ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.12, December 2007, pp.3015−3030
DOI: 10.1360/jos183015

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

面向服务的企业应用集成系统描述与验证*

张广胜[1,2+], 蒋昌俊[1,2], 汤宪飞[1], 徐 岩[3]

[1](同济大学 电子与信息工程学院,上海 201804)

[2](嵌入式系统与服务计算教育部重点实验室(同济大学),上海 200092)

[3](枣庄农村信用社 合作联社,山东 枣庄 277000)

Specification and Verification of Service-Oriented Enterprise Application Integration System

ZHANG Guang-Sheng[1,2+], JIANG Chang-Jun[1,2], TANG Xian-Fei[1], XU Yan[3]

[1](Electronics and Information Engineering School, Tongji University, Shanghai 201804, China)

[2](Key Laboratory of Embedded System and Service Computing, Ministry of Education (Tongji University), Shanghai 200092, China)

[3](Zaozhuang Rural Credit Cooperative Union, Zaozhuang 277000, China)

+ Corresponding author: Phn: +86-21-69589864, E-mail: zhanggstide@163.com

Zhang GS, Jiang CJ, Tang XF, Xu Y. Specification and verification of service-oriented enterprise application integration system. *Journal of Software*, 2007,18(12):3015−3030. http://www.jos.org.cn/1000-9825/18/3015.htm

Abstract: On the basis of the research findings on service-oriented architecture (SOA), this paper presents a formal systematic SOA analysis, verification and validation methodology called SOARM(SOA reference model) which is the ESB (enterprise service bus)-centric model based on Petri nets and temporal logic. SOARM is consumer-centric, in which the consumers can publish their application specifications/requirements for the service providers to follow when producing or customizing services to support the application. Service interface and enterprise service bus for service realization in service-oriented design are two key parts. When a service is provided or required via the Internet, the semantic consistency becomes the critical issue in the virtualized computing environments. This architecture model tackles the issue by proposing a novel scheme: Petri nets are used to visualize the structure and model the behavior of service architectures while temporal logic is used to specify the required semantic consistency of a service. It is suggested to introduce compositionality in SOA model checking and refinement checking based on the idea of divide-and-conquer, by which the verification task of the whole system is decomposed to several smaller subtasks on the subsystems and shown how to apply it to specify an integrated front-banking system and to analyze its constraints.

Key words: service-oriented architecture; architecture model; integrated front-banking system; temporal logic; Petri net; formal description; correctness verification

* Supported by the National Natural Science Foundation of China under Grant Nos.60534060, 60473094 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2007AA01Z136 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2003CB317002 (国家重点基础研究发展计划(973)); the 2006 Mountaineering Program of Shanghai, China under Grant No.06JC14065 (上海市科委 2006 年度"登山行动计划")

Received 2007-06-09; Accepted 2007-10-26

摘 要: 在对当前面向服务体系架构(service-oriented architecture,简称 SOA)研究的基础上,给出了一个以企业服务总线(enterprise service bus,简称 ESB)为中心的面向服务软件体系架构参考模型(SOA reference model,简称 SOARM),是集 Petri 网和时序逻辑于一体的形式化 SOA 分析、验证和确认方法.基于以客户为中心的面向服务架构设计理念,即根据用户提出系统规范/需求,服务提供者提供服务或组合服务来满足服务消费者,服务接口和 ESB 作为实现面向服务架构的关键部分.虚拟计算环境下,服务语义的一致性验证是十分必要的,SOARM 采用新的模式:通过 Petri 网为服务的行为建模,时序逻辑来描述服务语义一致性约束,综合运用分而治之的精炼检测思想和 SOA 模型检测合成方法,通过对这些子服务性质的检验来验证整个系统的规范.用商业银行综合前置系统说明了如何使用这种方法来实现面向服务的设计.

关键词: 面向服务的体系架构;体系架构模型;综合前置系统;时序逻辑;Petri 网;形式化描述;正确性验证

中图法分类号: TP393 **文献标识码**: A

服务科学[1]希望深度结合 Web 服务和商务,引入商务设计(business design)理念,使得商务运作过程(业务流程)设计与服务组合设计有机融合.作为“服务科学”关键技术之一,面向服务体系架构(service-oriented architecture,简称 SOA)[2]是一种追求敏捷性的面向服务的体系架构,遵循该体系架构所构造出来的应用系统能够适应业务和实现技术的不断变化,有利于软件复用和系统集成.领先的企业正在利用 SOA 降低其应用和 IT 环境的复杂性,促进模块化业务服务的开发,从而创建一个真正灵活和适应性强的 IT 架构.目前,经常运用自然语言、框图以及 UML 等方法描述 SOA 及 SOA 参考模型[2−4],它们简单、易用,但难以支持系统关键性质的严格分析和验证,阻碍体系架构风格在开发人员之间的沟通和交流,因而不利于有效支持服务的复用.面向服务架构起源于构件技术,尽管它们有一些本质的区别,但现有的基于构件的开发技术对服务的实现仍然有效,不足之处是现有构件模型不能让用户最大限度地利用该方法的优点[5].在 SOA 中,系统是按照它应提供的服务来连接多个模块,将不同模块连接起来的主要问题为[5]:

- 缺乏标准语法,而标准语法可以明确(无歧异)地表达服务信息;
- 缺乏标准语义模型,而标准语义模型使得企业可以用一致的语言来表达其业务;
- 缺乏标准的协议,而标准的协议是不同操作环境、不同企业之间消息传递的基础;
- 缺乏绑定事务文档与行为的标准程式.

另外,目前以生产为中心的 SOA(服务提供者-服务注册中心-服务消费者),服务提供者作为中心环节,如代理提供服务注册、服务规范和标准设计易于服务提供者提供服务,缺乏对服务消费者应用规范的发布,而且服务消费者难以定制服务[3].

针对以上问题,我们基于以客户为中心的面向服务架构设计理念,即用户提出系统规范/需求,自顶向下地把系统级规范逐级分解到具体服务规范,从服务提供者提供的服务构件到元服务、子服务,自底向上地组合服务来满足消费者.为此,用 Petri 网和时序逻辑描述服务的语法、语义规范,构造了以客户为中心的面向服务参考模型(SOA reference model,简称 SOARM).虚拟计算环境下,服务语义的一致性验证是十分必要的,本文使用时序逻辑来描述服务契约[4,5],用 Petri 网为服务的行为建模.我们研究了服务与相对成熟的构件技术之间的关系,分析了如何利用基于构件的开发技术来实现面向服务的架构,将服务接口和企业服务总线(enterprise service bus,简称 ESB)作为实现面向服务架构的关键部分,对系统/服务的需求分解为各子系统/子服务的具体规范,通过对这些子服务性质的检验来验证整个系统规范.用商业银行综合前置系统这一实例说明了如何使用这种方法来实现面向服务的企业应用集成系统设计,基于 Petri 网和时序逻辑从形式上对商业银行综合前置系统进行需求规范描述及功能正确性验证.进一步表明我们的验证方法与 Petri 网[6]和时间 Petri 网[7]相比,能够较好地表达事件间的因果关系和时序关系,以及 Internet 环境下某些具有时间限制的基本性质,如最终性(某些事件必须最终发生,某些条件必须最终满足).

1 基本概念

1.1 服 务

根据文献[8],我们给出服务(service)的定义.服务是自治、开放、自描述、实现无关的网络构件.逻辑上看,服务由一对接口和服务的实现体(service implemention)构成.服务接口定义了服务的标识和调用规范;服务实现体具体实现服务的功能.在 SOA,Web 服务已经成为服务的主要实现技术(其他技术如 J2EE,CORBA 和 IBM WebSphere MQ 经过封装实现 WSDL 描述也可以采用)[8].在本文中,元服务被描述为一个服务网(定义 1),当任意元服务被调用时,将相关当前状态输入信息构成的初始条件,表示为一阶谓词逻辑表达式,通过对该表达式真值的判断来确定该服务是否可以被调用.例如,基本服务 a 在系统环境 s 中执行的前提条件表示:$\pi_1\wedge\pi_2\wedge\ldots\wedge\pi_n\wedge input(\varphi_1,s)\wedge input(\varphi_2,s)\wedge\ldots\wedge input(\varphi_n,s)$,其中,$\pi_i$ 为环境 s 需要满足的状态信息(前提),φ_j 为输入信息,只有当该逻辑表达式为真时,服务才能被调用执行.另外,需要注意的是,服务执行前后的状态也会发生变化,也就是说,在一定状态下,某服务执行,其结果不但会增加新的状态,而且会使某些原有状态消失.下面我们给出服务网的定义.

