondemand*(*S*2) and not *Available*(*C*2)→*C*2*FailureCount*=*C*2*FailureCount*+1;

*Ondemand*(*S*2) and *Available*(*C*2) and *C*2*FailureCount*>0→*C*2*FailureCount*=*C*2*FailureCount*−1;

C2Failure<1→[CreateBND(S2,C2);UpDateCfg({⟨*S*2,*C*2⟩})]

@*C*2*Failure*>*Threshold* and *Available*(*CurrentBND*(*S*2))→

[*DestroyBND*(*S*2,*C*2);*UpdateCfg*({*S*2,*CurrentBND*(*S*2)})];]

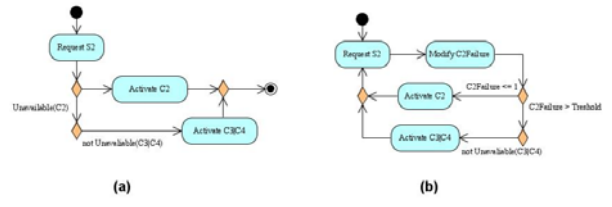


Fig.3 Adaptive strategy description

图 3 自适应规则

## 2 基于反射 Petri 网的 ASBS 性能模型

自适应是 ASBS 的重要特性之一,系统的适应是垂直于系统业务行为的,因此,系统的业务行为与适应动作的分离是 ASBS 性能模型构建过程中一个首要考虑的因素.目前大多数自适应系统性能分析过程中缺乏这方面

© 中国科学院软件研究所 http://www.jos.org.cn

的考虑,在性能模型构建过程中,将系统行为与适应动作相混杂,使得系统行为模型变得庞大复杂而难以理解,无法识别某一时刻系统的业务行为以及状态.同时,由于适应行为被“硬性地”建模于系统模型中难以分离,导致系统模型的维护、调整、重用等可能性的严重降低.最后,由于在设计期往往难以预计系统复杂的行为,新的适应行为的引入对整体建模方法带来了极大的挑战.

为此,本文在 Capra[9,10]等人研究的基础上提出了一种基于反射 Petri 网(reflective Petri net,简称 RPN)的 ASBS 性能分析模型,参照 ASBS 的特征使用两层 Petri 网进行建模,在基础层建模 ASBS 组合服务的行为,在反射适应层对 ASBS 的适应策略进行建模,使得 ASBS 组合服务的业务行为与 ASBS 的适应调整行为相独立,解决了硬性建模系统行为以及适应策略带来的上述问题.针对 ASBS 复杂的结构,这种层次的建模方法为 ASBS 的性能模型构建以及性能分析带来了很多的灵活性和便利性.

### 2.1 基于反射Petri网的ASBS性能模型

基于反射 Petri 网的 ASBS 性能模型主要包含基础层和反射适应层两个层次,这两个层次通过反射框架相连接,其结构如图 4 所示.

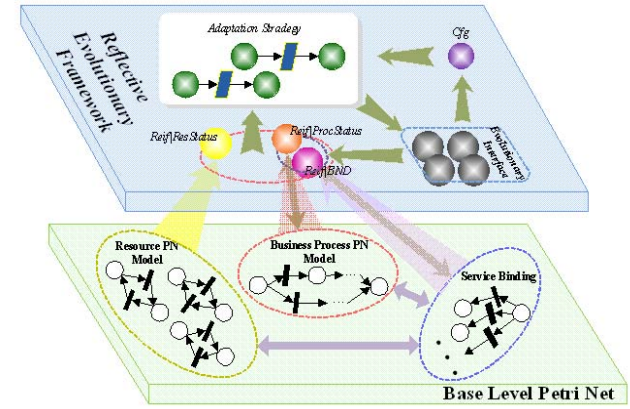


Fig.4 Reflective Petri net based ASBS performance model

图 4 基于反射 Petri 网的 ASBS 性能模型

基础层:基础层由描述系统业务行为的模型构成.对于 ASBS 组合服务的业务流程,资源及业务流程与资源的交互关系共同描述了组合服务的业务行为,是垂直于系统的适应动作的,因此,在 ASBS 的性能模型中,流程模型、资源模型以及流程资源之间的交互关系模型共同构成了性能模型的基础层.基础层模型在不了解上一层(反射适应层)的情况下独立运行,通过具体化(reification)将行为状态映射到上层,并且两者之间保持一种因果关系(causal connection),每当基础层模型状态发生改变时,基础层具体化立即随之更新,保持与基础层模型状态的向上一致性.为了对系统的性能以及可靠性等相关属性进行分析,本文采用广义随机 Petri 网 GSPN 描述基础层系统行为,为了描述方便,在后续内容中使用名词 Petri 网概括描述 Petri 网族.

反射适应层(演化层):反射适应层是相对于基础层的元模型层(meta-level).ASBS 的适应策略在反射适应层建模,由此体现了适应动作与系统业务行为之间的垂直关系.在反射适应层中,反射 Petri 网框架提供了基础层具体化以及演化接口位置,从而实现了对基础层模型的操作.ASBS 的适应策略在反射适应层中通过一种具有高相似性的有色 Petri 网——良构网(stochastic well-formed net,简称 SWN)进行建模,策略描述语言中的原语由基础层具体化位置获得相应的信息,而 ASBS 适应策略的基本指令通过演化接口位置实现对基础层模型的调整.另外,ASBS 配置作为一种特殊的策略在反射适应层中同样通过相应的位置进行描述.独立的 ASBS 适应策略模型以及反射 Petri 网框架通过上述互斥的接口位置集合(基础层具体化、演化接口以及 ASBS 配置)相连接,共同构建成了反射适应层.

