UDDI(统一描述发现和集成)

UDDI(统一描述发现和集成)

UDDI（Universal Description, Discovery and Integration）统一描述、发现和集成协议，是为解决 Web 服务的发布和发现问题而制订的新一代基于 Internet 的电子商务技术标准。它包含一组基于 Web 的、分布式的 Web 服务信息注册中心的实现标准，以及一组使企业能将自己提供的 Web 服务注册到该中心的实现标准。

UDDI 利用 SOAP 消息来查找和注册 Web 服务。并为应用程序提供了一系列接口来访问注册中心。

UDDI(统一描述发现和集成) 提供一种发布和查找服务描述的方法。UDDI 数据实体提供对定义业务和服务信息的支持。WSDL 中定义的服务描述信息是 UDDI 注册中心信息的补充。

UDDI 概述

UDDI（通用描述、发现和集成协议）是成功 Web 服务所必需的主要的构建模块方式之一。

UDDI 计划是一个广泛的，开放的行业计划，它使得商业实体能够 (1)彼此发现，(2)定义他们怎样在 internet 上互相作用，并在一个全球的注册体系架构中共享信息。UDDI 同时也是 Web 服务集成的一个体系框架。它包含了服务描述与发现的标准规范。

- 开放源代码 UDDI 工具
- UDDI 不是 WSDL 储存库

理解 UDDI

UDDI 项目鼓励 Web 服务相互操作和相互采用。它是一种工商界居于领先地位的企业之间的伙伴关系。UDDI 提供了一组基于标准的规范用于描述和发现服务，还提供了一组基于因特网的实现。本文将介绍 UDDI 和它在 Web 服务发展过程中所起到的促进作用。您可以了解到 UDDI 的工作原理，并发现 UDDI 规范新的即将出现的功能。

UDDI 基于现成的标准，如可扩展标记语言（Extensible Markup Language，XML）和简单对象访问协议（Simple Object Access Protocol，SOAP）。UDDI 的所有兼容实现都支持 UDDI 规范。

- 将 SOA 和 Web 服务分家（一）
- 将 SOA 和 Web 服务分家（二）
- SOA 注册和存储库之间的区别
- UDDI 帮助 NASA 创建了一个全球的“集会场所”

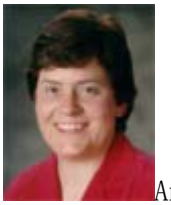
UDDI 的实现

UDDI 本质上是为解决当前在开发基于组件化的 Web 服务中所使用的技术方法无法解决的一些问题。UDDI 具有非凡的技术简单性，他为 Web 服务在技术层次上提供了重要的支持。

UDDI 之所以能够成功是由于它为广大的技术人员完成了技术实现方面的基础工作，这是成功的基础和保证。这些技术人员将使用 SOAP，UDDI 和其他渐渐出现的 Web 服务协议堆栈中的其他协议层来架构通往广泛的异构系统的协同、供应链以及 EAI 解决方案等的桥梁。

- WSDL 到 UDDI 的映射
- UDDI 客户机和 UDDI 实现
- SOA 和 Web Service 分道扬镳

开放源代码 UDDI 工具

 Anne Thomas Manes，Burton Group 的研究部总监，是 Web 服务领域知名的技术专家。她曾在 2001 年被《网络世界》评为“网络最具影响力的 50 位人物”之一；2002 年被《企业系统期刊》评为“100 名杰出 IT 领导者”之一。加入 Burton Group 之前，Manes 曾是 Bowlight 公司的创建者和 CEO，专注软件行业分析和咨询。她具有二十四年的行业经验，曾担任 Systinet 公司（从事 Web 服务基础设施）的首席技术官，革新 Sun 的 Web 服务战略，并在 Patricia Seybold Group 担任分析师。作为分布式计算技术领域的知名专家，Manes 还参与了 W3C, OASIS, JCP, UDDI 和 WS-I.的标准开发。她还是业界最权威出版务《Web 服务期刊》编辑专家组的会员之一；多次参加会议演讲，发表无数文章，并著有《Web Services: A Manager's Guide》一书，由 Addison Wesley 出版。

