Web服务合成研究综述*)

石 静 丁长明 赵泽宇 薛向阳

(复旦大学计算机科学与工程系 上海200433) (复旦大学智能信息处理开放实验室 上海200433)

摘 要 Web服务的发展将Web应用从信息交互领域扩展到了服务交互领域。Web服务诸多的特性使得它非常适合于商务应用集成,因此工业界和学术界都希望能够通过合成多个Web服务从而获得增值的新服务。在服务合成的过程中,不仅需要语义清晰的服务描述语言和直观的建模方法,并且应该能够动态地发现构件服务,顺利地执行复合服务以及对其进行事务处理。此外对Web服务合成的分析评估也将有助于理解复合服务中每一项活动的行为,从而促使不断改进服务合成技术。

关键词 Web服务,Web服务合成,动态服务发现,Web服务事务处理,性能分析

Overview of Web Services Composition Research

SHI Jing DING Chang-Ming ZHAO Ze-Yu XUE Xiang-Yang

(Dept. of Computer Science and Engineering, Fudan University, Shanghai 200433)

(Laboratory of Intelligent Information Processing, Fudan University, Shanghai 200433)

Abstract The development of Web Service has extended the role of Web applications from information interaction to service interaction. As we know, Web service technology is suitable for business applications integration. Therefore, researchers both from industry and academy try to compose individual Web services to obtain a new value-added service. In the process of service composition, we not only need semantically clear service description languages and intuitive modeling methods, but also want to have abilities of dynamic service discovery, service execution and transaction processing. Furthermore, performance analysis for Web services can help implementers understand the behavior of every activity in a composed process and improve the service composition technologies.

Keywords Web service, Web services composition, Dynamic service discovery, Web service transaction processing, Performance analysis

1. 简介

从上个世纪90年代到本世纪初,Web应用主要围绕Internet上的数据展开。在几乎所有的应用中,这种面向数据的应用总是由使用者通过使用Web浏览器等工具的方式完成。XML技术以及其他相关技术的快速发展改变了Web应用发展的格局:它将Web应用从信息交互的领域扩展到了服务交互的领域。这种服务交互就是当前引起工业界和学术界关注的Web服务(Web services)。

Web服务是自包含、自描述的模块化应用;它提供了从简单请求到复杂商务处理的功能;一旦Web服务被部署,其它的应用(包括Web服务)就能够发现并调用所部署的服务[22]。从技术角度来看,Web服务描述了一系列操作的接口,它使用标准的XML消息传递技术封装信息,并可经由网络访问这些接口和操作,完成特定的任务。服务实现与服务接口的分离,促使基于Web服务的应用成为松耦合、面向构件、跨技术的实现[33]。

正是由于Web服务上述的特性,使得它非常适合于当今商务应用的集成。然而一个单独的Web服务很可能限制其所拥有的能力,所以工业界和学术界都希望能够通过合成现有的Web服务来创造出新的服务功能。同时,B2B(business-to-business)商务的发展也使得Web服务合成获得了空前的关注。通过集成多个服务从而得到增值的新服务,这将是一个不容错过的商机。

所谓Web服务合成,指的是从互联网中选取相对简单、可用的Web服务并将它们组合成新服务的技术[3,6]。合成后的新服务被称为复合服务;用于合成复合服务的子服务称之为构件服务。Web服务合成大致可以分为两种类型:静态合成和动态合成。在设计阶段就定义了复合服务规范的合成方法是静态合成;相对地,如果在运行时刻所需服务才被选择和调用的服务合成方法属于动态合成。

Web服务技术,如统一描述发现集成(Universal Description, Discovery, and Integration, UDDI)[24],简单对象访问协议(Simple Object Access Protocol, SOAP)[25],Web服务描述语言(Web Service Description Language, WSDL)[26]都是用于描述、发布、发现和调用Web服务的基础标准。但是它们都未曾解决Web服务合成的问题。目前,IBM的Web服务流语言(Web Services Flow Language, WSFL)[14]、微软的"交换语言"(Exchange Language, XLANG)[23]以及业务流程执行语言(Business Process Execution Language for Web Services, BPEL4WS)[5]均支持对多个服务的请求。