© 中国科学院软件研究所 http://www.jos.org.cn

### 2.2 反射Petri网框架

反射框架同样由良构网 SWN 描述并具有固定的结构,在 ASBS 的业务流程与自适应策略之间建立连接,提供基础层的状态并实现具体的适应指令.基础层内容在反射 Petri 网中由具体化位置描述,不同部分被具体化为相应的有色位置,随着基础层模型状态的变化,其具体化位置中的标识随之变化.在反射框架中使用“Reif”前缀标注基础层具体化位置,位置 *Reif|ProcStatus*,*Reif|ResStatus* 以及 *Reif|BND* 分别描述了当前业务流程的执行状况、当前资源状况以及当前业务操作与服务资源的调用关系.另外,ASBS 的配置通过有色位置 *Cfg* 描述.在反射框架中,基础层具体化与基础层状态保持一致性,自适应策略通过演化接口对基础层具体化位置中的标识进行修改,实现对基础层的调整.演化接口位置同样通过有色位置描述,自适应策略执行的结果可以表现为对具体演化接口位置中传递相应的 *token*,从而触发对基础层具体化位置中标识的调整.类似基础层具体化位置的命名规则,在反射框架中使用“*EvInt*”前缀标注演化接口位置.演化接口位置通过反射框架中固定的结构连接自适应策略以及基础层具体化,根据操作的性质演化接口位置集合可以分为调整接口和查询接口两个子集.前者包含位置 *EvInt|FlushBOp*,*EvInt|CreateBND*,*EvInt|DestroyBND*,*EvInt|LoadCfg*,*EvInt|UpdateCfg* 以及 *Shiftdown*,这些接口位置描述了对基础层具体化位置 *Reif|BND* 和 ASBS 配置 *Cfg* 内标识的调整操作;而后者包含位置 *EvInt|CurrentBND*,*EvInt|Ondemand* 以及 *EvInt|Available*,这些接口位置描述了对基础层状态的查询.反射框架中的接口位置与ASBS自适应策略描述语言的相关原语以及操作之间具有直接的对应关系.反射框架的具体描述如图5所示.

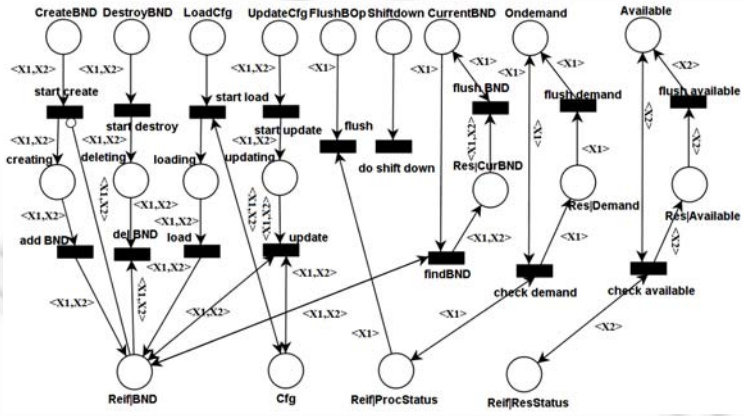


Fig.5 Reflective Petri net framework

图 5 反射 Petri 网框架

反射框架中通过颜色标识描述基础层内容,反射框架的基本颜色类可以划分成 *BusinessOp* 和 *Resource* 两个子类别,*BusinessOp* 对应了 ASBS 的业务操作集 *BService*,元素由 ASBS 操作集中的所有抽象服务构成;*Resource* 对应了 ASBS 的资源集 *SResource*,类似地,其元素由 ASBS 资源集中的资源构成.反射框架具体颜色定义可以描述为, $C = BusinessOp \cup Resource = s_1 \cup ... \cup s_n \cup res_1 \cup ... \cup res_m$ .

在反射框架颜色分类基础上,基础层具体化位置根据其具体的标识内容确定其颜色域,同样可以类似地确定演化接口的颜色域.在演化接口中,两个特殊的位置是 *EvInt|LoadCfg* 和 *EvInt|Shiftdown*,这两个位置中的标识是没有颜色的.反射框架中位置的具体颜色域描述如下:

$\partial(Reif|ProcStatus) = BusinessOp; \partial(Reif|ResStatus) = Resource;$
$\partial(Reif|BND) = BusinessOp*Resource; \partial(Reif|Cfg) = BusinessOp*Resource;$
$\forall p, p \in \{EvInt|FlushBOperation, EvInt|CurrentBND, EvInt|Ondemand\}, \partial(p) = BusinessOp;$
$\forall p, p \in \{EvInt|CreateBND, EvInt|Destroy, EvInt|UpdateCfg\}, \partial(p) = BusinessOp*Resource.$

通过前面的介绍,ASBS 基础层由业务流程操作通过调用服务资源构建组合服务实现,自适应策略对基础层系统的适应动作主要体现为对业务操作与服务资源之间调用关系的修改.演化接口 *EvInt|CreateBND* /*EvInt|DestroyBND* 分别描述在基础层中添加/删除业务操作与服务资源的调用关系,例如接口位置获得标识

© 中国科学院软件研究所 http://www.jos.org.cn

⟨*S*2,*C*2⟩意味着对基础层添加/删除业务操作 *S*2 与服务资源 *C*2 之间的调用关系;*EvInt*|*FlushBOp* 描述清除基础层中指定业务操作的当前调用,丢弃当前调用服务资源的返回结果,*EvInt*|*FlushBOp* 的实现由基础层模型中的特殊结构保证,其内容在后面的建模中将详细加以介绍;*EvInt*|*LoadCfg*/*EvInt*|*UpdateCfg* 描述了载入 ASBS 配置以及对配置的更新操作,前者将根据当前的配置重新建立业务操作与资源之间的调用关系,而后者则根据指定的配置信息更新到框架位置 *Cfg* 中.关于具体化位置中标识的定义,由于其内容相对复杂并且具有较强的结构相关性,本文将在具体的基础层位置建模中结合基础层模型进行介绍.

反射框架将基础层具体化为具有颜色的位置,为自适应策略提供了对基础层进行操作的接口,对基础层基础化内容调整,从而实现适应行为的建模.在反射框架中基础层具体化的颜色域定义前述已有介绍,而具体化位置的内容则由基础层模型的具体结构决定,下面将详细讨论基础层模型的建模.

## 3 ASBS 性能模型的构建及分析

### 3.1 基础层Petri网构建

在 ASBS 的结构中,基础层主要由业务流程、资源以及业务操作与资源之间的调用关系这 3 部分组成,因此,在对基础层的建模过程中,同样需要对这 3 部分分别建模,并考虑这 3 种模型在反射框架中具体化位置的描述.

#### 3.1.1 流程建模

目前关于应用 Petri 网对业务流程进行建模的研究已经得到了广泛的应用,本文由于篇幅限制,将忽略基础业务流程控制结构 Petri 网建模部分,而详细讨论业务操作的建模以及具体化的实现.在反射框架中,具体化位置 *Reif*|*ProcStatus* 将对基础层业务流程的执行状态进行描述,对应到流程中的业务操作,每次业务操作的执行都将导致该具体化位置中标识的改变.

