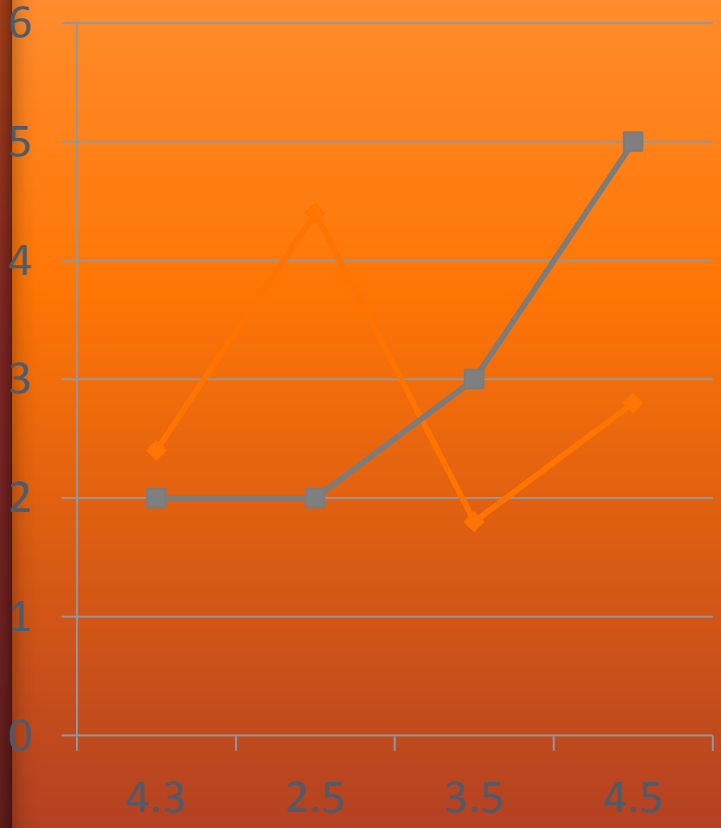


关于SOA服务管控和治理解决方案的讲义

平台研发项目组 雷强

leiqiangcn@gmail.com

治理和管理需求

面向服务的体系结构 (SOA) 是一项引人注目的技术，用于开发与业务模型保持最佳一致性的软件应用程序。不过，SOA 会提高业务和信息技术（Information Technology，IT）以及 IT 部门和各个团队间所需的合作和协调级别。这个合作和协调是通过 SOA 治理提供的，涵盖了用于指定和管理如何支持服务和 SOA 应用程序的各个任务和流程。

SOA 治理的一些重要方面

- 服务定义
- 服务开发生命周期
- 服务版本治理
- 服务迁移
- 服务注册中心
- 服务消息模型
- 服务监视
- 服务所有权
- 服务测试
- 服务安全

治理

治理意味着建立和执行工作组为了一起工作而一致同意的工作指南。具体来说，治理包括以下方面：

- 建立授权的责任链。
- 度量评估的有效性。
- 指导组织建立满足其目标的策略。
- 控制机制以确保遵从性。
- 进行沟通以使所有相关方都获得通知

治理确定谁负责制定决策，需要制定什么决策，以及使决策制定保持一致的决策。

治理不同于管理。治理规划需要制定什么决策，而管理是制定和实施决策的过程。治理重在建立决策，而管理重在贯彻执行决策。

IT治理和SOA治理

IT 治理是指针对 IT 的治理；即：针对 IT 组织及其人员、流程和信息应用治理，以提供指导，使这些资产支持业务需求。SOA 治理是 IT 治理的一种特殊化，其将关键 IT 治理决策置于服务组件、服务和业务流程的生命周期上下文中。SOA 治理对生命周期进行有效管理，生命周期是其关键目标。

IT 治理比SOA 治理更广泛。IT 治理涉及 IT 的所有方面，包括影响 SOA 的问题（如数据模型和安全性）以及 SOA 之外的问题（如数据存储和桌面支持）。SOA 治理重点关注服务生命周期的一些方面，例如：计划、发布、发现、版本治理、管理和安全性。

治理在 SOA 中比在普通 IT 中更为重要。在 SOA 中，服务使用者和服务提供者运行于不同的进程中，由不同的部门开发和管理，为了成功地一起工作，需要进行大量的协调工作。为了 SOA 能成功，多个应用程序需要能共享相同的服务，这意味着它们需要进行协调，以便共享和重用这些服务。这些就是治理问题，比采用独立应用程序时（甚至包括使用可重用代码和组件时）要复杂得多。

随着各个公司开始使用 SOA 来更好地保持 IT 与业务间的一致，可以使用 SOA 治理来非常理想地改进总体 IT 治理。如果公司要实现 SOA 的各种好处，采用 SOA 治理是非常关键的。对于 SOA 的成功，SOA 业务和技术治理不是可选的，而是必须使用的手段。