Web服务合成必须满足一定的要求,这些要求也成为了Web服务合成中的研究难点。它包括:(1)能够动态地发现满足需求的服务;(2)能够顺利地执行复合服务;(3)能够对复合服务进行事务处理。高度动态的商务应用环境使得Web服务合成应该具有高可用性,高可靠性和高度自适应性[18]。

*)基金项目:国家863计划(2002AA103065)、上海市政府科技发展基金(01QD14013,015115044)。石 静 硕士研究生,主要研究方向为Web服务和语义Web;丁长明 硕士研究生,主要研究方向为Web服务和语义Web;赵泽宇 硕士研究生,主要研究方向为Web服务和语义Web;薛向阳 教授,博士生导师,主要研究方向为多媒体信息处理和检索技术。

本文就目前的Web服务合成技术进行全面的介绍。首先会介绍服务合成的标准基础和理论基础。在第3部分，服务合成中的关键技术将是关注的重点；与此同时本文也会简单介绍一些现今较为著名的服务合成系统，以期希望能够从中洞悉Web服务合成发展的新方向。在文章的最后将会探索如何对服务合成进行分析评估。

2. Web服务合成概述

2.1 Web服务合成的分类

Web服务合成大致可以分为两类：静态合成与动态合成[6,12]。

在静态合成中，复合服务在设计阶段就被定义。合成过程对于服务请求者来说并不可见；服务请求者可以像调用基本服务那样调用复合服务。比如一个飞机订票系统，它应包括可用机票查询、信用卡信息查询、更新数据库并将机票预定给顾客这三个功能截然不同的服务。由于这项复合服务的使用频率会很高，服务提供者若将这三个服务事先集成在一起要比在每次客户请求时都创建一次复合服务更符合应用的需要。

动态合成指的是，在运行时刻选择和调用所需服务并将之合成为一个复合服务的过程。例如一个商务旅行管理系统，它通过与机票管理、酒店管理、路线管理、租车管理等系统的协作为客户安排旅行事宜。像这样的一个复合服务不可能事先就定义好所需的构件服务以及它们之间的工作流，而是应该依赖特定的应用域选择相应的构件服务并与之交互。

将一个复合Web服务设计成动态的还是静态的，这取决于复合服务的性质及其应用域。如果复合服务本身是希望与商业伙伴之间有一个相对固定的关系，并且它的构件服务很少进行改动，那么静态合成就会比较合适。相对地，动态合成就适合于那些对特定复合服务请求较少，同时它的构件服务也不十分稳定的情况。不过，动态合成可以更好地利用构件服务的当前状况，并根据实时参数(如带宽、执行花费等)在运行时进行优化。

2.2 Web服务合成语言

在工业标准中，WSDL[26]被用来描述单个Web服务的详细信息。然而WSDL并不支持序列化多个Web服务的调用或指定某个Web服务不同操作的调用次序。目前IBM发布的Web服务流语言WSFL[14]以及微软发布的XLANG[23]是两个最早定义Web服务合成的语言。两者都建立在WSDL之上。

WSFL描述了如何组织或协调一系列的Web服务调用，使它们成为一个整体的工作流或业务流程。WSFL可以被认为是基于图论的过程建模的方法。它既支持复合服务的静态配置，也支持在Web服务注册中心内动态查找构件服务。

XLANG是微软BizTalk[34]用于定义业务流程的语言。XLANG主要描述了一个角色收发讯息的内容和顺序，以及整个系统中所有角色之间的交互。但是XLANG要求复合服务是静态配置的。

BPEL4WS[5]是又一种指定业务流程和业务交互协议的语言。它吸取了WSFL与XLANG两者的长处：将WSFL基于图论的流程描述和XLANG基于结构化的构造过程结合在一起，力图成为Web服务合成的统一化标准。

Web服务合成的工业标准着重于构件服务之间业务逻辑的描述和控制，通过过程建模的方法来实现构件的重用和服务的合成。与工业标准相对应，学术界的研究工作也在不断进行。Web服务标记语言DAML-S(DARPA Agent Markup Language-Service)就是其中的一项研究成果。DAML-S[10]是基于DAML+OIL[9]的Web服务本体(service ontology)发展而来。它由DAML服务委员会提出的一套标记语言结构来描述Web服务的属性和服务的能力。