在对基础层业务流程进行建模之前,先考虑反射框架中基础层流程具体化位置 *Reif*|*ProcStatus* 的标识定义.该具体化位置描述了当前基础层业务流程的执行状态,这种执行状态在本文中描述为具体业务操作是否处于执行中,描述为 $M(Reif|ProcStatus)=\sum_{s\in BService}St(s)\bullet s$,其中 *BService* 为 ASBS 的业务操作集(即抽象服务集),而 *St*(*s*)描述了业务操作 *s* 当前的执行状态.

**定义 1(操作状态函数)**. 操作状态函数 $St:BService\to\mathbb{Z}$,将流程中具体操作的状态映射为整数,在具体化位置中用于描述该操作对应颜色的重数.

在位置 *Reif*|*ProcStatus* 标识定义的基础上考虑业务流程中业务操作的 Petri 网建模.可以构建相应的位置描述流程中业务执行状态,并建立基础层位置与业务操作之间的映射函数,从而实现基础层位置标识到具体化位置标识的转换.基础层业务操作的 Petri 网描述如图 6 所示,其中变迁 *invokeService* 和 *completeService* 分别建模了业务操作 *BOi* 的开始以及完成,业务操作的具体实施将根据资源调用情况由指定资源完成,位置 *ServiceRequire* 和 *ServiceResponse* 分别描述了服务请求的发出以及结果的返回,服务与资源调用关系模型描述将在第 3.1.3 节中具体地给出讨论.位置 *BOi* 描述业务操作 *BOi* 的执行状态,变迁 *invokeService* 的触发将使 *BOi* 获得 *token*,此时,业务操作 *BOi* 处于执行状态.当服务响应返回时,位置 *ServiceResponse* 将触发变迁 *completeService*,完成服务操作并消耗 *BOi* 中的 *token*,由此描述了业务操作 *BOi* 完成并返回未激活状态.

**定义 2(操作位置函数)**. 令基础层业务流程模型中所有 *BOi* 构成集合 *BO*,即 *BO*={*BOi*},则操作位置函数 $\xi_{BSO}:BService\to BO$ 描述了具体操作与业务流程模型中位置的映射关系.

根据图 6 中描述的基础层业务操作模型结构,*BOi* 直接描述了业务操作的执行情况,因此可以考虑一种简单的情况,直接使用 *BOi* 中的标识对 *St*(*s*)进行描述.由此,建立 ASBS 业务操作与基础层业务流程模型之间的关系,并以 *BOi* 中的标识作为业务操作的执行状态,可以得到:

$$St(s)=m(\xi_{BSO}(s)),M(Reif|ProcStatus)=\sum_{s\in BService}m(\xi_{BSO}(s))\bullet s,$$

其中,$\forall p, p\in BO$, $m(p)$为在基础层中位置 $p$ 中的标识.

© 中国科学院软件研究所 http://www.jos.org.cn

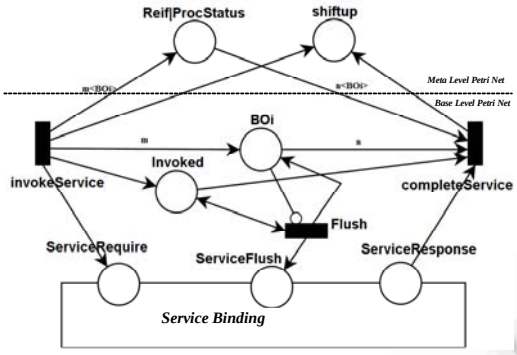


Fig.6 Business operation Petri net model

图 6 业务操作 Petri 网模型

在具体化位置标识定义的基础上,继续考虑业务操作执行状态变化的情况,变迁 *invokeService* 和 *completeService* 的实施均将带来业务操作执行状态 *BOi* 的改变,两个变迁与具体化位置 *Reif|ProcStatus* 之间的弧保证了具体化位置随变迁的实施而改变.当变迁 *invokeService* 实施时由于在业务流程模型中其与位置 *BOi* 之间的弧的权重为 *m*,由此具体化位置获得的 *Token* 为 *m*⟨*BOi*⟩.类似地,当变迁 *completeService* 实施时将消耗具体化位置 *n*⟨*BOi*⟩,本文考虑简单的情况,取 *m*=*n*=1.本文关于 *St*(*s*)由 ASBS 适应策略描述原语的描述来决定,仅考虑了一种十分简单而直观的情况以说明模型构建和具体化实现的方法,在复杂情况下,根据相应的适应策略描述原语需要对具体化实现以及模型构建方法进行调整.

在模型中一个特殊的位置是 *Invoked*,这个位置描述了操作的激活状态.在 ASBS 执行过程中可能会对正在执行的服务进行替换,丢弃已经调用服务的返回结果而重新调用新的服务完成业务操作.这类适应调整动作在策略描述语言中通过 *FlushBOperation* 操作描述,对应其实现在业务操作的 Petri 网利用位置 *Invoked*,*BOi* 配合 *Service Binding* 的特殊结构进行描述.在业务操作 Petri 网模型中,变迁 *invokeService* 的实施描述了业务操作处于执行过程中,此时变迁 *Invoked* 以及 *BOi* 同时获得 *Token*.当适应策略对业务操作进行 *FlushBOperation* 操作,修改 *M*(*Reif*|*ProcStatus*)时将改变 *BOi* 中的标识,从而使得变迁 *Flush* 与 *BOi* 之间的抑制弧失效,由此触发变迁 *Flush*,位置 *ServiceFlush* 获得 *Token* 并将 *BOi* 重置为执行中状态.位置 *ServiceFlush*,*ServiceRequest* 以及 *ServiceResponse* 属于 *Service Binding* 子模型,其具体结构将在第 3.1.3 节中加以介绍.

3.1.2 资源建模

根据 Q-WSDL 中的描述可以将 ASBS 的资源分为网络资源和主机资源两个方面,二者共同影响服务执行.在资源建模的构建中,根据二者各自关注的 Q-WSDL 属性构建相应的资源模型,如图 7 所示.首先,在 Q-WSDL 中,有关网络资源的描述主要由 *NetWork* 描述,变迁 *NetTrans* 描述了信息的传送,其参数值由 *BitRate* 确定.竞争关系的变迁 *PacketLoss* 以及 *PacketSend* 建模了信息的丢包率,其参数值由 *PacketLoss* 确定.在本研究中,假设超时重发的时间由网络延迟确定,即 *Delay*.其次,在 Q-WSDL 中有关主机资源的部分由 *Availability*,*Reliability* 以及 *OperationLatency* 描述,变迁 *Service* 描述了服务器端的请求处理时间,其参数由 Q-WSDL 中 *OperationLatency* 的 *ServiceTime* 属性确定;变迁 *HostFail* 则描述了服务的失效情况,其参数由 *TimeBewteenFailures* 确定;变迁 *HostRepair* 描述服务失效的修复,其参数由 *Availability* 类的 *TimeToRepair* 确定.在本文中考虑主机状态通过两个位置 *HostUpi* 和 *HostDowni* 分别描述主机可用和主机失效,主机失效将导致服务请求处理的延迟等待,通过在主机状态位置与服务处理变迁 *ProcessReq* 之间的禁止弧来描述.令基础层中所有的资源模型的主机可用状态位置集合为 *HostStatus*={*HostUpi*}.本文仅讨论一种简单的资源监测,通过对主机的监测判断资源的状态.