定义 1. 称满足如下条件的十一元组 $SN=(P,T;F,D,V,A_P,A_T,A_F,In,Out,M)$ 为服务网:

(1) P 是库所集表示服务接口,T 是变迁集表示基本服务,$P\cap T=\varnothing$,$P\cup T\neq\varnothing$;$F\subseteq(P\times T)\cup(T\times P)$ 是流关系;

(2) $D=\{d_1,d_2,\ldots,d_k\}$ 是一个有限的个体集;

(3) $V=\{x_1,x_2,\ldots,x_r\}$,$r\leq k$ 是 D 上的变量集;

(4) $A_P:P\rightarrow\pi$,其中,π 是 D 上的谓词集;

(5) $A_T:T\rightarrow f_D$,其中,f_D 是 D 上的公式集;

(6) $A_F:F\rightarrow f_S$,其中,f_S 是 D 上的符号和集;

(7) $In,Out\in P$,$\neg\Box t\in T;\langle t,In\rangle\in F;\neg\Box t\in T;\langle Out,t\rangle\in F$,$In$ 为服务网的初始输入接口,Out 为服务网的最终输出接口,它们在 Petri 网图中用半圆表示,一个服务网的最终输出接口与另一个服务网的初始输入接口必须通过 ESB 连接.

(8) $M:P\rightarrow f_S$,对 n 元谓词 $p\in P$,$M(p)$ 是 n 元符号和.

定义 2. 设 $SN=(P,T;F,D,V,A_P,A_T,A_F,In,Out,M)$ 为一个服务网,$t\in T$:

(1) 若存在一组替换 $\theta:z_1\leftarrow d_1,z_2\leftarrow d_2,\ldots,z_l\leftarrow d_l$,其中,$\{z_i|1\leq i\leq l\}$ 是变迁 t 的输入/输出弧的符号和中以及 t 上的公式 $A_T(t)$ 中出现的自由变量集,$d_i\in D$,$1\leq i\leq l$,满足 $A_T(t)(z_1\leftarrow d_1,z_2\leftarrow d_2,\ldots,z_l\leftarrow d_l)=\text{True}$,并且 $\forall p\in{}^\bullet t,A_F(t)(z_1\leftarrow d_1,z_2\leftarrow d_2,\ldots,z_l\leftarrow d_l)\in M(p)$,则变迁 t 在标识 M 可以以替换 θ 发生,记作 $M[t(\theta)\rangle$.

(2) 若变迁 t 在标识 M 以替换 $\theta:z_1\leftarrow d_1,z_2\leftarrow d_2,\ldots,z_l\leftarrow d_l$ 发生,则得到新的标识 M',记为 $M[t(\theta)\rangle M':M'(p)=M(p)\cup A_F(t,p)(\theta)-A_F(p,t)(\theta)$,称 M' 为从 M 直接可达的.

(3) 若存在序列 $\gamma=t_1(\theta_1)t_2(\theta_2)\ldots t_i(\theta_i)$,使得 $M[t_1(\theta_1)\rangle M_1[t_2(\theta_2)\rangle\ldots\rangle M_i[t_i(\theta)\rangle M_i$,则称序列 γ 是变迁发生序列,称 M_i 为从 M 可达的,记为 $M[\gamma\rangle M_i$,所有从 M 可达的标识构成的集合称为 M 的可达标识集,记为 $R(M)$.

每一个元服务作为一个动作,其动作执行的前提条件是服务前提和输入“状态”的合取.例如,商业银行某元服务 VisaCard Service,其可执行的前提条件表示为一阶谓词的合取:$Hasbalance(v)\wedge Amount_K(x)\wedge CreditCard_K(v)$,其中,个体变元 v 和 x 分别表示“信用卡”和“商品”两个实体,$Hasbalance$ 是服务的前提谓词,而 $Amount_K$ 和 $CreditCard_K$ 分别表示输入参数 $Amount$,$CreditCard$ 的值已知.如果满足上述前提条件,则该服务可执行,执行后会增加两个新状态,即服务的效果 $Debited(v)$ 满足和服务的输出参数 $Authorization$ 的值成为已知;同时,原有的前提 $Hasbalance(v)$ 不再成立.第 3.5 节给出了文件传输应用的两个元服务 $SS1$,$SS2$.

1.2 时序逻辑

时序逻辑[9]是在经典模态逻辑的基础上扩展了 4 个时序算子:○(next)“下一次”、◇(eventually)“最终”、□(always)“总是”和 until“直到”.○p 为真表示在当前状态的下一状态,命题 p 为真;◇p 为真表示存在当前状态的某一未来状态,使得命题 p 为真;□p 为真表示在当前状态之后的所有未来状态,都使得命题 p 为真;p until q 为真表示在命题 q 变成真的第 1 个状态之前的所有状态,都使得命题 p 为真,或者说,从当前状态开始直到命题 q 为

真为止(不包括 q 为真的那个状态),命题 p 总是为真.

定义 3[9−11]. 时序逻辑中的公式由如下几部分构成:由原子命题 $p_1,p_2,p_3,\ldots$组成的集合 P;布尔联结词:$\wedge,\neg$;全称量词和存在量词:$\forall,\Box$;时序算子:$\bigcirc,\Box,\Diamond$,until.

定义 4[9−11]. 时序逻辑公式形成规则:

(1) 一个原子命题 $p\in P$ 是一个公式;

(2) 若 f 和 q 是两个公式,则 $f\wedge g,\neg f,\bigcirc f,\Box f,\Diamond f,f$ until g 也是公式.

逻辑或($\vee$)算子和逻辑蕴涵($\Rightarrow$)算子被作为缩写记号.当需要明确表达算子的作用对象或优先顺序时,公式中也可以使用圆括号().

服务网网结构和时序逻辑公式结合,清晰地描述了服务契约规范和行为,下面给出在服务网系统下的时序逻辑结构定义.

定义 5. 设服务网接口 PORT 是一个原子命题集合,三元组 $\Sigma=(S,R,L)$称为一个服务网系统时序逻辑结构,当且仅当:

(1) $S=\{M\},\{M\}$是 Petri 网的可达集,S 是非空状态集;

(2) R:S 上的二元关系,通过变迁的发生来表示;

(3) L:$M\rightarrow PORT$,$PORT$ 是接口集合,在状态 s 下为真的命题集.

1.3 企业服务总线

目前,ESB 是 SOA 集成中最普遍采用的方法[2,4,8],是一种在松散藕合的服务和应用之间标准集成方式和服务集成基础架构,它提供了服务管理的方法和在分布式异构环境中进行服务交互的功能[8],其主要特点是:

(1) 在 ESB 系统中,被集成的对象明确被定义为服务,而不是传统 EAI 中各种各样的中间件平台;

(2) ESB 明确强调消息处理在集成过程中的核心作用,是提供服务的智能化集成基础架构;

(3) 事件驱动成为 ESB 的重要特征;ESB 很容易通过事件驱动机制向 SOA 的基础架构服务传递信息.ESB 处于整个 SOA 架构的核心地位;完成对原始服务组件的封装、注册、发布、服务访问路由、服务状态控制、错误处理和日志等核心功能[8].

关于 ESB 的实现,Web Services 加上 WS-*协议是目前开放标准的 ESB 实现方式.另外,也可以通过选择成熟的 EAI 中间件来实现服务的集成与互操作性.这样做的好处是开发过程会很顺畅,因为它已经稳定且有丰富的工具支持(例如:IBM 的 Web Sphere MQ5.3,Sun JCA).