问：是否有好的开放源代码 UDDI 工具？

答：据我所知，三开放源代码 UDDI 实现：

- Apache jUDDI
- Ruddi
- OpenUDDI

这些都是开放源代码 UDDI 的客户端项目：

- UDDI 浏览器

- Apache Scout
- UDDI4J
- uddi4r
- UDDI: Lite
- Sun 的 JAXR 参考执行 JWSDP

没有方案是充满活力的。UDDI 实现了什么，我所指的“vanilla”，实现了 UDDI 规范。他们没有提供额外的工具，功能或价值。客户端程序库不这样做的很多到抽象的复杂事物，UDDI 数据模型。因此，答案是： “是的，没有良好的开放源代码 UDDI 工具” 。

(作者：Anne Thomas Manes 来源：TechTarget 中国)

UDDI 不是 WSDL 储存库

通常人们认为 UDDI（统一描述、发现和集成）注册表为 WSDL 文档提供储存库但是实际上 UDDI spec 并不知道 WSDL。

UDDI 为获取信息定义了一个相对简单的数据模型：

- 服务提供者(businessEntity)
- 服务(businessService)
- 服务绑定(bindingTemplate)
- 技术模型和规范(tModel)

通过使用名称/值对，UDDI 同时提供分类信息的方法 (keyedReference)。

该模型是能够扑捉与任何服务类型，旧应用，规范以及软件资产相关信息的通用信息模型。例如，你可以注册命名空间，图解，要素结构，XSLT 样式表，UML 模型，组件，甚至 COBOL CopyBook。

OASIS UDDI-spec 技术委员会发布的技术说明定义了一个直接将 WSDL 定义映射到 UDDL 数据模型的直接方法。

WSDL 端口类型和绑定以被注册为 tModel，WSDL 被注册为 businessServices，WSDL 端口被注册为 bindingTemplates。真正的 WSDL 文档并没有储存在注册表中。bindingTemplate 指向服务访问点，你能用追加的方法找到服务 WSDL 吗？或者端点 URL 的 WSDL。代表不同 WSDL artifacts 的 tModels 将指向其相关的 WSDL 文档。不过要记住，如

果真是在实施 SOA，那么 WSDL 端点类型和绑定就是可重用的组件并不一定代表单一的一个服务。

UDDI-spec TC 同时也为 BPEL4WS 和 ebXML 定义了一个标准的映射。TC 在其它的标准映射基础上进行工作。

（作者：Anne Thomas Manes 译者：杨君 来源：TechTarget 中国）

将 SOA 和 Web 服务分家（一）

人们将面向服务架构误认是与 Web 服务是相同的概念。而且这种误解非常普遍，深深了影响了设计师，开发商，咨询师以及供应商的工作。由于 ZapThink 每天要解决许多问题，并且仓促的调查不足以清楚的解释人们对于二者之间关系的困惑。因此对 SOA 的定义是不充分的：“SOA 是组织 IT 资源以便更好满足不断变化业务需求的方法。”“对于软件功能和数据来讲，Web 服务是建立在标准之上受限接口。”但是，毕竟这些有关 SOA 片面的定义，对 SOA 的不了解，一直困扰这人们。那么，为什么我们会一直对此困惑不解呢？我们怎样才能够解决这个问题呢？首先，让我们看一看这两个相互联系但又相互分离的两个概念，以帮助我们理解二者的不同之处。

首先将 SOA 和 Web 服务结合在一起

有一个迹象能够表明 SOA 和 Web 服务是两个不同事物，即早在 Web 服务产生很久以前，SOA 就已经存在了。同公用对象请求代理[调度]程序体系结构和微软的分布部件对象模(DCOM)一样，从 90 年代开始，分布式计算方法就是能够以订立合同的方式抽取软件功能的结构方法，这种订立合同的方式为我们提供了一定量的松耦合，并且同那些利用紧耦合接口的方法相比，更具灵活性——换句话说，它们是以服务为导向的。CORBA 和 DCOM 都在市场上取得了一定的成功，DCOM 是单一供应商的架构，CORBA 表面上却是供应商中立的，但是由于事实证明多个供应商在实施 CORBA 互不相容，因此在实践中 CORBA 不可能有特定的供应商。

但是事情的进展并没有引起人们更多的兴趣，直到 90 年代末，几个供应商才意识到以下重要两点：首先，只有 SOA 的实施独立时，SOA 才能提供灵活性。其次，较为新颖的可扩展性标识语言(XML)可以制定一个理想的协议，尽管其最初目标是成为文档标识语言。供应商这些独到的见解帮助他们最终将工作重心放在了核心规范上。该核心规范为 Web 服务提供了依据：Web 服务描述语言(WSDL)，统一描述、发现和集成(UDDI)以及简单