**定义 3(资源状态函数)**. 资源状态函数 $\xi_{SRS}:SResource\rightarrow HostStatus$, 将资源的状态映射为资源状态集中的具体状态,在具体化位置中用于描述资源对应颜色的重数.资源状态集 *HostStatus* 中的状态定义是适应策略相关的,策略对状态进行识别并选择具体的适应动作.

由此,具体化位置 *Reif*|*ResStatus* 中的标识可以描述为 $M(Reif|ResStatus)=\sum_{p\in SResource} m(\xi_{SRS}(p))\bullet p$.

© 中国科学院软件研究所 http://www.jos.org.cn

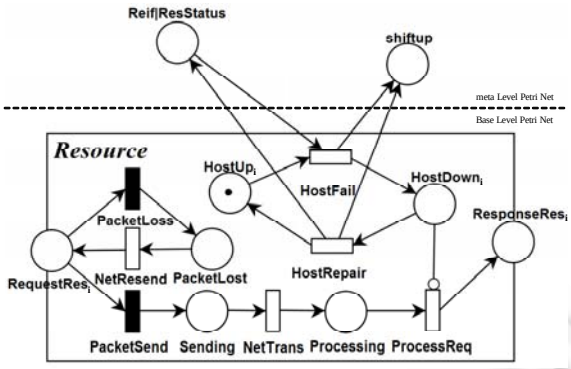


Fig.7 Resource Petri net model

图 7 资源 Petri 网模型

上述内容给出了结合 Q-WSDL 对服务资源性能模型的构建,对资源状态的判断相对简单,针对复杂的资源状态分析,需要结合自适应策略中的相应需求对资源模型进行修改,以提供更详细的描述能力.

3.1.3 业务操作与资源之间调用关系建模

由于在ASBS中假设与不同操作相匹配的资源集是互斥的,因此可以根据资源与操作之间的匹配关系构建固定的模型,本节通过对第 1.4 节所介绍的 ASBS 详例中 *S*2 的资源调用关系建立 Petri 网模型,详细说明 ASBS 中业务操作与资源之间调用关系的建模方法.类似于业务操作,资源模型的构建以及具体化的描述同样对应 ASBS 适应策略中的原语,仅描述了一种简单的情况.图 8 给出了具体的 *S*2 与资源 *C*2,*C*3 以及 *C*4 之间调用关系的 Petri 网模型,模型主要由两个部分构成,图中虚线框内部分利用位置 *BND_S*2 与变迁 *BND_resC*2, *BND_resC*3,*BND_resC*4 之间可修改的弧描述了服务请求的动态选择,其余固定结构的部分则建模了服务响应的接收处理.调用关系模型通过接口位置 *ServiceRequest*,*ServiceFlush* 以及 *ServiceResponse* 与业务流程模型相连接,并通过 *RequestCi*,*ResponseCi* 与具体的资源模型相连接.

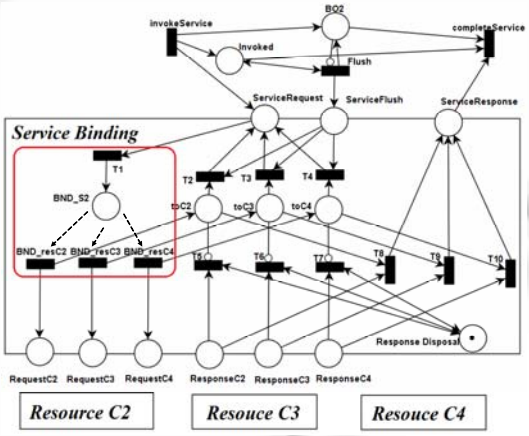


Fig.8 Binding between business operation with resource PN model

图 8 操作资源调用关系 Petri 网描述

位置 *toCi*,*i*=2,3,4 分别描述了服务请求正在资源 *C*2/*C*3/*C*4 中处理的情况.当业务操作被激活时,资源调用接口位置 *ServiceRequest* 获得 *token*,根据具体的资源调用情况选择所调用的资源以触发变迁 *BND_resCi*,*i*=2,3,4,从而使位置 *toCi* 获得 *token*.服务请求的响应根据其对应 *toCi* 中的标识情况触发相应的变迁生成服务执行结果,*ServiceResponse* 中获得 *token* 描述服务请求的正常执行,请求的丢弃则描述 *token* 传递至 *Response Disposal* 中.基于上述过程,*toCi* 中的标识随 *ServiceFlush* 接口标识的改变对服务请求的响应进行处理,而 *ServiceFlush* 接口的标识情况可见第 3.1.1 节中的介绍.

© 中国科学院软件研究所 http://www.jos.org.cn

系统根据资源状态的不同,需要动态地对业务操作请求所调用的服务资源进行调整,选择合适的资源处理服务请求,业务操作与资源之间的调用关系在 ASBS 实例运行的起始阶段,由自适应策略根据 ASBS 配置确定(起始阶段的载入配置被看作是一种特殊的自适应策略),随实例运行其后续调整时同样由自适应策略生成的具体改动指令.业务请求与资源之间的调用关系在反射框架中通过具体化位置 *Reif|BND* 描述,该位置具有二元颜色域 *BusinessOp*Resource*,其标识取值为 $\forall s \in BService, \forall r \in SResource, M(Reif|BND)\langle s,r\rangle = bnd(s,r)$.