2 SOA 参考模型(SOARM)

本文遵循主流观点,即用构件实现服务,服务构件或者是对现有 IT 资源所提供功能的封装,或者需要新建;服务是构成业务流程的单元.服务总线分离了与业务逻辑相关的应用层和与实现平台相关的构件基础设施,遵循该体系结构所构造出来的应用能适应业务和技术变化.图 1 描述了 SOA 模型的层次结构.从上层到下层逐渐由业务向技术实现过渡,分别是:最上层是服务消费者,中间层为业务流程层(business process)和 ESB 层,而最下层是最终的服务提供者(包括服务层、服务组件层和操作系统).各层次分别承担相对独立的职责,以提高资源的可重用性和整个系统的灵活性.

以服务构件等技术支持的软件实体以开放、自主的方式存在于 Internet 的各个节点之上,任何服务可在开放的环境下通过某种方式加以发布,并以各种协同方式与其他服务进行跨网络的互连、互通、协作和联盟(组合),保证集成后的系统设计不违背系统总体需求是十分重要的问题.在 SOARM 的每一抽象层,不仅描述了服务(如元服务、ESB 以及其组合)行为语义,而且也给出了严格的约束语义,即在每一设计层,服务及其组合必须满足层的约束条件,体系架构级的服务集成必须满足客户的总体需求.这样,设计的可追踪性和一致性得到了保证.从横向上看,每一 SOA 层,系统模型可以帮助设计人员对服务进行组合或流程编排.从纵向上看,每一服务可以在被继续垂直分解直至原子服务构件.因此,该模型支持自顶向下和自底向上的系统开发方法.自顶向下的方法是用来把系统级规范逐级分解到具体的服务接口规范和本地 ESB 适配器接口规范;自底向上的方法是通过

组合现存的服务和本地 ESB 规范来实现系统级的服务规范.可见,在 SOARM,无论采用哪种方法,从开始到结束,每一步的服务设计和需求/规范都与系统级的需求/规范保持一致(满足某约束条件不能导致违背其他约束,给定层的服务或子服务的行为模型和语义必须满足该层的行为和语义约束).

在 SOARM,一个业务过程由一些活动组成,一个基本活动对应一个元服务,一个(子)业务过程对应一个组合服务.由于业务过程中的活动是由服务实现的,业务过程就变成了服务流(如图 1 所示).SOARM 的核心部分是其规范模型,包括 3 个基本组成部分:服务模型及规范(服务的描述和实现)、ESB 模型及规范和多个设计层组成的系统约束.遵循 SOA 将外部可见行为与行为内部实现区分开来的原则,服务接口提供了操作设计(服务与 ESB)和架构描述规范的关联.借鉴文献[12−15]的思想,我们用定义在接口的时序逻辑公式来刻画服务约束,所有的约束都通过接口来规范,用户可以把服务和 ESB 当作“黑盒子”,不必知道服务和 ESB 的具体实现.

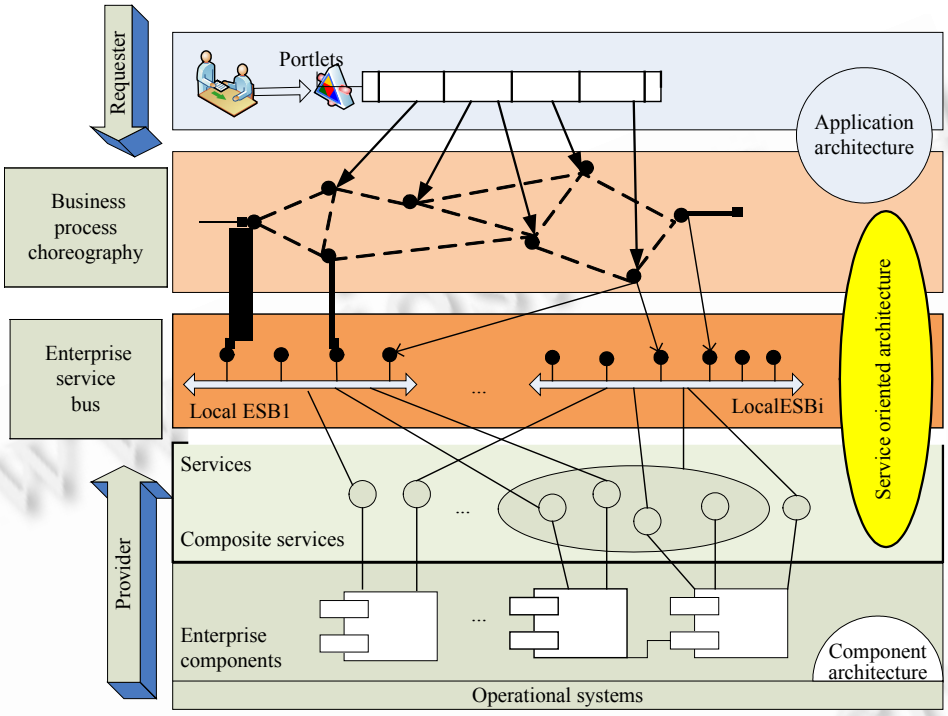

Fig.1 Framework of the SOA reference model

图 1 SOA 参考模型框架

形式上,SOARM 是一个合成的服务流、ESB 与系统约束 CS 的集合,其中,每个组合对应于一个设计层或子结构的概念.$SOARM=(SF,ESB,CS)$,其中:

(1) $SF=\{S_1,S_2,\ldots,S_k\}$,并且对任意 $S_i=(SN_i,Sm_i)$,Sm_i 是服务 SN_i 的服务契约[4,5],我们用一组 CTL 逻辑公式[9−11]来定义;SN_i 是一个服务的行为模型,它通过元服务(或元服务的组合服务)来定义.

$$SN_i.PORT=\{p|p\in SN_i.P\wedge((^{\bullet}p\cap SN_i.T=\Phi)+(p^{\bullet}\cap SN_i.T=\varnothing))\},$$

其中,$SN_i.PORT$ 是服务 SN_i 接口集合,它定义了服务的接口.

契约可以用来提高软件的可靠性[5],我们使用契约描述服务接口的语义.契约把接口同抽象数据模型及其服务功能规范、互操作协议联系起来.对任意契约 c,用 $c.PORT$ 表示作为 c 的原子命题那些端口的集合.这样,对于任意 $c\in Sm_i$,公式 $c.PORT\subseteq SN_i.PORT$ 成立,即服务契约仅用该服务的接口来描述.接口契约提供了接口中每种方法的功能以及与 ESB 交互时需要遵循的协议.

(2) ESB 是一些本地 ESB 的组合,对任意 $localESB_i\in ESB$,$localESB_i=(SCESB_i,SNESB_i)$,其中,$SCESB_i$ 是 ESB

规范,用 CTL 公式来表示 ESB 的功能需求;$SNESB_i$ 刻画 $localESB_i$ 的行为模型用 Petri 网来描述.$localESB_i$ 的会话接口满足如下条件:

$$\bigcup_{localESB_i \in ESB} localESB_i.P \cap \left(\bigcup_{SN_i \in SF} SN_i.P \setminus SN_i.PORT \right) = \varnothing$$

$$\bigcup_{localESB_i \in ESB} localESB_i.T \cap \left(\bigcup_{SN_i \in SF} SN_i.T \right) = \varnothing$$

上述条件表明,ESB 接口只能与服务的接口进行会话(ESB 可以通过适配器来定义自己的接口).

类似地,对于任意 $c \in SCESB_i$,公式 $c.PORT \subseteq localESB_i.PORT$ 成立.通过集成 ESB 和 SF 来定义面向服务系统的整体行为模型.

(3) CS 是系统规范.任意 $CS_j \in CS$ 是一个 CTL 公式,仅用接口作为其原子命题,且满足

$$\bigwedge_{c \in Sm_i \cup SCESB_i \cup CS} c \tag{1}$$

可以类似定义 ESB 层规范和服务层规范.