Web服务合成语言属于服务描述栈最高层次的规范[33]。它们的优越之处在于：它们均能够组合或协调其他Web服务，从而创建新的、高层的Web服务。创建更多的Web服务也就意味着可以采用更多的方式将它们组合成为功能更为强大的Web服务。这样就实现了网络效应，使得Web服务成为网络应用研究的一个新的领域。

2.3 Web服务合成的理论基础

Web服务合成可以用多种方法来进行建模。其中，有向图、状态图和Petri网是最常用的三种工作流建模方法。

有向图被广泛用于业务流程建模。它通过活动节点、控制链和数据链这三个元素来表示一个复合服务。活动节点代表了复合服务中的各项子任务；控制链定义了各构件服务之间的依赖关系；数据流位于控制流之上，描述了商业文档如何在构件服务之间流动。

状态图侧重于描述系统状态的变化，它具有分析复合服务所必须具备的语义。在已经存在的工作流规范语言中，状态图提供了最为丰富的控制流结构(分支、并发、结构化循环等)[3]。

Petri网是一种图形化语言，并且具有形式化的语义定义。一个Petri网模型加上相应的语义就能描述一个业务流程。由于Web服务的行为基本上是操作的一个偏序集(partially ordered set)，所以可以直接将Web服务映射到一个Petri网上。操作对应于变迁元素(transition)；服务的状态对应于位置元素(place)。变迁元素和位置元素之间的有向箭头用来定义两者的因果关系[19]。Petri网能够同时显式描述系统的状态和事件，这有利于研究者对系统进行分析和测试。

3. Web服务合成中的关键问题

除了直观的、易于表达的、语义清楚的描述语言是简化建模和匹配复合服务的重要因素之外，另一方面，在运行时的动态服务发现，服务执行和事务处理同样也是Web服务合成的关键所在。

3.1 动态服务发现技术

如前所述，动态的服务合成能够充分利用网络资源，实时地为服务请求者搜索满足需求的Web服务。同时人们期待拥有更加灵活的商务应用环境为其创造出更好的商机。自然地，动态服务合成成为了人们所关注的重点。

与静态服务合成相比，动态服务合成具有更高的复杂性。动态的服务合成取决于服务发现过程是否能够自动实现。自动发现的Web服务不仅要在功能上满足要求，还有一些其他特殊的服务要素，例如成本、花销、运行时间等都有可能需要被考虑在内。

当服务请求需要满足一定要求的时候，服务双方必须达成一定的共识。比如服务提供者应该保证在最终期限之前执行完成复合服务，并且只有满足条件的商业文档(数据流)才是正确的。然而当服务提供者将自己的服务的一部分代理给第三方的时候，问题就由此产生。因为服务提供者必须保证其代理者也能够在规定期限内完成服务的执行，但是在实际情况中很有可能做不到这一点。解决这个问题的一种方法是让服务提供者和服务请求者双方以及代理者和被代理者之间达成双边协议。他们在合同中要指定合同参数，即条件，例如满足什么样的价格、期限、质量的服务才能被调用[4,13]。为了使整个过程更加容易地进行，服务提供方可以事先准备好他愿

意接受的合同类型。在这样的一个合同模板[4]中，有些合同参数是可以协商的，而有些合同参数则是固定的。

因此，为了促使服务的自动发现，服务提供者应该提供合同模板来描述他们的服务。模板不仅要写明服务功能方面的信息，还要有技术方面的情况，例如所要求的文档格式等。同时，服务请求的描述也应包括控制流和数据流，以便用于查找合适服务的过程。接着可以根据比较模板和服务请求的结果来决定哪一个 Web 服务是最佳服务。那些被选中的服务模板将被返回给请求者，然后服务请求者可以与服务提供者商议服务调用条件[13]。