**定义 4(资源调用函数)**. 资源调用函数 *bnd*(*s*,*t*)描述了业务操作 *s* 与资源 *r* 之间的调用关系,当二者之间存在调用关系时函数值为正,而当二者之间不存在调用关系时函数值为 0,如式 a)所描述.根据在 ASBS 中的假设,与不同操作相匹配的资源集互斥,由此其调用关系可描述如式 b)所示.

a) $bnd(s,r) = \begin{cases} >0, & \text{exists binding between } s \text{ and } r \\ 0, & \text{no binding exists between } s \text{ and } r' \end{cases}$

b) $\forall s_1, s_2 \in BService, s_1 \neq s_2, \forall r \in SResource,\ bnd(s_1,r) > 0 \rightarrow bnd(s_2,r)=0.$

在调用关系 Petri 网模型中,具体服务请求的处理通过位置 *BND_Si* 到变迁 *BND_Resi* 之间的弧来决定,具体如图 8 所示,*BND_S*2 和变迁 *BND_ResC*2/*BND_ResC*3/*BND_ResC*4 之间的弧可以将业务操作 *BO*2 的服务请求传递到指定的资源接口 *RequestC*2/*RequestC*3/*RequestC*4.

**定义 5(调用关系模型函数)**. 令基础层所有调用关系模型中的位置 *BND_Si* 组成的集合为 *BND_Service*, *BND_Service*={*BND_Si*},而所有的位置 *BND_Resi* 组成的集合为 *BND_Res*={*BND_Resi*},则业务操作调用关系模型函数 $\xi_{Sbnd} : BService \rightarrow BND\_Service$ 与服务资源调用关系模型函数 $\xi_{Rbnd} : SResource \rightarrow BND\_Resource$ 分别描述了业务操作及资源与调用关系模型中上述集合之间的映射关系.

通过位置 *BND_Si* 与 *BND_Resi* 之间的弧描述业务操作与资源之间的调用关系,得到 *Reif|BND* 的标识描述如下:

$$\forall s \in BService, \forall r \in SResource,$$
$$M(Reif|BND)\langle s,r\rangle = bnd(\xi_{Sbnd}(s),\xi_{Rbnd}(r)) = W^{-}(\xi_{Sbnd}(s),\xi_{Rbnd}(r)),$$

其中,$W^{-}(p,t)$ 为位置 *p* 以及变迁 *t* 之间输入弧的权重.

### 3.2 反射适应层模型

反射适应层模型由固定的策略结构和灵活构建具体策略模型两部分构成,策略结构描述了基础层变化激活适应策略进行判定分析并具体执行满足条件的适应策略的过程,策略结构如图 9 所示.在前述介绍中描述了基础层模型与其相应具体化位置之间关联的建模,具体化位置标识变化由 *Shiftup* 位置描述,即具体化位置标识的每次改变都将伴随着 *Shiftup* 位置获得 *token*,由此可以通过该位置触发 ASBS 的适应策略,对基础层状态变化进行适应调整.变迁 *evaluate Guards* 的实施开始时是对各适应策略的判定进行检验,若无满足条件的情况,则通过 *noStragedyChoosen* 位置描述自适应策略重置等待新的 *Shiftup* 触发判定检测.适应策略的判定检测根据自适应策略描述通过调用自适应接口(*OnDemand* 等)来构建 Petri 网描述模型.当判定条件得到满足时,将触发相应适应策略的执行,具体适应动作同样通过调用具体的演化接口构建 Petri 网模型来实现对各具体化位置中标识的调整.适应策略的执行最终通过演化接口位置 *Shiftdown*,完成根据具体化位置中的标识对基础层模型的调整,如前所述,演化接口位置与基础层具体化位置通过反射框架相连接.

根据图 9 中的适应策略结构可以对本文第 1.4 节所介绍的 ASBS 详例中描述的两个自适应策略分别构建 Petri 网描述,如图 10 所示.两个适应策略规则具有类似的判定:*Ondemand*(*S*2)和 *Available*(*C*2),如图 10 所示,这两个判定可以通过演化赋予演化位置 *Ondemand* 和 *Available* 相应的 *token* 实现.根据对演化接口返回信息,在 Rule1 中,当服务 *S*2 处于执行状态且资源 *C*2 失效时将触发后续的适应动作,在 *C*3,*C*4 中随机选择可用的资源替换 *S*2 与 *C*2 之间的调用关系,替换动作包含消除 *S*2 与 *C*2 之间的调用(接口位置 *DestroyBND*)以及创建 *S*2 与 *C*3/*C*4 之间的调用(接口位置 *CreateBND*)两个实现步骤,适应规则最后通过接口位置 *FlushBOp* 丢弃当前调用的结果并重新执行 *S*2.在策略规则 Rule2 中,判定由基础层状态和变量 *C*2*FailureCount* 共同决定,因此,通过位置 *C*2*FailureCount* 对该变量进行描述,基础层状态变化的判定将对位置中的标识进行修改而影响到后续的策略规则.相对于 Rule1,Rule2 主要描述了通过接口位置 *UpdateCfg* 对 ASBS 配置的调整,对当前执行中的实例并不产

© 中国科学院软件研究所 http://www.jos.org.cn

生影响.两个适应策略结束都会通过演化接口 *Shiftdown* 根据具体化位置中的标识重新调整基础层模型,描述了 ASBS 系统基础层根据策略对状态改变所进行的适应响应.