对象访问协议(SOAP），如同访问对象不再是其存在原因一样，它们现在不再是简单的缩略词。

人们早期对于这三个标准启用的“寻找—绑定—发布”三角形的研究关注业务到业务(B2B)应用，人们最初对于全球“绿页”目录的独到见解准许自动发现和通过网络绑定到业务 Web 服务。问题是，没有人想用这样的方式完成业务。如同从黄页目录里寻找水暖工一样，谁愿意自动化流程并在在系统之间添加任意互操作呢？结果人们早期对于 SOA 的业务到业务(B2B）观点完全瓦解了，其只不过是 dot.com 繁荣发展接近尾声时，″Web 1.0″B2B eMarketplace 败笔的一部分。

Web 服务占据了中心位置

Dot.com 的结束以及后来 IT 行业的逐渐下滑给许多标准设置了障碍。2002-2003 年度标志着 Web 服务的黄金时代。供应商意识到在困难时期唯一有希望产生业务的就是节约成本这个提议，对于 Web 服务来说就是要：降低集成成本。从专有接口向基于标准的接口转移，这样就会节省许多资金！但是，现今的标准远远不能满足 Web 服务快速发展的需求，至少对于那些寻找新商机的供应商来说，这些业务实例足够坚固了，因此 Web 服务市场也就诞生了。

根据我们早期对于 Web 服务技术和趋势，XML&Web 服务安全，服务定向管理以及 Web 服务测试的调查，ZapThink 公司自然要借助 Web 服务的威力。然而早在 2002 年 2 月 XML 和 Web 服务相关的书籍出版以前，我们在谈论架构之时就建议供应商要涉及 SOA，因为市场还没有能力应对以 SOA 代表的如此复杂和业务为中心的议题。相反人们的目光集中到了 Web 服务架构这个基于标准的集成化方法上。

然而，我们意识到我们在服务定向集成报告中写到的 2002 年的基本事实和今天的事实一样具有可预测性：单靠 Web 服务就可以降低集成成本，只有使用 SOA，机构才能降低业务交易的长期成本。换句话说，Web 服务为你参加舞会买单，但是你还得学会跳舞。

(作者：Jason Bloomberg 译者：杨君 来源：TechTarget 中国)

将 SOA 和 Web 服务分家（二）

Web 服务黄金时代的结束

鉴于人们对 SOA 的讨论越来越激烈，Web 服务也进入了快速发展时期。单纯能够为 WSDL, UDDI,和 SOAP 提供支持不足以确保互操作性，更无法令任意系统到系统互操作标准化。这种局限性导致：人们努力建立 Web 服务互操作性组织(WS-I），该组织提高了标准的互操作性收益，以及建立其它用 WS-*标注的标准(它应该读作"WS 星,"，这里的星是旧 Unix 术语，表示“一切和所有”）人们的这些努力需要花费许多时间，并且由于不同的标准体包含了大部分供应商，整个工作就变得更为复杂了，并且交付给那些试图降低集成成本机构的互操作性少之又少，结果 Web 服务无法发挥其应有的潜力，目前还处于 SOA 的边缘地带。

事实上，当 IT 产品供应商看到 SOA 这口井中的金矿时，Web 服务这轮马车的轮子就要掉了，这些供应商开始快速使用 Web 服务接口，并称其为服务定向方法，事实上，应用或者数据的 Web 服务接口，甚至是置于专有通信中间件之上的 Web 服务配置器都不能使用 SOA 建造。

同时，在 SOA 工具和最佳实践以及服务定向流程报告中，ZapThink 将 SOA 以业务流程为中心的远景设计为企业架构。早在 2004 年，我们就开始建议供应商要关注在通信中对 SOA 关注要多于对 Web 服务的关注。我们为企业规划的远景蓝图远比直接采用 Web 服务更具挑战性，因为 SOA 要重新思考业务如何通过多种方式利用 IT 资源。Web 服务仍旧只是其中的一部分。现在我们知道 Web 服务对于 SOA 来说并不是必不可少的，也不是必需品。

ZapThink 采取的措施

因此在 2007 年，Web 服务和 SOA 分了家。在 SOA 环境下，我们所说的服务和特定的接口标准相比更为抽象，开发商利用这些接口标准支持机构的互操作性要求。许多这样的服