目前采用上述方法实现服务自动发现的系统有 eFlow，AgFlow 等。eFlow[6,7,18]是 HP 实验室开发的电子商务服务合成系统。它的动态服务发现代理(broker)能够及时发现新启动的服务。如果该服务比原先的更适合正在运行的服务流程(service process)，eFlow 引擎就能够将此新服务插入其中。这使得复合服务更具有自适应性。AgGlow[28]是为基于 Agent 的服务发现而设计的系统。如果构件服务运行的时间超过估计，它可以通过实时更换服务提供者来完成执行任务的剩余部分。

另一种实现动态服务发现的思想是借助于包含语义的标记语言来统一描述异构服务。一个好的服务发现架构是应该有能力对 Web 服务描述进行语义层上的推理，进而找到合适的 Web 服务。

当前工业上对 Web 服务检索的方法只有关键字查询和分类查询这两种，基于这些方法的服务检索可能会返回过多的结果以致于无法对所需的 Web 服务进行有效地选择。因此研究者们希望通过在服务描述中引入包含上下文信息的语义描述来提高服务检索的能力，使得服务请求者和提供者具有共同的语义理解，从而驱动服务的动态发现和调用[1,20,21]。

为 Web 服务添加语义信息的方法一般有两种：(1)通过扩展工业标准的 Web 服务接口描述语言来添加相应的语义信息。这种做法主要是把 WSDL 中扩展描述操作的描述、前置条件和后置条件的元素及属性映射到 DAML+OIL 本体上。(2)直接使用资源描述框架(resource description framework，RDF)和本体对网络服务接口描述和注册信息进行置标[15]，例如使用 DAML-S 语言。和 WSDL 相比，DAML-S 不仅说明了构件服务之间，服务与客户之间是如何发送信息的，还说明了发送什么，为什么发送等语义方面的内容。

这两种方法各有其优缺点。通过扩展 WSDL 包含语义的方法可以较为方便地应用到当前的工业标准中去。WSDL 灵活的扩展性使服务提供者可以根据自身的需要，选择在服务接口描述中添加或删除语义信息。它的缺点是仅能提供部分的语义描述，因此不能很好地支持自动服务合成。DAML-S 置标解决了 WSDL 不具有或通过扩展 WSDL 后只是部分具有语义信息的问题。但是由于 DAML-S 对服务的描述是抽象的，目前的 UDDI 机制还不能有效地处理这种抽象的语义信息，所以必须把 DAML-S 的描述映射到具体的服务描述上，服务才能被使用，但是映射过程中又面临 DAML-S 语义损失的问题，因此如何将 DAML-S 具体化将是一个新的研究课题。

3.2 Web 复合服务执行技术

复合服务的执行是指按照事先定义好的顺序逐个调用构件服务的过程。一般有两种模式实现这个过程：中央调度和分布式调度[4,13]。

中央调度模式是在服务提供者的控制之下执行 Web 复合服务的。在这种模式中，服务提供者持有一个复合服务调度程序。它根据复合服务的定义指导自己的行为。当复合服务被调用时，一个调度程序的实例就被创建，它按照一定的顺序和一定的条件调用各构件服务[13]。运行中的调度程序将接收和处理与服务实例相关的事件(特别是终止服务信号和错误信号)。为了使服务实现和服务接口分离，调度程序可以使用外部交互网关(External Interactions Gateway，EIG)[4]来处理进出的服务请求。这样，远程过程调用和其他分布相关问题都将被代理给这个软件实体。至于复合子服务(即指构件服务是复合服务)，将会由另外一个调度程序来执行它。目前 eFlow，AgFlow 等系统都采用中央调度的模式来执行复合服务的。

中央调度程序把所有的复合服务的执行控制权交给了服务提供者，这是个非常简单并且直观的方法。然而，中央处理的模式会带来许多新问题。例如，当每一个构件服务被调用的时候，调度程序至少要处理一组请求/确认信息。当巨大数量的服务同时需要控制的时候，调度程序与构件服务之间的通信将成为一个瓶颈。