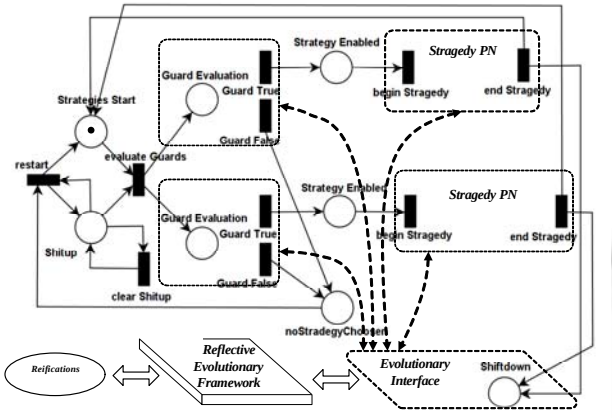


Fig.9 Adaptive strategy structure Petri net model

图 9 适应策略结构 Petri 网描述

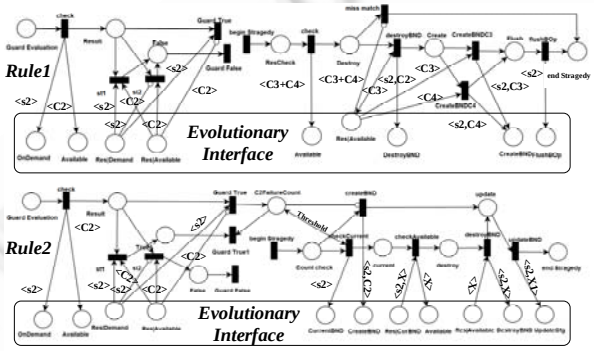


Fig.10 Adaptive strategy Petri net model

图 10 自适应策略的 Petri 网描述

### 3.3 模型分析

基于反射 Petri 网的 ASBS 性能模型,对 ASBS 的各组成部分分别构建相应的模型并在反射 Petri 网框架下将各部分模型相结合,构建 ASBS 的整体性能模型.与传统的基于 Petri 网的性能模型分析方法相类似,对基于反射 Petri 网的 ASBS 性能模型的分析也是建立在状态可达图的基础上的.如前面章节介绍,在 ASBS 的反射 Petri 网性能模型中,适应策略在演化层使用良构网 SWN 进行建模,而基础层系统则通过 GSPN 进行描述,二者之间由 SWN 描述的反射框架相关联.在前述模型介绍中详细讨论了基础层模型与反射框架中具体化位置之间的连接,基础层模型状态的改变将同步地更新对应位置的标识,在演化层适应策略根据基础层具体化位置的标识情况触发相应的规则,利用演化接口对具体化位置中的标识进行调整,并通过 *Shiftdown* 操作根据具体化位置标识调整基础层模型,从而保证了反射 Petri 网模型两个层次的一致性(关于 *Shiftdown* 操作的实现,由于篇幅限制,本文将不作详细介绍).根据上述反射 Petri 网模型原理的描述,下面来考虑反射 Petri 网模型的状态可达图 *rpnRG*,状态可达图的构建方法参照 Capra[11]等人的相关研究.

首先考虑反射 Petri 网模型的状态,由于反射 Petri 网模型包括了基础层模型(GSPN 描述)和演化层模型(SWN 描述),因此,对于反射 Petri 网模型的状态可以定义为两个层次模型状态的组合.下面,我们来考虑反射 Petri 网模型的状态转移,在反射 Petri 网模型中,变迁 *Shiftdown* 是相对其他变迁较为特殊的,由此可以分两种情况讨论反射 Petri 网模型的状态转移规则.对于模型中的任意不是 *Shiftdown* 的变迁 *t*,*t*!=*shiftdown*,变迁的实施规

© 中国科学院软件研究所 http://www.jos.org.cn

则与普通 Petri 网模型中相同(即遵循 GPSN 或 SWN 中的实施规则),并在普通规则下实施得到新的状态即为反射 Petri 网模型的新状态,描述为 $M_i \xrightarrow{\lambda(t_c)} M_j$,反射 Petri 网模型在状态 $M_i$ 下触发变迁 $t$(或颜色变迁实例 $t_c$),得到新的状态 $M_j$,即为反射 Petri 网模型的新状态;对于 *Shiftdown* 变迁,其描述了根据具体化位置对基础层模型的调整,在对反射 Petri 网模型的状态可达图进行分析时,可以简单地将其考虑为一个瞬时变迁,并且 $M_i \xrightarrow{Shiftdown} M_0'$,其中,根据普通 Petri 网变迁实施规则触发 *Shiftdown*,并将基础层模型根据具体化位置描述修改从而得到 $M_0'$ 状态.

资源的复用使得反射 Petri 网基础层中的许多元素在反射适应层中具有一定的相似性,由此通过 SRG 的颜色分类技术可以对系统的状态空间进行一定的压缩.在 SRG 中利用象征表示(symbolic representation)对状态空间中的等价关系进行描述,并通过象征标识(symbolic marking)将同一颜色子类中的相同分布进行分组,生成动态子类(dynamic subclass),在象征标志表示中实际颜色被忽略,仅保存在位置中的分布.根据 ASBS 中的约束条件匹配不同业务操作的资源集合彼此互斥,而在系统的执行过程中,这些资源有着相似的行为特征,因此,在不考虑时间性质的情况下,可以利用这种相似性对资源集合进行静态子类(static subclass)的划分,并在其基础上生成 SRG 的象征标识表示.如第 2.2 节所述,在反射 Petri 网模型中颜色集合被定义为

$$C = BusinessOp \cup Resource = s_1 \cup ... \cup s_n \cup res_1 \cup ... \cup res_m.$$

在此基础上,定义函数 $\psi : BusinessOP \to 2^{Resource}$,描述了业务操作与资源之间的匹配关系,从而有 $Resource = \bigcup_{s_i \in BusinessOp} \psi(s_i)$.以 $\psi(s_i)$ 替换反射 Petri 网标识 $M$ 中每个 $res_i, res_j \in \psi(s_i)$ 得到标识 $M$ 的象征标识 $\overline{M}$.在 $\overline{M}$ 的表示(representation)中,*Resource* 的动态子类数为 $|BusinessOp|$,每个动态子类的核 $card(\psi(s_i))$ 为与 $s_i$ 匹配的资源的总数.关于 SRG 象征标识的表示可参见相关文献[12].根据所得到的 SRG 象征标识的表示,可以应用文献[11]中介绍的算法求得反射 Petri 网模型的象征可达图 *rpnSRG*,并在其基础上进行相应的系统分析.

然而,由于资源的差异性,在上述静态子类的划分中并不能保证静态子类内所有的元素具有相同的时间行为,因此,为了对 ASBS 系统性能进行分析,需要进一步对静态子类中的元素进行划分,在同一静态子类中选择具有相同时间行为的资源构建成新的子类,描述如下:

$$\begin{aligned}&\psi(s_i) = \bigcup_{t \in \Sigma} \psi^*(t),\\ &\forall res_1, res_2 \in \psi(s_i), \lambda(res_1) = \lambda(res_2) \to \exists t, res_1 \in \psi^*(t) \wedge res_2 \in \psi^*(t),\\ &\forall t_1, t_2 \in \Sigma, \forall res_1 \in \psi^*(t_1), \forall res_2 \in \psi^*(t_2), t_1 \neq t_2 \to res_1 \neq res_2,\end{aligned}$$

其中,$\lambda(res)$ 描述了资源 *res* 的时间行为,在 ASBS 中,$\lambda(res)$ 由资源的失效时间和响应时间决定.通过时间行为的划分,$\psi(s_i)$ 被进一步划分成具有相同时间行为子集 $\psi^*(t)$ 的合集,象征标识表示则根据新的静态子类构建,从而生成具有时间一致性的 ASBS 反射 Petri 网模型 *rpnSRG*.关于具有时间一致性的 *rpnSRG* 的性能模型构建以及分析方法可以参考文献[12]中相关的算法介绍.象征标识表示方法的引入利用系统的等价关系,使得 *rpnSRG* 描述的聚合半马尔可夫过程在很大程度上得到了状态空间的压缩,为性能分析的数值解带来显著的效果,然而,象征标识表示取决于系统本身的相似性,对于系统本身结构差异显著的情况,仍然会存在状态空间难以处理的问题.