务都是 Web 服务，但还有很多服务不是 Web 服务。此外，Web 服务的大多数应用是在 SOA 环境外部发生的。事实上，许多这样的服务都是从业务到业务 B2B 的，并最终返回到原始的 Web 服务远景（即使没有绿页）。然而大多数 B2BWeb 服务只是基于标准的应用程序接口，缺乏一个架构所能够提供的松耦合，位置独立，以及业务灵活性。另外，还有许多机构试图实施 SOA,但实际上却是在实施 Web 服务，最终只得到了冗长的，无法兼容和难以管理的服务，这些服务实际上和架构毫不相关。

当我们回过头来 Web 服务和 SOA 以往的历史就会发现二者之间的结合充满了艰难与曲折，因为 Web 服务在 SOA 的引入过程当中扮演了十分重要的角色，尽管 SOA 的发展远远超出了 Web 服务的范围，我们的工作还是没有完成，因为 Web 服务 vs. SOA 之间许多令人极为困惑的问题有待解决，也许人们最大的挑战还是要确保 SOA 确实和业务流程有关，而不是和集成化有关。只要供应商还错误地坚持软件是 SOA 成功关键，我们的工作就远没有完成。

（作者：Jason Bloomberg 译者：杨君 来源：TechTarget 中国）

SOA 注册和存储库之间的区别

讨论关于 SOA 注册库(Registry)的声音还没有退去，人们又更多地意识到 SOA 中另一个关键的组件存储库(Repository)。注册库和存储库之间究竟有什么区别呢?在 ZapThink 网站最近举办的一次关于注册与存储库的 Webcast 中，大多数人认为注册库存放的是对事物的引用，而存储库中则存放事物本身。

很多人认为注册库保存元数据信息而存储库则用来存放数据。这个区别似乎不是很明显，但是事实上情况要复杂得多。元数据就是关于数据的数据，对吗?如果是这样的话，那么文档是什么呢?比如，现在有一个微软的 Word 格式文档，那么我们可以把它看作是数据本身对吗?先别急，因为文档是描述服务的，所以我们可以把它当作是元数据。然而，微软又承诺，Microsoft Office 系列文档的基本格式是 XML，那么 XML 是否应该被看作是 Word 文档的元数据呢?其实，要区别数据和元数据的关键是它们之间的关系，很有可能一个人的数据是另一个人的元数据。而在现实生活中，某种数据极有可能刚开始扮演数据的角色，然后变更为元数据的角色，最后又回到数据的角色，不停的变换。

可以说，这就是为什么数据和元数据界限的模糊性是一个成熟的市场越来越趋向于提供集成的注册库和存储库解决方案的原因。

有很多的区别可以帮助人们找出数据和元数据的界限。一个很有效的方法是找出设计时间和运行时间的不同。注册库和存储库两者都包含了设计时间和运行时间两个特性。设计时间元数据绝大多数情况下专注于描述和发现，而运行时间元数据则专注于分发合约和策略信息。设计时间通常反映出人工时间，例如代码的编写，所以通常设计时间存储库使用一些类似 CVS(协同版本系统)的标准。而运行时间的存储库则通常存储消息并提供查询、审核、日志或一些归档的能力。

	设计时间	运行时间
注册库	• 探索 • 描述	• 合约 • 策略 • 版本
存储库	代码版本文档	• 可查询式信息存储 • 日志 • 审核

很明显，除此之外，在这方面的产品还会附带很多其他的功能，例如治理、联合、订阅与通告、安全性、验证、报告以及管理等。Infravio 自从其第一个产品(被称作 X-Registry)开始就提供了对一个集成的注册/存储库模型的支持。X-Registry 使用了 JAXR 进行开发，JAXR 是用于在标准注册顶层开发应用程序的 Java 编程 API，它包含了统一描述、发现和集成(UDDI)、ebXML RIM 以及注册信息模型。

Sun 微系统公司年初也曾发布了一款轻量级的注册库(Registry)产品，该举动证明这种解决方案的认可程度。这款产品是基于 FreebXML 代码基础的，并集成了注册库和存储库的功能，同时支持 JAXR。

此外，Systinet 公司最初的产品只基于 UDDI 注册，现在也改变了策略。该公司代号为 Blizzard 的产品的下个版本将集成基于 XQuery 接口的运行时间存储库。

如此繁杂的产品和技术也许会给我们带来一个疑问，和存储库相关的标准究竟是什么呢?