消除这个瓶颈的方法是以最少的通信方式来执行复合服务。在分布式调度模式中，复合服务仍旧由自己的调度程序来控制。所不同的是，为了能够在对等环境下(peer-to-peer)执行复合服务，系统将使用“协调者”(亦可称为协调程序)来包装每个构件服务。协调者能够接收从其它协调者发来的消息并在必要的时候调用它的构件服务[4,13]。复合服务的完成或中断将依赖于所有相关的协调者。同时协调者还应该有能力为每个构件服务定义前置条件和后处理行为。当前仅当前置条件被满足时，协调者才会调用该构件服务，并且通知所有负责后继服务的协调者，让它们做好准备。

Self-Serv[2,3]是目前较为著名的采用对等服务协调模型的服务合成系统。它利用状态图来表达复合服务操作的业务逻辑；同时也引进了服务社区(service community)的概念来构架大规模动态的复合服务。协调者负责初始化、控制、监控相关构件服务，并与它们所在的端点(peer)合作来管理服务的执行。在复合服务的执行过程中，协调者所需的知识将从服务的状态图中获得并以一种表格化的形式表示。这样，协调者就不需要实现复杂的服务调度算法了。

分布式服务执行的确大大降低了消息交换的数量，然而这会引起另外的一个问题：协调者必须是事前建立好的。但是由于服务驻留在不同的服务提供者的站点上，因此只能采取静态绑定的方式执行复合服务。此外，在 Self-Serv 中，服务注册中心将持有所有被发现服务的协调者，这样一来就又几乎重新集中控制服务了。目前的一种解决方法是通过构件服务之间的消息传递来定义复合服务。然而这样做的结果将导致消息的增长，从而抵消了原先减少的消息数量[13]。

还有一个值得考虑的问题：协调者如何自主决定是否执行它的构件服务。设想有这样的一个情况，一组分散的协调者希望能够保证在某一个时刻只有个一个构件服务被执行。这时候可以尝试将中央调度和对等服务执行的方法结合起来使用。

3.3 Web 复合服务的事务处理

Web 复合服务由多个构件服务组成，这些构件服务被分别调用。倘若其中有些服务被顺利执行，而有些则以失败告终，那么对于复合服务的提供者来说这就会造成数据的不一致。所以 Web 复合服务应被视作一项事务来执行。

传统的事务处理模型需要满足原子性，一致性，独立性，持久性，也就是众所周知的 ACID 特性[29~31]。这些特性可以用来验证事务的正确性。但是，传统事务模型的 ACID 特性并不能完全应用在 Web 服务的事务处理模型中[32]。

(1)原子性:一个事务流程中的多个执行步骤要么执行成功,要么就不执行。这种严格的限制对于 Web 服务并不完全适用。Web 服务合成的事务处理复杂性在于 Web 服务的提供者及与其语义等价的其他 Web 服务的存在。比如:一个 Web 服务的执行失败可以由与其语义等价的其他 Web 服务的执行来补偿。因此,在一个复合 Web 服务的执行流程中的某些步骤的失败并不能确定整个流程执行的失败。

(2)一致性:事务必须确保系统从一个稳定的状态进入另一个稳定的状态。为了确保一致性,那么整个系统的运行必须能够被本地子系统完全监控。但是由于 Web 服务严格的自治性及大量 Web 服务的存在使得上述的监控方法无法使用。因此一个研究的关注点就是在 Web 服务严格自治条件下如何监控 Web 服务的一致性。

(3)独立性:就是控制事务的并发执行以提供某种措施使得事务的并发处理能以一种序列化的方式执行。通常情况下是使用锁定机制解决这个问题。但是不同的 Web 服务可能支持不同的事务,同时可能某些 Web 服务不支持和根本不允许通过 Web 服务访问方式的锁定机制。此外不当的确保独立性会潜在地影响事务合成的性能。

(4)持久性:传统的持久性在 Web 服务合成中完全需要被支持。也就是说当事务完成其执行过程以后,即便某些步骤执行失败,它的结果也必须被持久的保存下来。这一点在 Web 服务合成中也是如此。

那么如何保持 Web 复合服务的 ACID 的性质呢?