## 4 相关研究

在近年来的研究中,动态自适应技术逐渐成为软件工程研究中关注的热点,在软件设计中融合自治计算方面成熟技术的研究得到了广泛开展[1,2,13,14].在基于服务的软件系统领域,欧洲的研究项目 WS-Diamond[15]从服务监测以及诊断等方面为自愈服务(self-healing Web service)提供了框架,而美国 ASU 的研究项目 SOD 则提出了 ASBS[4,5],从 QoS 角度出发对自适应基于服务的软件系统进行了研究.系统性能是自适应的主要驱动力之一,然而据我们目前了解,在系统性能分析的研究中尚缺少对 ASBS 系统性能的关注.考虑到 ASBS 的特征与自适应组合服务系统具有紧密的相关性,为此,本文介绍了当前自适应组合服务系统性能分析方法的相关研究,并以此作为本文研究的出发点.

在组合服务性能分析方面,结合其特性,在目前组合服务性能分析中广泛采用基于 MDD[15]的性能分析方

© 中国科学院软件研究所 http://www.jos.org.cn

法,并参考了组件性能评价领域的相关技术[16].文献[17]给出了一种自动的服务选择方法,并将组合服务设计模型转换为性能模型,应用模型检测技术对系统性能进行分析.Ambrogio[18]等人在其先前的研究基础[19]上,借助 Q-WSDL 基于 MDD 研究了组合服务 LQN 性能模型的构建方法.Grassi[7]等人同样通过对其先前的研究成果进行扩展,通过 D-KLAPER 中间语言描述组合服务系统性能相关属性,在组合服务设计模型与系统性能模型之间构建中间模型,并采用基于 MDD 的方法实现模型之间的转换.上述对组合服务的性能分析为后续的自适应组合服务性能分析提供了基础.

在自适应系统性能评价方面,文献[20]对适应性的度量给出了多方面的考虑,将适应作为一系列的动作序列,研究其收益,而文献[21]则提出 QoA,描述实际适应效果与理想适应效果之间的距离.Grassi[22]则提出了应用 D-KLAPER 对自适应组合服务系统的描述方法,并给出基于 MDD 的性能模型构建方法,对自适应组合服务系统的性能以及可靠性进行分析.针对自适应组合服务系统,Palacin[8]等人提出了利用 D-KPLAER 对系统进行描述,并基于 MDD 构建随机 Petri 网性能模型,分别建模系统监测以及适应动作的方法.对自适应组合服务性能分析的研究为 ASBS 性能分析提供了思路,然而上述方法在 ASBS 性能分析中的应用仍面临以下问题.首先,在自适应组合服务系统中,系统的适应策略以及业务逻辑是相混淆的,由此使得性能分析中对适应策略的分析相对困难.而在 ASBS 中,关注适应策略的选择以及资源配置的方案是性能分析中的重要关注点,这也为上述方法的应用提出了挑战.其次,在 Grassi 以及 Palacin 等人的研究中,对系统的性能分析更多地关注于系统自配置(self-reconfiguration),而对系统自愈(self-healing)方面的建模则不够重视,并未考虑系统实例的运行调节,在 ASBS 系统性能评价中表现出相应的不足.

正如 IBM 关于自治系统的描述,自适应系统应划分基础业务和自适应控制两个层次,对 ASBS 的性能分析同样需要从两个层次加以考虑,这样的思想在前述的相关研究中并未得到相应的体现.Capra[9,10]等人的研究从适应策略与业务逻辑相分离的角度出发,提出了应用反射 Petri 网对动态系统进行建模的方法,相对于文献[23]等方法,在保持 Petri 网定义的基础上,通过引入反射的概念实现了静态模型对动态系统的建模方法,为本文关于 ASBS 性能模型构建方法的提出提供了思路.

本文参考 ISTFP6[24]中的分类规则,对 ASBS 的描述结构进行了定义,并给出了具体描述,根据 ASBS 的描述给出了基于反射 Petri 网的 ASBS 性能模型构建方法.相对于目前的自适应组合服务系统性能分析方法,基于反射 Petri 网的 ASBS 性能模型分析方法从业务流程和适应策略两个垂直的角度对系统构建性能模型,以满足 ASBS 对策略和资源配置方面性能分析的需求;并且,在本文的性能模型构建过程中,对系统实例的自愈行为给出了相应的描述,补充了当前研究方法中这一方面的不足,对自适应系统性能评价方法进行了完善.

## 5 总 结

针对 ASBS 系统性能评价问题,本文提出了基于反射 Petri 网的性能分析方法,由 ASBS 结构描述定义出发,详细讨论了性能模型的构建方法及应用.ASBS 的反射 Petri 网性能模型从 ASBS 的业务行为和适应行为两个垂直的角度对系统进行了 Petri 网建模,通过反射框架将两部分相连接构建系统整体的性能模型.本文给出了反射框架的结构和 ASBS 系统各部分模型构建的方法,通过具体化位置以及演化接口连接系统各个组成部分,构建基于反射 Petri 网的 ASBS 性能模型.对反射 Petri 网性能模型,本文给出了其状态可达图 *rpnSRG* 的构建方法,以便于对其同构的聚合半马尔可夫过程进行分析,求解系统性能.

对 ASBS 进行分层次的性能模型构建有助于在设计期对 ASBS 的适应策略、资源选择给出相应的性能以及可靠性等方面的预测,从而指导 ASBS 的设计.同时,相对于目前大多数自适应组合服务性能分析方法,本文提出的基于反射 Petri 网的性能模型分析方法具有很强的策略模型描述能力以及灵活的系统模型构建能力,并且针对目前大多数研究,对实例调整描述方面的不足进行了相应的完善.综合而言,相对于目前大多数建模方法,对于 ASBS 的性能模型构建,本文提出的基于反射 Petri 网模型的性能建模方法具有如下特色:

(1) 关注点的分离.ASBS 的反射 Petri 网建模方法中两层模型从不同的角度对系统行为进行了建模,由此对系统功能以及非功能的关注可以在两个不同的层次分别展开,对具体影响因素的分析限制在相应的层次,从而

© 中国科学院软件研究所 http://www.jos.org.cn

隔离了系统其他行为的干扰.

(2) 结构复用.考虑 ASBS 的高动态性,建模过程中强调对资源的独立建模,由此提供了资源对系统影响的有效分析.同时,资源的相似性又为建模中结构复用提供了基础,通过资源模型的复用直接缩小了基础层模型规模,而在反射适应层,则可以作为颜色归类的依据,一定程度地压缩了状态空间.