要回答这个问题，我们需要关注一下 SOA 生命周期以及相关的功能。在设计时间里，类似 CVS 这样的标准用于存储人工数据，例如代码，这就是一种存储库。而当你进入 SOA 运行时间，你可能需要存储信息数据。比如你希望获得信息查询的能力，那么你就需要使用 XQuery 作为逻辑接口。XQuery 允许用户对分布式 XML 数据进行查询，整个过程看起来

就像是对一个数据库进行操作一样。除此之外，如果你期望获得更丰富的信息模型和类似可审核信息日志这样的东西，那么你可能会需要 ebXML 注册技术。

最后，人们越来越多地把注册和存储器作为 SOA“平台”的一个整合部分来看待了。而当你需要在注册技术和存储器之间做出选择时，可以根据你所关注的内容是设计时间还是运行时间来考虑。只要你理解了在 SOA 服务生命周期中需要什么样的功能，你就能够做出一个更好的选择。

（作者：Miko Matsumura 来源：TechTarget 中国）

UDDI 帮助 NASA 创建了一个全球的“集会场所”

商业总是希望可以构建更加灵活的架构，这样的架构可以使他们能够发现和以新的方法来重复使用它们的 IT 资产，这些可以从 NASA 的报告中看到相关的记录。

在类似 SOA ，SOAP 和 UDDI 这样的缩写字母成为本国语言的一部分之前，地球科学委员会，在 NASA 的赞助下，开始构建一个为地球观测系统交换中心(ECHO)的 SOA。该中心允许科学家访问，搜索并共享地球科学的海量数据。现在 NASA 已经铺开了一个 ECHO 的扩展服务，该服务基于一个来自 Burlington, Mass.的 Systinet 有限责任公司 的 UDDI 注册系统。该 UDDI 系统允许第三方发布和访问数据。

ECHO，开始于 1998 年，从 get-go 开始构建而得到的，利用 XML 和 Web 服务技术。ECHO 系统类似一个在数据伙伴和客户伙伴之间的中间件而运行。客户端的伙伴开发可以访问信息的软件。

最初地，ECHO 是地球科学数据的注册场所，在那儿数据的提供者可以发布和注册他们所具有的数据。“我们总是试图去为这些组织提供相同的能力，并且人们愿意来提供在地球科学领域的服务。”ECHO 的主要系统工程师 Keith Wichmann 在政府任务的承担人地球科学技术有限公司(GST)说。地球科学技术有限公司(GST)和次承担人，弗吉尼亚的 Vienna 的 Blueprint 技术有限公司一起实现了这个系统。“我们需要一个机制来发布这些服务。UDDI 看起来似乎是一个最好的选择。同时他需要使用我们的数据注册来完成被集成。”

新的服务注册的扩展“允许第三方发布他们的 Web 服务能力和通过在他们可以做到之上的数据来联结这些服务，并会在 ECHO 的数据注册中表现出来。”地球科学数据和信息系统工程的主要的信息管理系统工程师，Robin Pfister 说。

在 ECHO 背后的概念是一个“讨论场所”， GST 的程序开发经理 Mark Nestler 说。“已经有了很多的数据了。给我们的挑战就是发现它并且可以找到新的方法来使用它。”

根据 Pfister 说的，ECHO 之后的驱动器，和为了他的服务的定位，就是科学家希望可以有更多的对资源的控制。

“ECHO 正在满足一个需求，该需求是被地球科学委员会需要为了完成他们的研究和应用而访问数据的方式的改变驱动的。”她说，根本的，这个改变就是那些科学家想要对他们贡献的资源具有更多的控制，并且他们想要更多地直接访问其他科学家贡献的资源。如果我们把这些资源——数据，数据变换服务等等——作为服务来看待，这个时候面向服务架构自然而然的适合这个协会的需求。

灵活性也是很重要的，Blueprint 的总裁和 CEO Jeanne O’Kelley 说:“当你和来自世界范围的用户一起工作的时候，这就意味着每个人都有他们自己的方式来完成这个事情。在设计中给予他们灵活性真的是非常重要的。”

ECHO 被以一个分层的架构开发出来的，这个分层架构使得它可以适应新的标准和改变，Wichmann 说。他说整个小组也同意“和系统交互的最好的办法是所有的参与者向系统发送 XML 信息同时系统将会响应， 并且我们构建了一个这样的机制来完成这个。然后 SOAP 随之而来了，我们扩展它为 SOAP。”