目前的一个解决办法是在一个分布式事务的上下文中调用构件服务。当且仅当所有的构件服务顺利结束它们的工作时,系统才会提交所有产生的变更。一旦有一个构件服务失败并且无法进行补偿时,其他所有正在运行或是已经结束的构件服务将必须撤消所有的变化并退回到原来的状态。这样,复合服务的 ACID 属性才能被保证。全局事务管理器(global transaction manager, GTM)可以实现所谓的分布式事务处理。假设构件服务提供局部事务管理器(local transaction manager, LTM)的功能,GTM 就能够通过使用显式提交协议(explicit commit protocol,例如3PC)与 LTM 进行合作[13]。

然而另一方面,事务处理的需求可能会导致可用服务的附加约束,从而影响服务合成的效率。在这种情况下通过引入不严格事务处理[13]的概念可以简化这个问题,即如果有一个构件服务执行失败,系统并不停止所有的工作而是想办法寻找一个可替代的构件服务来完成请求的复合服务。

eFlow 系统支持定义事务区域。当一个事务被回退时,eFlow 会调用一个补偿程序(compensation routine)将所有的构件服务退回到调用之前的状态。AgFlow 系统则在构件服务执行失败时设法调用其他的服务。

4. 服务合成的分析评估

由于任何一项构件服务的正确性、有效性及安全性均会影响到整个复合服务,所以在复合服务投入商业运用之前对其所包含的构件服务进行分析评估是 Web 服务合成中必不可少的一个环节。在实验环境下,对构件服务的分析评估可以包括以下几个步骤[17]:

(1)在不同的负载条件下,模拟构件服务的运行;

(2)检测构件服务能否按照要求完成任务;

(3)验证构件服务是否提供对特定属性的维护;

(4)对构件服务进行性能分析。

模拟对于理解当 Web 服务被部署时如何运作有极大的辅助作用。例如,JSIM[16]可以用来模拟复合服务的集中式执行或分布式执行。模拟允许通过替换那些没有达到平均所需服务时间的服务,或是通过修改复合服务的结构,来变更服务合成的设计,并提供复合服务的反馈情况。

测试构件服务有效性的工作可以借助一些特定的模型来完成。例如在斯坦福大学的研究中,研究者采用了将 DAML-S 描述的复合服务以 Petri 网的形式表现出来的方法,利用对 Petri 网的分析技术为服务合成进行分析评估[17,19]。

性能分析有助于评估复合服务中每一项活动的代价。然而就这一方面而言,目前工业界和学术界均没有明确的评测方法和衡量标准。但是,事实上对于服务合成者来说,复合服务总的执行时间应该是衡量服务效率的关键。如果假设一个给定 Web 服务 ws 的总调用时间为 T,则 T 大致可以表示为:$T(ws)=S(ws)+M(ws)+W(ws)$。其中,S 为服务时间,表示服务执行其任务所花费的时间;M 表示消息延迟的时间;W 为等待时间,表示由于系统负载而造成的服务调用延迟的时间[12]。

负载测试是性能测试的另一个方面。其基本思想是:当客户调用不断增加时,服务的负载也会随之上升,在到达一个特定的负载点后服务的性能开始有所下降。这个标志点就是 Web 服务有效工作的负载范围(load range)[12]。这项测试工作可以用于为复合服务选择性能更好的构件服务(在提供相同功能的条件下)。

负载性能测试方法可以帮助合成者处理如下问题:

(1) 通过估计每个构件服务调用的等待时间来确定哪些服务是超载的;

(2) 通过估计每个构件服务调用的消息延迟时间来确定消息通讯的总开销;

(3) 估计每个服务能够有效工作的负载状况。

(4) 通过利用这些测试结果,对复合服务调用进行监控。

这里提及的性能分析技术能够提供复合服务质量(QoS)的反馈信息。但它们也有自身的缺点:只有当所涉及的 Web 服务在控制范围内的时候,这些技术才能够产生较好的估计。例如可能由于服务器其他程序的过度负载而造成对服务时间的估计不准确;或是当调用单个服务需要高开销的时候,这些性能分析方法可能会不适用。