(3) 适应行为支持.自适应系统的适应行为根据作用的范围不同可以区分为系统级适应和实例级适应,目前,性能建模方法由于结构相对固定,大多仅对其中一种情况加以讨论.本文提出的 ASBS 反射 Petri 网性能建模方法,通过反射适应层灵活的规则描述以及基础层相应的结构改进,对两级适应行为均提供了良好的支持.

由于系统复杂结构导致的性能模型状态空间爆炸问题是目前大多数性能分析方法难以解决的问题,本文同样面临这样的困境.通过 *rpnSRG* 的象征标识表示,利用动态子类对状态空间进行压缩可以在一定程度上缓解系统状态空间的膨胀,然而,当系资源不具备等价特征的情况下,这种问题仍无法得到解决.为此,在后续工作中将着重关注 *rpnSRG* 的状态空间压缩技术.另外,针对 ASBS 设计情况的不同,对 ASBS 监测器的监测策略的描述以及建模是未来关注的另一部分工作.根据 ASBS 的特征,不同的监测策略同样可能会对适应策略的执行带来影响,为此,可以考虑在 ASBS 的结构描述中引入监测策略的描述并结合反射 Petri 网模型对其建模进行分析,这项工作将在后续展开.

**References**:

[1] Yau SY, Ye N, Sarjoughian H, Huang DZ. Toward development of adaptive service-based software systems. IEEE Trans. on Services Computing, 2009,2(3):247−260.

[2] http://dpse.eas.asu.edu/sod

[3] Cheng B, de Lemos R, Giese H, Inverardi P, Magee J. Software engineering for self-adaptive systems: A research roadmap. Software Engineering for Self-Adaptive Systems, 2009,5525:1−26.

[4] Salehie M, Tahvildari L. Self-Adaptive software: Landscape and research challenges. ACM Trans. on Autonomous and Adaptive Systems, 2009,4(2):1−42.

[5] Cardellini V, Casalicchio E, Grassi V, Presti F, Mirandola R. Towards self-adaptation for dependable service-oriented systems. In: Architecting Dependable Systems VI. LNCS 5835, 2009. 24−48.

[6] Bocciarelli P, D'Ambrogio A. Model-Driven performability analysis of composite Web services. Performance Evaluation: Metrics, Models and Benchmarks, 2008,5119:228−246.

[7] Perez-Palacin D, Merseguer J. Performance evaluation of self-reconfigurable service-oriented software with stochastic Petri nets. Electronic Notes in Theoretical Computer Science, 2010,261:181−201.

[8] D'Ambrogio A. A model-driven WSDL extension for describing the QoS of Web services. In: Proc. of the Int'l Conf. on Web Services. Washington: IEEE Computer Society Press, 2006. 789−796.

[9] Chiola G, Dutheillet C, Franceschinis G, Haddad S. Stochastic well-formed colored nets and symmetric modeling applications. IEEE Trans. on Computers, 1993,42(11):1343−1360.

[10] Capra L, Cazzola W. Self-Evolving Petri nets. Journal of Universal Computer Science, 2007,13(13):2002−2034.

[11] Capra L. A symbolic reachability graph and associated Markov process for a class of dynamic Petri nets. In: Proc. of the 18th Annual IEEE/ACM Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. Florida: IEEE Computer Society Press, 2010. 458−461.

[12] Ardagna D, Ghezzi C, Mirandola R. Rethinking the use of models in software architecture. In: Quality of Software Architectures, Models and Architectures. LNCS 5281, 2008. 1−27.

[13] Grassi V, Mirandola R, Sabetta A. A model-driven approach to performability analysis of dynamically reconfigurable component-based systems. In: Proc. of the 6th Int'l Workshop on Software and Performance. Buenes Aires: Association for Computing Machinery, 2007. 103−114.

[14] Capra L, Cazzola W. A Petri-net based reflective framework for the evolution of dynamic systems. Electronic Notes in Theoretical Computer Science, 2006,159:41−59.

© 中国科学院软件研究所 http://www.jos.org.cn

[15] Kephart JO. Research challenges of autonomic computing. In: Proc. of the 27th Int'l Conf. on Software Engineering. St. Louis: Association for Computing Machinery, 2005. 15−22.

[16] Garlan D, Schmerl B, Cheng S-W. Software architecture-based self-adaptation. Autonomic Computing and Networking, 2009, 1:31−55.

[17] Koziolek H. Performance evaluation of component-based software systems: A survey. Performance Evaluation, 2010, 67(8):634−658.

[18] Ardagna D, Ghezzi C, Mirandola R. Model driven QoS analyses of composed Web services. In: Towards a Service-Based Internet. LNCS 5377, 2008. 299−311.

[19] D'Ambrogio A, Bocciarelli P. A model-driven approach to describe and predict the performance of composite services. In: Proc. of the 6th Int'l Workshop on Software and Performance. Buenes Aires: Association for Computing Machinery, 2007. 77−89.

[20] Reinecke P, Wolter K, van Moorsel A. Evaluating the adaptivity of computing systems. Performance Evaluation, 2010, 67(8):676−693.

[21] Grassi V, Mirandola R, Randazzo E. Model-Driven assessment of QoS-aware self-adaptation. In: Software Engineering for Self-Adaptive Systems. LNCS 5525, 2009. 201−222.

[22] Gjorven E, Eliassen F, Aagedal O. Quality of adaptation. In: Proc. of the 2006 Int'l Conf. on Autonomic and Autonomous Systems. San Hos: IEEE Computer Society Press, 2006. 9−14.

[23] http://cordis.europa.eu/fp6/dc/index.cfm?fuseaction=UserSite.FP6HomePage. 2011.

[24] Cabac L, Duvigneau M, Moldt D, Rölke H. Modeling dynamic architectures using nets-within-nets. In: Applications and Theory of Petri Nets 2005. LNCS 3536, 2005. 731−751.

**葛亮**(1982－),男,辽宁大连人,博士生,主要研究领域为服务计算性能评价.

**刘莹**(1981－),女,博士生,讲师,主要研究领域为服务计算.

**张斌**(1964－),男,博士,博士生导师,CCF 高级会员,主要研究领域为 Web 信息处理,服务计算,数据挖掘.

**李飞**(1975－),男,博士生,讲师,主要研究领域为 Web 服务,网络监测.

© 中国科学院软件研究所 http://www.jos.org.cn