(4) $\forall S_i \in SF, \forall S_{il} \in SF.S_i$ 即 $S_i=\{S_{i1},S_{i2},\ldots,S_{il}\}$,层映射 $h:S_{il} \to S_j, j \neq i$,使得

$$S_{il}.PORT=S_j.PORT\_EXT \tag{2}$$

$$S_{il}.Sm_{il} \subseteq S_j.CS \tag{3}$$

$S_j.PORT\_EXT$ 为服务组合时那些没有被 ESB 使用的端口集合,式(1)表明所有规范应当相互一致,并且建立了水平约束/规范的一致条件;式(2)建立了结构一致性条件,表明当一个服务精炼为其他服务的子服务(服务组件)时,子服务必须继承原服务的所有接口作为子服务的外部接口;式(3)表明,当一服务精炼为一子服务时,子服务不能违背原服务的行为一致性条件.可见,式(2)和式(3)给出了垂直接口的一致性条件.

3 应用:商业银行综合前置系统

大前置(即本文所述商业银行综合前置系统)、核心帐务处理和数据中心被称为商业银行的三大主线,其他业务以此为基础.目前,商业银行综合前置系统是由最初的银联前置演化而来,其传统的面向应用的构架设计已难以满足不断发展的技术要求和业务需求,主要原因是大前置系统架构设计采用紧耦合设计,业务流程的变更和增加困难,扩展性和灵活性差;各个应用之间有着复杂的"点到点"集成,缺乏统一的数据规范和接口标准,数据在系统之间低效地"穿行",会造成很长时间处理延迟;再者,现有的大前置难以提供统一的前置管理平台和监控平台;大前置连接各个系统,为不同系统之间的通信提供一个平台,随着业务的扩展,如 ATM/POS 联网、电话银行、Call Center、企业和网上银行、手机银行、网上支付、个人外汇实盘买卖、中间代理业务和个人消费信贷等,大前置的整合势在必行,新一代商业银行综合前置系统应运而生.

3.1 面向服务的商业银行综合前置系统

按照 SOARM,在 IBM 企业应用系统整合解决方案的基础上,采用 ESB(WBI message broker)建立某城市商业银行综合前置系统信息整合平台,通过开发接入系统的适配器(adapter)来完成原始系统与整合平台的信息交换,提供整合服务,从而提高企业自身的业务扩展能力,如图 2、图 3 所示.ESB 消息流是商业银行综合前置系统的核心(如图 3 所示),接入系统分为"服务请求方",如银行卡服务以及终端服务(ATM,e-bank 等),以及"服务提供方",如商业银行核心服务两类(如图 2 所示).其消息处理流程为:

(1) 服务请求方通过其前端适配器把原始的请求报文转换为 ESB 系统格式的 XML 报文,写入请求输入队列,交由 ESB 核心系统处理;

(2) ESB 核心系统首先使用 ESB 通用程序包的封装功能来处理传入的 XML 请求报文,按照预先定义的规则将请求报文分解为若干个原子服务报文,并按照顺序控制路由至相应的请求输出队列;

(3) 后端适配器侦听到到达的请求,将请求的 XML 格式的报文转换为后端系统可识别的原始报文,然后

向后端服务提供系统请求服务;

(4) 服务提供系统收到服务请求后,处理请求,并将结果返回;

(5) 后端适配器收到处理结果后将报文格式转换为 ESB 系统格式,返回给 ESB 系统的响应输入队列;

(6) ESB 系统收到响应报文后,判断服务处理的状态,决定是否继续请求新的服务.直到所有的服务处理完毕,合并每一次请求处理结果的响应报文,并返回给响应输出队列;

(7) 前端适配器收到响应报文后,转换为前端系统所能识别的原始报文格式,并返回给前端系统.

图 3 给出了面向服务的商业银行综合前置系统服务模型.本文仅给出了综合前置系统中文件传输系统服务(包括 ESB 消息流)的设计、分析和验证.

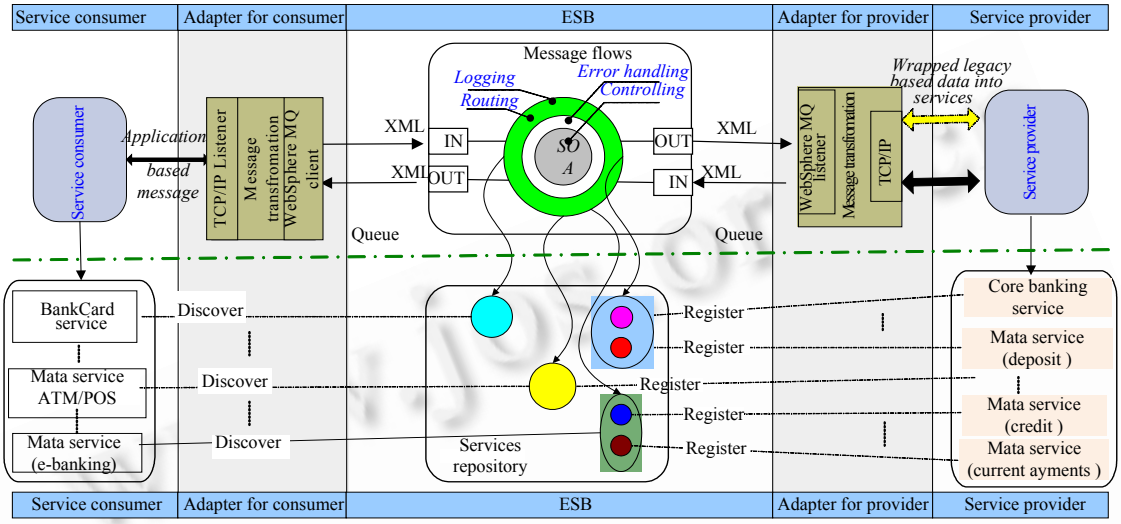

Fig.2 Enterprise service bus connecting diverse applications

图 2 ESB 连接多个业务应用

3.2 文件传输系统服务设计与分析

3.2.1 文件传输系统服务

现有的商业银行系统中,交易文件传输(如报表交易、代发工资、记账服务等)是十分关键的通用服务,它们的设计及正确性验证对系统总体设计来说是非常关键的.与消息传输相比,文件传输有低频次、大数据量和长延时等特点.考虑到架构的完整性,在开发商业银行综合前置系统时,需要制定一套同时包含消息传输和文件传输的总体文件传输系统服务解决方案.目前,商业银行系统中的文件传输方式有两种:一种是把文件分解为报文进行传输;另一种比较复杂,是使用系统调用执行 ftp 命令来传输,本文仅考虑后一种,因此,我们采用类似目前银联大前置的方法在商业银行综合前置系统上设立一个文件传输应用服务(如图 4 所示),负责文件传输.图 4 中的接口说明见表 1.