结束语 目前 Web 服务合成技术在国内仍处于起步阶段。由于它需要其他相关技术的支持,如服务动态发现、复合服务执行及事务处理等技术的协作,所以如果要使 Web 服务合成发挥出其所拥有的潜能,仍然有许多的研究工作需要开展。

参 考 文 献

1 Ankolekar A, et al. DAML-S: Web Service Description for the Semantic Web. International Semantic Web Conference, Sardinia, Italy, 2002. 348~363

2 Benatallah B, Dumas M. The Self-Serv Environment for Web Services Composition. IEEE Computer Society, 2003

3 Benatallah B, Dumas M. Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. In: Proc. of the 18th Intl. Conf. Data Engineering (ICDE. 02). IEEE Computer Society, 2002

4 Boualem Benatallah M, Fauvet M-C, Rabbi FA. Towards Patterns of Web Services Composition. University of New South Wales, Sydney. 2001

5 BPEL4WS: Business Process Execution Language for Web Services version 1.1 May 2003. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/ 2003

6 Chakraborty D, Joshi A. Dynamic Service Composition: State-of-the-Art and Research Directions: [Technical Report TR-CS-01-

19]. UMBC, December, 2001. http://research.ebiquity.org/re/papers.html
7 Chakraborty D. Service Composition in Ad-hoc Environments: [Technical Report TR-CS-01-20]. UMBC, October, 2001. http://research.ebiquity.org/re/papers.html
8 Gunther C W. Implementing Interorganizations Workflows via Web Services Orchestration. Hasso Plattner Institute for Software Systems Engineering, Germany. 2002
9 DAML Joint Committee. Daml + oil (march 2001) language. http://www.daml.org/2001/03/daml+oil-index.html 2001
10 DAML Joint Committee. Daml Services (Daml-S) May 2003. http://www.daml.org/services/ 2003
11 Fensel D, Bussler G. The Web Service Modeling Framework WSMF. Electronic Commerce Research and Application. 2002. 113~137
12 Miller J A, et al. Composition, Performance Analysis and Simulation of Web Services. http://lsdis.cs.uga.edu/lib/download/CMSA+02.pdf 2002
13 Bierhoff K. Web Service Composition Patterns. Web services for B2B Integration-report on the seminar. Hasso Plattner Institute for Software Systems Engineering, Germany. 2002
14 Leymann, F. Web Service flow language (WSFL) 1.0. http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf 2001
15 Paolucci M, Kawamura T, Payne T R, Sycara K. Importing the Semantic Web in UDDI. http://citeseer.nj.nec.com/540826.html 2002
16 Miller J, Nair R, Zhang Z, Zhao H. JSIM: A Java-based Simulation and Animation Environment. In: Proc. of the 30th Annual Simulation Symposium, Atlanta, GA, 1997. 31~42
17 Narayanan S, McIlraith S. Simulation, Verification and Automated Composition of Web Services. In: Proc. of the Eleventh Intl. World Wide Web Conf. Honolulu, HI, 2002. 77~88
18 Dussel P. Composition of e-Services in a highly dynamic environment. Hasso Plattner Insititue, Germany. 2002
19 Hamadi R, Benatallah B. A Petri Net-Based Model for Web Service Composition. In: 14th Australasian Database Conf. (ADC2003), Adelaide, Australia. Conferences in Research and Practice in Information Technology, Vol. 17, 2003
20 Semantic Web Service Architecture—Evolving Web Service Standards toward the Semantic Web. American Association for Artificial Intelligence, 2001
21 McIlraith S A, Son T C, Zeng H. Semantic Web Services. IEEE Intelligent System, 2001
22 Tidwell D. Web Services: The Web's Next Revolution. http://www-106.ibm.com/developerworks/webservices/ 2000
23 Thatte S. XLANG: Web Services for Business Process Design. http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm 2001
24 UDDI Spec Technical Committee Specification. UDDI Version 3.0, July 2002. http://www.uddi.org/pubs/uddi-v3.00-published-20020719.htm 2002
25 W3C. Soap version 1.2 Proposed Recommendation, May 2003. http://www.w3.org/TR/soap12-part0/ http://www.w3.org/TR/soap12-part1/http://www.w3.org/TR/soap12-part2/ 2003
26 Web Services Description Working Group. Web Services Description Language (WSDL) Version 1.2, March 2003. http://www.w3.org/TR/wsdl12/ 2003
27 Waldt D, Drummond R. EBXML: The Global Standard for Electronic Business. http://www.ebxml.org/presentations/global_standard.htm 2001
28 Zeng L, Benatallah, B, Nguyen P. AgFlow: Agent-based Cross-Enterprise Workflow Management System. In: Proc. 27th Intl. Conf. on Very Large Data Bases, Italy, 2001
29 Gray J, Paper I. The Transaction Concept: Virtues and Limitations. In: Proc. of the 7th Intl. conf. on Very Large Data Bases, Sep. 1981. 144~154
30 Lynch N A, Merritt M, Weihl W E, et al. Atomic Transactions. Morgan Kaufmann, ISBN 1-55860-104-X, 1993
31 Bernstein P A, Newcomer E. Principles of Transaction Processing for Systems Professionals. Morgan Kaufmann, ISBN 1-55860-415-4, 1996
32 Paulo de Figueiredo Pires. WEBTRANSACT: A framework for specifying and coordinating reliable web services compositions. http://www.cos.ufrj.br/~pires/webTransact.pdf/pires02webtransact.pdf 2002
33 Graham S. Building Web Services with Java-- Making Sense of XML, SOAP, WSDL, and UDDI. China Machine Press, 2003
34 BizTalk Server Homepage. http://www.microsoft.com/china/biztalk/default.asp