ECHO 环境由一个 Oracle 集团的 Oracle 9i 的后台数据库组成，并且 Oracle 9i 正计划升级为 Oracle 10g。VEA 系统有限公司的 WebLogic 应用服务器处理了大多数的核心业务逻辑。Apache Axis 和 Tomcat 也被使用了。Systinet UDDI 服务器被用来提供这些服务的通过工业标准的 UDDI 协议完成的注册。开发者使用 Eclipse 和一些列的其他开发工具。

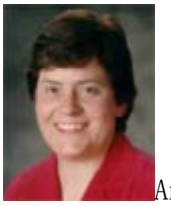
Wichmann 说开发团队现在正在进行 ECHO 系统的升级，以在适应 Web 服务的互操作(WS-I)的基本协议架构。系统的 version 8(现在的版本是 6) 中，他说，“ECHO 自己就将被作为一个 Web 服务获得。”

“当 ECHO 已经在自 SOAP 的标准之后对系统具有一个 SOAP 的认识，ECHO 使用的标准需要杯补充以满足 Web 服务基本架构的热潮，从而我们的终端用户——开发者——可以对构建再 ECHO 的能力之上的工业工具起到杠杆作用。”Pfister 说“这些能力也会在服务注册处注册的。”

根据 Systinet 的市场副总裁 David Butler 说，“我们现在正在看到 SOA 和 Web 服务标准已经正在被用于一些相当大的工程中。这是一个相当大范围的发现工程，并且多余一个的 SOAP/WSDL/UDDI 的概念堆栈的例子正在变成支持和管理这些服务的推荐方式。 ”

(作者：Colleen Frye 来源：TechTarget 中国)

WSDL 到 UDDI 的映射



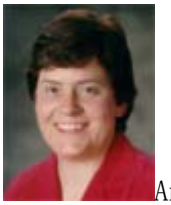
Anne Thomas Manes，Burton Group 的研究部总监，是 Web 服务领域知名的技术专家。她曾在 2001 年被《网络世界》评为“网络最具影响力的 50 位人物”之一；2002 年被《企业系统期刊》评为“100 名杰出 IT 领导者”之一。加入 Burton Group 之前，Manes 曾是 Bowlight 公司的创建者和 CEO，专注软件行业分析和咨询。她具有二十四年的行业经验，曾担任 Systinet 公司（从事 Web 服务基础设施）的首席技术官，革新 Sun 的 Web 服务战略，并在 Patricia Seybold Group 担任分析师。作为分布式计算技术领域的知名专家，Manes 还参与了 W3C, OASIS, JCP, UDDI 和 WS-I.的标准开发。她还是业界最权威出版务《Web 服务期刊》编辑专家组的会员之一；多次参加会议演讲，发表无数文章，并著有《Web Services: A Manager’s Guide》一书，由 Addison Wesley 出版。

问：是否有像“Systinet Registry”那样的开源代码，用于 WSDL 到 UDDI 的映射，可以自动将 WSDL 文档自动发布成为 UDDI（tModel）？

答：Eclipse Web 工具项目包含了在线注册向导（作为 Web 服务资源管理工具的一部分）。在 SourceForge 上的 WSDL4REG 还提供了映射向导，虽然这个项目看上起已经终止。

(作者：Anne Thomas Manes 来源：TechTarget 中国)

UDDI 客户机和 UDDI 实现

 Anne Thomas Manes，Burton Group 的研究部总监，是 Web 服务领域知名的技术专家。她曾在 2001 年被《网络世界》评为“网络最具影响力的 50 位人物”之一；2002 年被《企业系统期刊》评为“100 名杰出 IT 领导者”之一。加入 Burton Group 之前，Manes 曾是 Bowlight 公司的创建者和 CEO，专注软件行业分析和咨询。她具有二十四年的行业经验，曾担任 Systinet 公司（从事 Web 服务基础设施）的首席技术官，革新 Sun 的 Web 服务战略，并在 Patricia Seybold Group 担任分析师。作为分布式计算技术领域的知名专家，Manes 还参与了 W3C, OASIS, JCP, UDDI 和 WS-I.的标准开发。她还是业界最权威出版务《Web 服务期刊》编辑专家组的会员之一；多次参加会议演讲，发表无数文章，并著有《Web Services: A Manager's Guide》一书，由 Addison Wesley 出版。