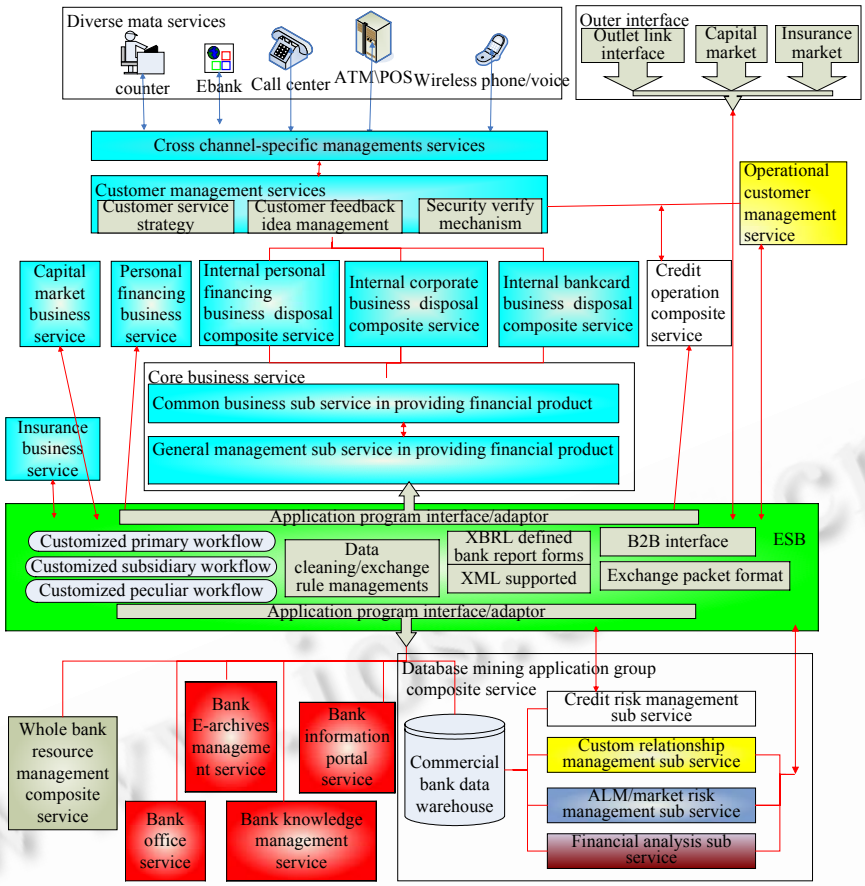

Fig.3 Overall logical service model of the integrated front-banking system (IFBS)

图 3 商业银行综合前置系统的总体逻辑服务模型

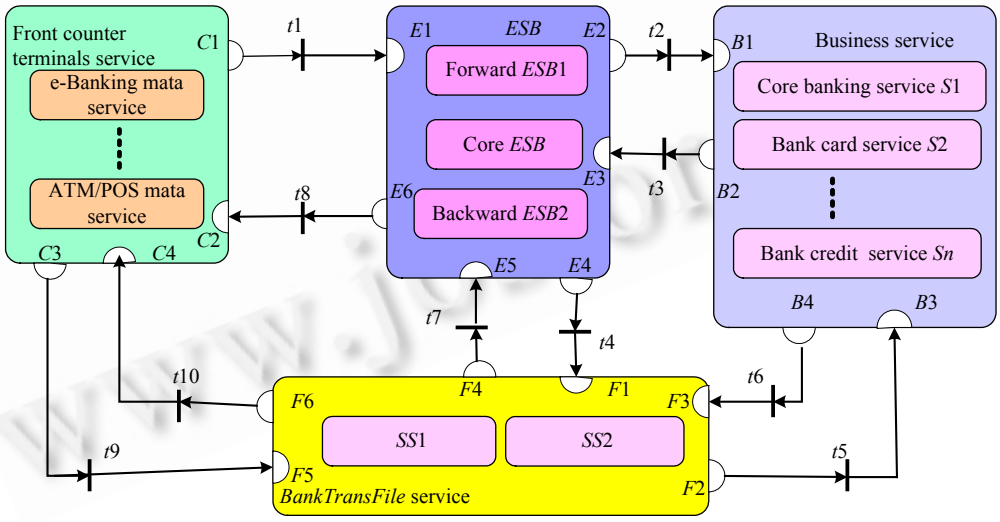

Fig.4 Top-Level architecture of transmission file service

图 4 文件传输服务的顶层架构

Table 1 Legends of partial ports in Fig.4

表 1 图 4 中接口说明

Port	Description	Port	Description
*C*1	Counter ready to send request to IFBS	*B*4	BS send message to start transmit file
*E*1	A message from counter arrived	*F*3	TransFile receive the message of transmit file
*E*2	IFBS transmit the message to BS	*F*4	TransFile send message to MQ of IFBS
*B*1	BS receive the message from IFBS	*E*5	IFBS receive message from TransFile
*B*2	Business system response the request from the IFBS	*E*6	IFBS response the packet to counter
*E*3	IFBS receive the message from BS	*C*2	Counter receive message from IFBS
*E*4	IFBS send the command to ftp server	*C*3	Counter send command to TransFile
*F*1	TransFile receive message from IFBS	*F*5	TransFile receive command from counter
*F*2	TransFile send command to BS	*F*6	TransFile send message of transmitting file to counter
*B*3	BS receive command from TransFile	*C*4	Counter receive message from TransFile

根据图 4,首先给出文件传输服务业务流程编排:

(1) 柜面发出业务请求报文(*t*1);

(2) ESB 接受请求并转发给相关业务系统,如卡系统(*t*2);

(3) 业务系统返回应答报文(*t*3);

(4) ESB 扩展节点(*BackwardESB*2)截获消息流返回的应答报文,检查文件标识位是否为真,若为真,把文件名和业务系统名称信息放入特定 MQ 队列(*t*4);

(5) 文件传输应用程序不断监听 MQ 队列,如果有消息则取出并解析,然后去相应系统取文件(*t*5),文件传输完毕后(*t*6),将文件传输完毕的消息放入 MQ 队列(*t*7);

(6) ESB(IBM MB)扩展节点监听到文件传输完毕信息后,如果需要合并文件,则将文件合并为柜台要求的格式,然后发送应答报文给柜台(*t*8);

(7) 柜台收到应答报文中如果有文件,就可以向文件传输应用程序发送获取文件的名称信息(*t*9),文件传输应用程序根据文件信息向柜台传输文件(*t*10).

3.2.2 系统规范和一致性验证

形式化系统规范的好处在于将面向服务的架构需求转化为具体的架构约束,这些约束是合成子服务必须遵守的.从文件传输服务的系统规范得出:

(1) 若柜面发起交易类文件传输请求报文,那么,它总会从综合前置系统得到业务系统传输文件的响应报文,即

$$\Box(C1(x)\Rightarrow\Diamond C2(y)) \tag{4}$$

(2) 一旦式(4)满足,柜面就将获取文件的信息发送到文件传输应用服务,文件传输应用服务则将相应的文件发送到柜面,即

$$\Box(C3(u)\Rightarrow\Diamond C4(v)) \tag{5}$$

其中,变量 x,y 为消息报文,u,v 为文件变量.要直接验证任意时序公式的一致性是十分困难的[13],因此,按照文献[13,14]给出的算法建立时序逻辑公式与 Petri 网间的映射关系.假定 C 是组成系统的服务集合,根据每个服务 $sf\in C$ 的规范,可以得到一个 Petri 网,这个 Petri 网被称为服务 SF 的约束模型(constraint model),记为 $SFCM(sf)$. $SFCM(sf)$是通过将表示服务约束的 CTL 公式转换为 Petri 网得到的,其中,CTL 公式的原子命题是服务网的 PORTs,所以,服务 sf 的约束与 Petri 网具有相同的 PORTs[14].我们将建立的 $PNs\{SFCM(sf)|sf\in C\}$组合起来,生成一个完整的 Petri 网模型,称为系统的服务规范模型.这个 Petri 网表达了服务约束组合后的模型,如果 PNs 满足系统约束,那么服务约束将与系统约束相一致.图 5 给出了验证系统规范的 Petri 模型,利用 Petri 网的可达性分析技术来验证约束的一致性.

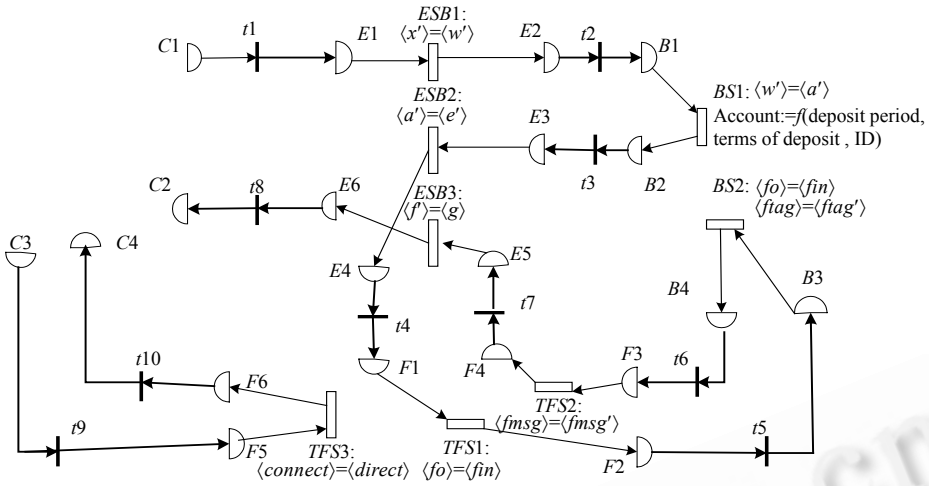

Fig.5 Architecture constraint model file transmission service

图 5 文件传输服务系统约束模型

定理 1. 设组合服务系统的服务网模型为 $PN=(P,T;F,D,V,A_P,A_T,A_F,In,Out,M_0)$,$M_0$ 为初始标识,$R(M_0)$是 M_0 的可达状态集,则 $R(M_0)$必为有限集.