(上接第51页)

二种设计方法就是利用 Rational ROSE 提供的数据库描述语言(DDL)生成功能,首先在类的定义窗口中的 DDL 接口定义域内定制库表的列名、列的类型以及主键等方面的细节,然后利用 ROSE 来生成包含可执行 SQL 语句的后缀名为.sql 的文件,借助于它就可以建立相应的数据库表。第三种方法是通过在前面所述的 E-R 图上作一定的修改来完成数据库的设计。鉴于目前本项目开发时没有 Erwin 软件包,同时采用 ROSE 生成 DDL 的方法较难生成相应的实体-关系图、物理模型图和相关文档,而 Power_ Designor 则在逻辑模型的建立、逻辑模型到物理模型的转换以及各种设计文档的生成等方面则具有强大的功能,所以采用了第三种方法进行了数据库的具体设计。

结论 本文从面向对象的角度全面论述了基于 TMN 的 SDH 网管系统的分析与设计。UML 是一种通用的建模语言,因此本文的分析结果都用其进行描述,但 UML 不是一种方法,因此面向对象中已有的方法将继续得到应用。在系统的开发过程中,我们注意到不能盲目地抛弃一些实践证明有效的一些传统方法在面向对象开发中的应用,如数据流方法及实体-关系图,本文的成果是一种成功的尝试,对电信网管系统的面向对象的应用具有积极的参考价值。

参 考 文 献

1 Booch G, Jacobson I, Rumbaugh J. UML Notation Guide (Version1.1). Rational Corporation, Santa Clara, 1997
2 Fowler M, Scott K. UML Distilled: Applying the Stand Object Modeling Language. Reading, MA: Addison-Wesley, 1997
3 Object Management Group. OMG Unified Modeling Language Specification, v.1.3 June 1999
4 Rumbaugh, James, et al. Object-Oriented Modeling and Design. Upper Saddle River, NJ: Prentice Hall, 1991
5 Jacobson, Ivar, et al. Object-Oriented Software Engineering: A Use Case Driven Approach. Reading, MA: Addison-Wesley, 1992
6 Fowler M, Scott K. Applying the Stand Object Modeling Language. Addison-Wesley, Reading. MA, 1997
7 Filippidou D. Designing with scenarios: a critical review of current research and practice. Requirements Engineering, 1997, 3:1~22
8 Richter C. Designing Flexible Object-Oriented Systems with UML. Macmillan Technical Publishing, Indianapolis, IN, 1999
9 Jacobson I, Booch G, Rumbaugh J. The Unified Software Development Process. Addison-Wesley, Reading, MA, 1999