问：开放源代码的实现和开放源代码的客户机之间的区别是什么？

答：这是一个复杂的问题，关于开放源代码 UDDI 选项的先决问题。UDDI 的实施是一个 UDDI 兼容的注册服务，即一个 Web 服务。通过 UDDI API 应用程序接口，实现了 UDDI 的规格。UDDI 的客户端是一个客户端库或工具，与 UDDI 兼容的注册服务相结合。

(作者：Anne Thomas Manes 来源：TechTarget 中国)

SOA 和 Web Service 分道扬镳

可能在市场上围绕着面向对象的架构(service-oriented architecture，SOA)误解最深的是 SOA 和 Web Service 是同一个概念。这个误解传播的很广，影响了架构师和开发人员、咨询师和厂商等。但是尽管 ZapThink 不断的在日常的事务上澄清很多问题，这种误解还是存在于一些匆匆检查中无法轻易辨别的细微之处。结果，仅仅站起来喊一下“SOA 是组织 IT 资源更好的满足业务不断变化的需求的一种方法!”“Web Service 是基于标准的、协议化的软件功能和数据的接口!”是远远不够的。毕竟，如果仅仅是关于不同术语的各自定义的问题，那么这种误解早就消失了。所以，为什么这么基础的误解至今还困扰着我们?我们该做些什么来解决这个问题并最终取得进步呢?对这两个相关、但是各自不同的概念的历史进行简要回顾将有助于澄清这个差异。

最初 SOA 和 Web Service 是捆绑在一起的

表明 SOA 和 Web Service 是不同的概念的第一个证据是 SOA 在 Web Service 出现之前早已存在。早在 1990 年，像公用对象请求代管者体系结构(Common Object Request Broker Architecture，简写为 CORBA)和微软的分布式组件模型(Distributed Component Object Model，缩写为 DCOM)的分布式计算方法都是以一种协议的方法抽象软件功能的架构方法，能够提供一定程度的松耦合性，提供比使用其他方法的紧耦合性接口的架构更大的灵活性——换句话说，它们是面向服务的。虽然 CORBA 和 DCOM 都在市场上获得了一定的成功，但是 DCOM 明显就是一家厂商的架构，而 COBRA 虽然表面上是厂商独立的，但是在实施中还是和厂商有关，因为 CORBA 不同的厂商的实现被证明是互相不兼容的。

这个故事还算不上有趣，直到 1990 年代的后期，当时一些厂商达成了两点基本的共识：第一，SOA 的方法不会提供真正的灵活性，除非它是与实现独立的，第二，相对较新的可扩展标记语言(eXtensible Markup Language，XML)会成为理想的消息协议，即使它

的最初的目的是作为文档标记语言。这些观点导致了多家厂商的标准的努力并最终确立了为 Web Service 提供基础的规范的核心：Web Service 描述语言(Web Services Description Language，WSDL)，统一描述、发现和集成协议(Universal Description, Discovery, and Integration，UDDI)和简单对象访问协议(Simple Object Access Protocol，SOAP)(现在已经不是这个的缩写了，因为访问对象已经没有意义)。

SOA 由三个标准支持的“发现-绑定-发布”三角的早期工作主要集中在商业对商业(business-to-business，简称 B2B)的应用上，同时还有全球的“green pages”目录，它允许对整个 Internet 上商业 Web Service 的自动发现和绑定。问题是，没有人愿意按照这样的方式来做业务。实际上从黄页上选择一个水管工都十分冒险，所以还有谁会自动化流程，在系统之间增加任意的交互?结果，作为.com 繁荣期末尾的“Web 1.0”B2B 电子商场趋势的失败的一小部分，这个早期的 SOA 的 B2B 计划失败了。

Web Service 唱主角

虽然疯狂的.com 的终结和随之而来的 IT 的衰退打击了很多标准的制订工作，但是我们将 2002-2003 期间称为 Web Service 的黄金时代。厂商们意识到在艰难时期唯一有希望产生业务的故事就是节约成本的方案，并且 Web Service 有一个伟大的特点：降低集成的成本。从私有的接口转到基于标准的接口，想想你可以节约多少钱!虽然那些日子里标准还远不是成熟，但是至少商业案例对于投资者而言已经足够美妙了，这些投资者都在泡沫破灭的回忆中战战兢兢并且寻找新的机会。就这样，Web Service 的市场诞生了。