证明:结论是显然的.按照定义 2 标识的定义,$\forall M\in R(M_0),\forall p\in P,M(p)$是 n 元符号和,因为个体集 D 及变量集 V 是有限集,因此,$M(p)$中的元素(个体)数目必为有限多个,其上界为排列 $P^n_{|D|+|V|}$,又因为库所集 P 的元素(谓词)数目是有限多个,显然,有限多个个体在有限多个谓词中的组合必为有限多个,因此,可达标识的数目是有限的,即 $R(M_0)$必为有限集. □

算法 1. 面向目标状态的可达图生成算法.

输入:服务网 SN,初始状态 M_0,目标状态 M_f;

输出:可达图 $RG(SN)$,布尔变量 *reach*.

步骤 1:*reach*←False;

步骤 2:设 M_0 为初始节点,并标记为“新”;

步骤 3:While 存在标注为“新”的节点 do

步骤 4: 任选一个标注为“新”的节点,改为 M;

步骤 5: If 不存在变迁 t 及其替换θ,使得 $M[t(\theta)>$ then

把 M 的标注改为“端点”,返回步骤 3;

步骤 6: For 每个满足 $M[t(\theta)>$的变迁 t 及其替换θ do

计算 $M[t(\theta)>M'$中的标识 M';

If $RG(SN)$中已经存在一个标识 $M''=M'$ then

从 M 到 M''画一条有向弧,并把此弧标以 $t(\theta)$;

Else

在 $RG(SN)$中引入节点 M',从 M 到 M'画一条有向弧,标以 $t(\theta)$;

If $M_f\subseteq M'$ then

将节点 M'标注为“目标”;

reach←True;

Else

将节点 M'标注为“新”;

擦去节点 M 的“新”标注,返回步骤 3.

在算法 1 中,当得到的标识 M' 包含了目标状态 M_f 时,算法就不继续求解 $R(M')$ 了.事实上,对于规范验证而言,M' 就是满足目标的状态,而继续求解其可达状态对于验证是没有意义的.因而,算法 1 得到的可达图 $RG(SN)$ 的节点集是可达标识集 $R(M_0)$ 的一个子集,因此也是有限集,从而也保证了算法 1 是可终止的.分别在两个服务网的初始输入库所 $C1$,$C3$ 中各放一个 token,执行算法 1,可以看到 token 最终分别到达 $C2$,$C4$,式(4)、式(5)成立.

3.3 ESB约束与验证

基于以客户为中心的面向服务架构设计理念,在 SOARM,自顶向下,根据系统级的水平约束条件(根据式(1)),结构一致性条件(根据式(2))和行为一致性条件(根据式(3)),可以导出子系统(服务)的约束条件.限于篇幅限制,我们仅给出 ESB 服务的设计和验证,对于柜面服务、文件传输应用服务和业务系统服务可以类似地分析和验证.由于进入 ESB 和从 ESB 中送出的报文均为 XML 格式的消息报文,其格式比较复杂,总体的报文格式定义如下:

```
〈?xml version=“1.0” encoding=“UTF-8”?〉
〈Service〉
        〈Service_Info〉
        〈service_sn〉〈/service_sn〉
        〈service_id〉〈/service_id〉
        〈requester_id〉〈/requester_id〉
        〈msglogflag〉〈/msglogflag〉
        〈compensation_info〉〈/compensation_info〉
        〈service_name〉〈/service_name〉
        〈timeout〉〈/timeout〉
    〈/Service_Info〉
    〈Service_Status〉
        〈end_timestamp〉〈/end_timestamp〉
        〈start_timestamp〉〈/start_timestamp〉
        〈service_response〉〈/service_response〉
    〈/Service_Status〉
〈Service_Ext〉
    〈/Service_Ext〉
    〈Service_Steps〉
    〈steps currentstep=“1”〉
      〈step id=“1”〉
          〈svc_component_id〉〈/svc_component_id〉
          〈start_timestamp〉〈/start_timestamp〉
          〈end_timestamp〉〈/end_timestamp〉
          〈target_q〉〈/target_q〉
          〈target_id〉〈/target_id〉
          〈timeout〉〈/timeout〉
          〈compensation_flag〉〈/compensation_flag〉
          〈status〉〈/status〉
      〈/step〉
    〈/steps〉
```

〈/Service_Steps〉
〈Service_Data〉
〈request〉
〈/request〉
〈multi_request〉
〈/multi_request〉
〈all_response〉
〈/all_response〉
〈response〉
〈/response〉
〈/Service_Data〉
〈/Service〉

服务报文总体格式由如下几小节构成:

〈Service〉〈/Service〉:是服务报文的根节点;

〈Service_Info〉〈/Service_Info〉:ESB 服务的基本信息描述;

〈Service_Status〉〈/Service_Status〉:ESB 服务的执行状态描述;

〈Service_Ext〉〈/Service_Ext〉:服务请求者与服务提供者之间交互的辅助数据,ESB 程序包不对其做解析;

〈Service_Steps〉〈/Service_Steps〉:ESB 服务的步骤分解控制信;

〈Service_Data〉〈/Service_Data〉:ESB 服务的业务数据域,包含业务请求数据及业务处理结果数据.

为了分析方便,本文用小写斜体字母表示消息报文,如(x),(x'),服务网中用〈x〉标注在弧上表示消息报文,其详细格式不再给出.

下面给出 ESB 规范,即活性性质.

ESB 接受柜面请求并转发给相关业务系统:

$$\Box(E1(x')\Rightarrow\Diamond E2(w')) \tag{6}$$

ESB 截获消息流返回的应答报文,并发送消息给文件传输应用服务:

$$\Box(E3(a')\Rightarrow\Diamond E4(e')) \tag{7}$$

ESB 从文件传输应用服务接受消息并响应柜面请求:

$$\Box(E5(f')\Rightarrow\Diamond E6(g)) \tag{8}$$

这些性质给出了设计 ESB 的具体约束,体现了自顶向下的需求逐级分解方法,根据图 6,采用系统规范验证方法可以验证式(6)~式(8)成立,如在接口 $E1$ 放置 token,执行算法 1,token 最终到达 $E2$,式(6)成立.

图 6 给出了 ESB 服务的行为模型及其内部表示.通过图 6,我们可以分析 ESB 服务的一些行为性质,如活性、可达性和安全性[6]等.$t101$~$t106$ 表示 Adapter 实现遗留系统与 ESB 之间的报文格式及通信协议的转换功能.对于冲正服务($t114$)及订阅/发布服务($t115$),我们将其定义为 CoreESB 的内部子服务.在整个交互过程中,主要包含两个主要的流程:服务请求流程和服务响应流程.

服务请求流程:(1) 请求报文进入 ESB 后($E101$),首先对报文格式及内容进行基本的校验($t107$),校验状态为 $E102$,如果校验不通过($t108$),则转入服务错误处理流程($E104$);(2) 如果报文校验通过($t109$),则进入数据准备流程($E103$),从数据缓存中查询得到必要的服务路由控制信息($t110$),设置服务状态及属性信息($E105$),转入服务控制模块;(3) 在服务错误处理模块($t111$),设置当前的处理步骤及其状态,记录服务流水($E106$);(4) 检查服务的状态是否已经超时,服务完成情况($t112$),服务已经完成($E107$),设置路由($t113$),则响应输出报文($E08$);否则,进入冲正服务($t114$),$E111$ 为服务审核,冲正成功则转入 $t111$,审核错误($t120$)则返回 $E103$.

服务响应流程:(1) 响应报文进入 ESB 后,首先设置其状态信息($E110$),根据服务类型设置路由路径($t116$),根据报文的路由执行相关路由,如原子服务报文进入原子服务响应队列,组合服务报文则进入组合服务响应队

列(*E*109);(2) 发布并审核服务(*t*115),如果服务已经完成(*E*112),则路由至响应输出队列(*t*117),并同时记录流水(*E*106);如果服务失败,则转入(*E*111).

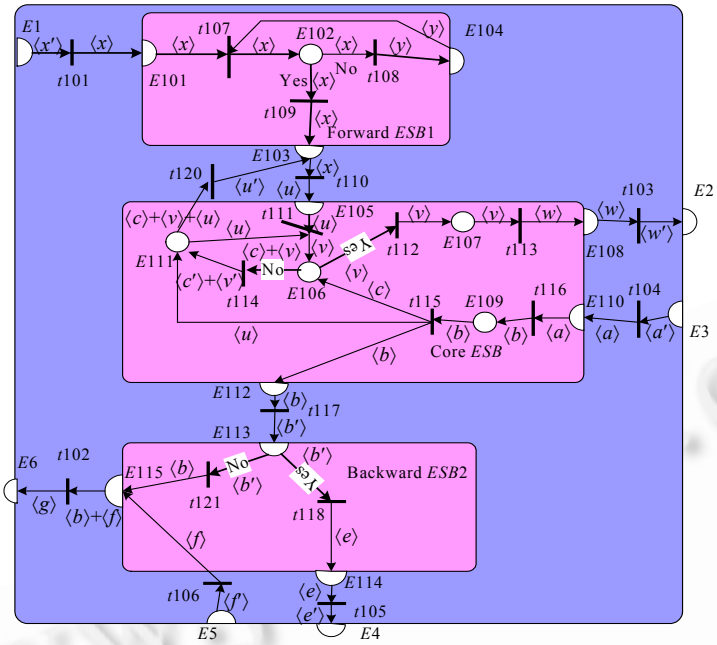

Fig.6 The internal representation of ESB

图 6 ESB 的内部表示模型

Backward *ESB*2 拦截到消息流之后,JavaCompute 节点(*E*113)读取表 TBL_FILETRANS_MSG 的内容并将其映射到共享缓存中;置缓存标志为 true,当拦截到第 2 个消息流的时候,可以根据缓存标志判断是否需要加载数据表到共享缓存;JavaCompute 节点根据表 TBL_FILETRANS_MSG 来得到此交易的交易码/文件名/系统 id/文件传输条件/传输类型,然后,根据以上信息来判断是否需要文件传输:如果不需要传输文件(*t*121),则返回应答报文(*E*115)到柜面;否则,读取系统 id_ip 对应表,根据系统名称得到系统 IP 地址、端口号并将文件名和系统 IP、端口号、传输类型作为新的字段填入原报文结尾(*t*118),然后发送到队列 *Q*1(*E*114).

在整个交互的过程中,服务请求者系统总是向服务请求输入队列提出请求,然后从服务响应输出队列获得请求处理的结果;而服务的提供者系统则总是从服务的请求输出队列中获得对原始服务的请求进行服务处理,然后将结果输出到服务响应输入队列中,报文驱动了各个消息流程的处理.

3.4 子服务之间消息传输规范

为确保合成服务不违背系统需求,验证子服务之间的交互约束(环境约束)是非常有必要的.可以用时序逻辑清晰刻画各个子服务之间的环境约束.对于记帐类交易文件传输有严格的要求,即满足活性和安全性.根据图 4,给出文件传输应用服务和业务系统服务之间文件传输的性质约束:

活性:传输的文件最终被接收,

$$\Box(B4(x)\Rightarrow\Diamond F3(x)) \tag{9}$$

安全性:文件传递是有序的,

$$\Box(B4(x1)\ \text{Until}\ (B4(x2)\wedge\neg B4(x1))\Rightarrow(\neg F3(x2)\ \text{Until}\ F3(x1))) \tag{10}$$

根据图 5,执行算法 1 可以验证式(9)成立,式(10)等价于证明若$\neg B4(x2)$ Until $B4(x1)$,则有$\Box B4(x1)\Rightarrow(\neg F3(x2)\ \text{Until}\ F3(x1))$,则根据时序逻辑公理[9]以及$\langle M,\alpha\rangle\vdash f$ until g 当且仅当对每一 i:$0\le i\le|\alpha|$,$\langle M_i,\gamma_i\rangle\vdash f$;或者存在

某一 i:$0\le i\le|\alpha|$,$\langle M_i,\gamma_i\rangle \vdash g$,且对每一 j:$0\le j<i$,$\langle M_i,\gamma_i\rangle \vdash f$,依据网结构,按照文献[11]给出的方法可以证明,过程略.

3.5 自底向上组合元服务、子服务

采用自底向上设计方法,如图 6 所示,上文给出了 ESB 的内部表示,对于柜面服务、文件传输应用服务和业务系统服务,它们本身也是由多个元服务合成的,如文件传输应用服务(如图 4 所示)是由 *SS*1 和 *SS*2 组合而成.图 7 给出其内部表示,其中,⟨*m*⟩表示消息报文,⟨*f*⟩表示文件变量.一旦所有子服务的行为模型设计完成,我们便完成了整个面向文件传输系统服务的行为模型设计(服务网).并且,该服务网是可执行的,这种设计方法支持增量设计方法.由于 SOARM 接口隐藏了子服务的内部实现细节,因此,根据系统需求可以在相应位置插入某个子服务,从而支持服务精炼设计.需要指出的是,在满足相应约束条件下,我们也可以用一个新的服务来置换原来的服务[13].