当然，凭借着早期对于 Web Service 技术和趋势、XML&Web Service 安全、面向服务的管理和测试 Web Service 等报道，ZapThink 又一次领导了 Web Service 的潮流。并且，早在 2002 年二月当我们在《XML and Web Services Unleashed》书中讨论架构的时候，我们建议那时的厂商不要谈到 SOA，因为市场还没有准备好 SOA 所代表的更加复杂、以商业为中心的逻辑。相反，这些报道以 Web Service 架构为中心，该架构是基于标准的集成方法。

并且，回顾 2002 年的时候，我们意识到在那一年 6 月的面向服务的集成报告中有一个在今天来看都是预言性的一个基本的真理：虽然 Web Service 单独就可以降低集成的成本，只有转移到 SOA 方能降低组织机构的业务变化的长期成本。换句话说，Web Service 让你获得了去舞会的入场卷，但是你还要学会跳舞。

Web Service 黄金时代的结束

随着 SOA 的讨论开始增加，Web Service 则进入了挑战的青春期。支持 WSDL、UDDI 和 SOAP 并不能保证互操作，并且也不足以标准化随机的系统对系统之间的交互的复杂性。这些限制导致了两方面的努力：Web Service 互操作组织(Web Services Interoperability Organization，缩写 WS-I)，它致力于提供标准的互操作，和其他各种后续的标准，其中很多都被用术语 WS-*来概括(发“WS star”的音，*在老的 Unix 中表示“一切”)。这些努力，当然，会花费时间，并且随着各种标准的正文都是由各种厂商用自己的计划来填充的，整个情况就开始变的复杂的、政治的泥潭，而这些并没有给试图降低集成成本的组织结构提供些许的真正的互操作。结果，Web Service 并没有达到原有的期望，而现在也越来越成为 SOA 故事的边缘部分了。

事实上，当许多 IT 产品厂商看到 SOA 井中的黄金之后，Web Service 的花车慢慢的没落了。这些厂商开始拍着他们产品上的 Web Service 接口，叫嚷着这是面向服务的，这种方法无异于给小猪画上口红。事实上，对应用或者数据库的 Web Service 接口，或者是在私有消息中间件上的 Web Service 适配器，都不算 SOA 的。

同时，在我们的 SOA 工具和最佳实践以及面向服务的流程报道中，ZapThink 指出以商业流程为中心的 SOA 是企业架构，并且在 2004 年，我们开始建议厂商的广告要集中在 SOA 上而不是 Web Service。我们为如今的企业所绘制的图景要比直接采取 Web Service 更加具有挑战性，因为 SOA 包括对业务如何从很多不同的方面来利用 IT 的重新思考。Web

Service 仍然是故事中的一部分，但是如今很清楚的是 Web Service 不是 SOA 的核心，并且更进一步，SOA 并不需要 Web Service。

ZapThink 的行动

因此，2007 年的故事是将 Web Service 从 SOA 中分离。我们在 SOA 环境中所谈到的服务要比开发人员用来支持组织机构的互操作要求的某个接口标准具有更高的抽象级别。很多这样的服务是 Web Service，但是很多并不是。此外，很多 Web Service 的应用发生在 SOA 的环境之外。事实上，很多这样的应用是 B2B，重新回到了 Web Service 最开始的景象(虽然很感激，没有了 green pages)。并且，大部分这样的 B2B Web Service 仅仅是基于标准的应用编程接口(API)，缺乏提供松耦合、位置无关和业务敏捷性的架构。此外，有很多的组织机构想尝试实施 SOA，但是仅仅是实施了 Web Service，造成了和架构毫无关系的冗余的、不兼容的、常常是无法管理的服务。

回顾 SOA 和 Web Service 的历史，展示了一段非常有趣的迂回曲折的婚姻，由于 Web Service 在将 SOA 引到台前中发挥了关键的作用，即使 SOA 现在已经超出了 Web Service 可以提供的范畴。并且，我们的任务还没有完成，因为围绕着 Web Service 和 SOA 还有太多的误解需要澄清。可能最大的挑战就是建立一种观点，SOA 是关于业务流程的，而不是集成的。只要厂商还在利用他们的软件对 SOA 的成功非常关键这样的错误观念来销售集成软件，这种挑战就存在。

(作者：Jason Bloomberg 来源：TechTarget 中国)