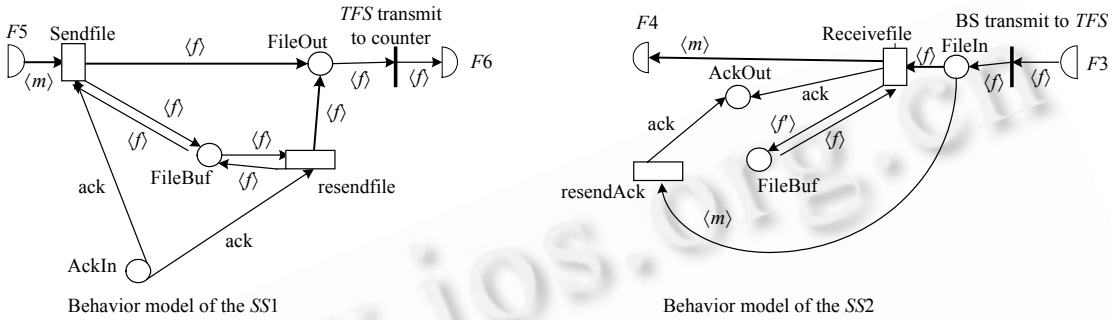

Fig.7 The behavior model of mata services *SS*1 and *SS*2

图 7 元服务 *SS*1 和 *SS*2 的行为模型

4 相关工作

目前,经常运用自然语言、框图以及 UML 等方法描述 SOA 以及 SOA 参考模型[2,4],难以准确刻画服务的语义,也不能完整定义服务之间的交互模式.我们认为,应该使用建模语言和建模方法来支持面向服务设计,将面向服务设计看作是利用面向服务的范例进行模型构造和模型转换的过程.文献[5]构造了一个层次化的 SOA 模型,用 Java 风格的形式语言 rCOS 来描述服务的语法和语义,服务的行为由卫式设计来建模;使用接口来描述服务的语法、契约来描述服务的语义,没有专门引入有关 ESB 的概念.在文献[3]中也提到以客户为中心的面向服务概念,主要从需求发布、发现和协作活动等方面来讨论,难以实现形式化的验证和分析.文献[16]描述了基于 SOA 的网上银行解决方案,与本文类似,也采用了自底向上和自顶向下相结合的策略,其中间层是跨通道架构(cross channel architecture),负责核心业务与前置系统的通信,其顶层是基于 SOA 的业务应用平台(类似前置系统).与本文采用 ESB 作为消息传递的基础不同,该系统集成是以同步和异步消息传递为基础的应用与应用之间的 Web 服务组合,这样容易导致各个应用之间有着复杂的“点到点”集成.该文没有提及系统正确性验证问题.OASIS 提出了 SOA 参考模型[17],其目标是定义最小的一组 SOA 核心概念,并确定其间的关系,用于提供构造具体 SOA 的共同语义且非常抽象,主要用于提供对 SOA 进行共同的理解,没有阐述如何运用这些概念深入地构造 SOA 的基本结构[4,17].基于 SOA 架构,工业界如 IBM[2,18],BEA[19]等侧重技术实现角度从表示、业务过程、服务、ESB 和操作平台这几个层次描述了一个 SOA 参考模型,鲜见从形式上对应用系统进行需求规范描述及功能正确性验证的公开文献.在系统性质验证时,我们给出了改进的可达图[6]的求解算法,而没有采用模型性质检测[20,22,23]技术,这是因为 SPIN 和 SMV 这类模型检测工具只能对有限状态的系统进行检测,而且,由于 SPIN[20]一般用于通信协议的检测和线性时序逻辑模型检测,在验证系统的公平性方面存在不足;SMV[22,23]基于系统的状态可达图,而可达图是根据系统模型初始标识计算出来的,有效性受到模型状态空间爆炸问题的影响.本文给出的服务网模型存在无限库所的情况,如图 7 中的库所 *F*5 的类型是无限的,因为信息报文可以是任意字符[10,13],

所以不能直接使用 SMV,SPIN 等模型检测工具.文献[10]中,在库所有限的情况下,He 等人把一个谓词符号看作一些命题符号的集合,对于无限库所则在不关心信息内容的情况下,将某些无限库所归结为有限库所,然后再采用 SMV 工具进行性质验证,其合理性有待进一步证实.利用 Petri 网做模型检测有如下优点:基于 Petri 网的状态聚合法能够有效地在状态可达图的构造之前简化模型的复杂性;偏序技术和偏序语义技术则提供了一种基于系统语义且极其有效的状态空间的简化算法[21].

5 结 论

基于以客户为中心的面向服务架构设计理念,本文给出用于设计面向服务体系架构的参考模型 SOARM.该模型是以企业服务总线为中心的,用 Petri 网和时序逻辑描述服务的语法、语义规范,分析了如何利用基于构件的开发技术来实现面向服务的架构,将服务接口和 ESB 作为实现面向服务架构的关键部分.把客户的需求分解为各子系统/服务的具体规范,通过对这些子服务性质的检验来验证整个系统的规范.用商业银行综合前置系统说明了如何使用这种方法来实现面向服务的设计,从形式上对商业银行综合前置系统进行了需求规范描述及功能正确性验证.由于客户需求、网络的开放性和动态性导致面向服务的体系架构的演化性和复杂性进一步增强,需要网构软件在运行过程中能够在合适的时刻、合适的场合准确捕捉变化并进行合理的适应性调整,以满足功能和质量的需求.具体表现为服务实体的自适应性和 SOA 的自适应性,这些问题是我们下一步工作的重要内容.

References:

[1] Chesbrough H, Spohrer J. A research manifesto for services science. Communications of the ACM, 2006,49(7):35−40.

[2] Arsanjani A. Service-Oriented modeling and architecture: How to identify, specify, and realize services for your SOA. Whitepaper from IBM, 2004. http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/

[3] Tsai WT, Xiao B, Huang Q, Chen Y. Collaborative software design in an SOA environment. Science in China (Series F), 2006,49(6):821−842.

[4] Ma ZY, Chen HJ. A service-oriented architecture reference model. Chinese Journal of Computers, 2006,29(7):1011−1019 (in Chinese with English abstract).

[5] Liu J, He JF, Liu ZM. A strategy for service realization in service-oriented design. Science in China (Series F), 2006,49(6):864−884.

[6] Murata M. Petri nets: Properties, analysis and applications. Proc. of the IEEE, 1989,77(4):541−580.

[7] Berthomieu B, Diaz M. Modeling and verification of time dependent systems using time Petri nets. IEEE Trans. on Software Engineering, 1991,17(3):259−273.

[8] Papazoglou MP, van den Heuvel WJ. Service oriented architectures: Approaches, technologies and research issues. The VLDB Journal, the Int'l Journal on Very Large Data Bases, 2007,16(3):389−415.

[9] Manna Z, Pnueli A. The Temporal Logic of Reactive and Concurrent Systems Specification. New York: Springer-Verlag, 1992.

[10] He XD, Yu HQ, Shi TJ, Ding JH, Deng Y. Formally analyzing software architectural specifications using SAM. The Journal of Systems and Software, 2004,71:11−29.

[11] Du YY. Modeling and analyzing electronic commerce systems using Petri nets [Ph.D. Thesis]. Shanghai: Tongji University, 2003 (in Chinese with English abstract).

[12] Wen YJ, Wang J, Qi ZC. Compositional model checking and compositional refinement checking of concurrent reactive systems. Journal of Software, 2007,18(6):1270−1281 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/1270.htm

[13] Deng Y, Wang JC, Tsai JJP, Beznosov K. An approach for modeling and analysis of security system architectures. IEEE Trans. on Knowledge and Data Engineering, 2003,15(5):1099−1119.

[14] Wang J, He X, Deng Y. Introducing software architecture specification and analysis in SAM through an example. Information and Software Technology, 1999,41(7):451−467.

[15] Heineman GT, Councill WT. Component-Based Software Engineering, Putting the Pieces Together. Boston: Addison-Wesley, 2001.

[16] Shan TC, Wachovia C, Hua WW. Service-Oriented solution framework for internet banking. The Int'l Journal of Web Services Research (JWSR), 2006,3(1):29−48.

[17] OASIS. Reference model for service oriented architecture 1.0. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm-cs.pdf

[18] IBM Software Group. WebSphere process server V6.0 WebSphere integration developer V6.0. IBM Proof of Technology. 2005.

[19] BEA. The enterprise service bus: Building enterprise SOA. http://dev2dev.bea.com/pub/a/2004/12/soa_ibarra.html

[20] Holzmann GJ. The model checker SPIN. IEEE Trans. on Software Engineering, 1997,23(5):279−295.

[21] Jiang YX, Lin C, Qu Y, Yin H. Research on model-checking based on Petri net. Journal of Software, 2004,15(9):1265−1276 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1265.htm

[22] SMV Group. http://smv.unige.ch/tiki-index.php

[23] The SMV system. http://www.cs.cmu.edu/~modelcheck/smv.html

附中文参考文献:

[4] 麻志毅,陈泓婕.一种面向服务的体系结构参考模型.计算机学报,2006,29(7):1011−1019.

[11] 杜玉越.电子商务系统的 Petri 网建模理论与分析技术研究[博士学位论文].上海:同济大学,2003.

[12] 文艳军,王戟,齐治昌.并发反应式系统的组合模型检验与组合精化检验.软件学报,2007,18(6):1270−1281. http://www.jos.org.cn/1000-9825/18/1270.htm

[21] 蒋屹新,林闯,曲扬,尹浩.基于 Petri 网的模型检测研究.软件学报,2004,15(9):1265−1276. http://www.jos.org.cn/1000-9825/15/1265.htm

张广胜(1975－),男,山东济南人,主要研究领域为网格计算,网络安全,语义网格,Petri 网.

汤宪飞(1983－),男,主要研究领域为 Web 服务组合与验证,Petri 网理论及应用.

蒋昌俊(1962－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网格计算,语义网格,Petri 网.

徐岩(1982－),女,主要研究领域为会计学,计算机金融.