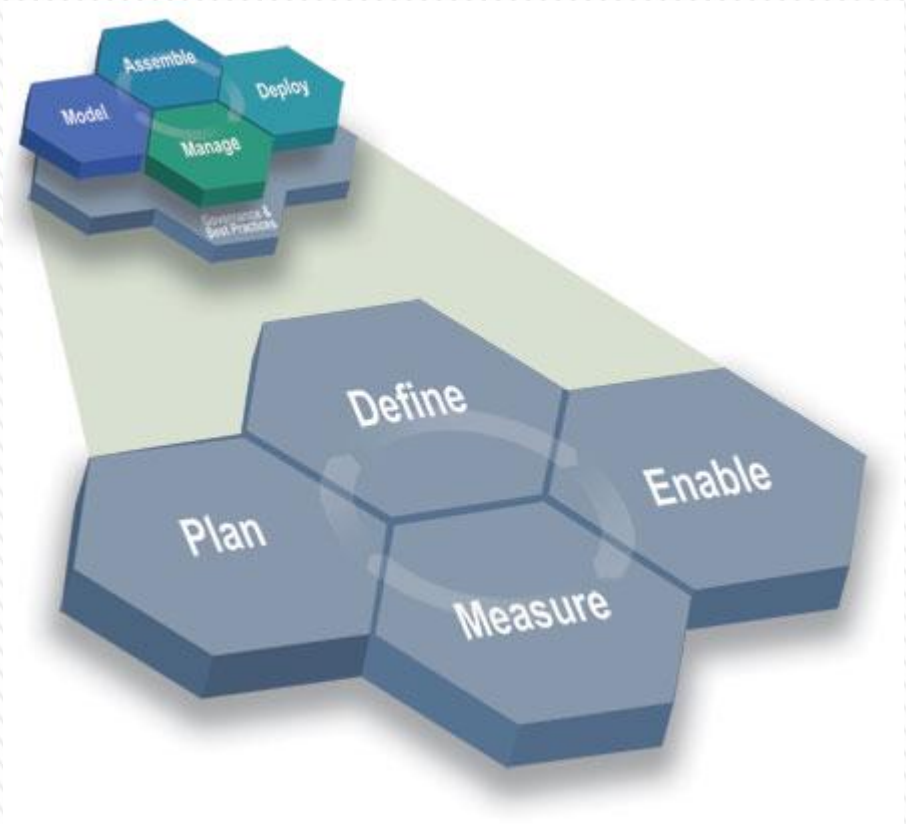
SOA 治理实践

- 在实践中，SOA 治理指导可重用资产的开发，确立如何设计和开发服务，以及这些服务如何随时间增长进行更改。它将在服务提供者和服务使用者之间建立一个协议，告知使用者可以希望得到什么功能，告知提供者应该提供什么功能。
- SOA 治理并不设计服务，而是指导将如何设计服务。
- SOA 治理以现有 IT 治理技术和实践为基础。
- 治理更多的是政策问题，而不是技术或业务问题。技术的重点是匹配接口和调用协议。业务的重点是为客户服务的功能。技术和业务都关注的是需求。虽然治理也涉及这些方面，但它更多的是要确保所有部分一起工作，独立的工作彼此并不会冲突。治理并不会确定决策的结果是什么，而是考虑必须进行哪些决策以及谁进行这些决策。
- 使用者和提供者双方彼此就如何一起工作达成一致。这些认识上的一致大部分都能在服务水平协议（Service-Level Agreement，SLA）进行捕获，形成服务提供者愿意提供且服务使用者愿意接受的可度量目标。这个协议就像各方之间的契约，而且可以成为事实上的具有法律效力的合同。SLA 至少要清楚说明提供者必须进行怎样的工作，以及使用者可以期望得到什么服务。
- SOA 治理是由精英团队（Center of Excellence，COE）（一组知识丰富的 SOA 技术人员）制订的，他们负责建立策略并监督其执行，从而帮助确保企业的 SOA 成功。COE 将建立各种策略，以用于标识和开发服务、建立 SLA、管理注册中心以及进行其他提供有效治理的工作。COE 成员随后将这些策略付诸实施，指导和帮助团队开发服务和组合应用程序。

治理生命周期和方法

服务开发遵循生命周期原则，IBM 将这一生命周期称为 SOA 生命周期。SOA 治理也同样遵循生命周期原则，即 SOA 治理生命周期。这两个生命周期相互配合、同时运行，并且一起被使用来产生 SOA 组合应用程序及其服务。 SOA 治理生命周期产生一个治理模型来管理 SOA 生命周期。

SOA 治理生命周期如图 下图所示:



SOA Governance and Management Method (SGMM)

IBM 的 SOA Governance and Management Method (SGMM) 是一个完整的流程，用于执行 SOA 治理生命周期，以将治理应用于 SOA 生命周期。SGMM 包括四个阶段：

- 计划——确定 SOA治理的重点
- 定义——定义 SOA 治理模型
- 启用——实现 SOA 治理模型
- 度量——改进 SOA 治理模型

SOA 治理方面列表

- 服务定义（服务的范围、接口和边界）
- 服务部署生命周期（各个生命周期阶段）
- 服务版本治理（包括兼容性）
- 服务迁移（启用和退役）
- 服务注册中心（依赖关系）
- 服务消息模型（规范数据模型）
- 服务监视（进行问题确定）
- 服务所有权（企业组织）
- 服务测试（重复测试）
- 服务安全（包括可接受的保护范围）

服务定义

- SOA 治理最基础的方面就是监视服务的创建过程。必须对服务进行标识，必须描述其功能，确定其行为范围并设计其接口。治理 COE 可以不执行这些任务，但要确保有人执行这些任务。COE 对创建服务和需要使用这些服务的团队进行协调，以确保满足相关的需求，并避免重复工作。
- 功能应该与一组可重复的业务任务匹配。服务的边界应当封装一项可重用、不受上下文约束的功能。接口应该公开服务进行的工作，但要隐藏服务是如何实现的，并允许更改实现或采用替代实现。从头设计服务时，可以将其设计成对业务进行建模；包装现有服务时，创建并实现好的业务接口可能会更难。
- 定义服务边界的潜在困难的一个有意思的例子就是在何处设置事务边界。服务通常在自己的事务中运行，确保其工作要么完全完成，要么完全回滚。不过，服务协调程序（也称为协调器或编排器）可能希望在单个事务中调用多个服务（最好是通过指定的交互，如 WS-AtomicTransactions）。此任务要求服务接口公开其事务支持，以便能参与调用方的事务。但这样的公开要求对调用方信任，对提供者具有很大的风险。例如，提供者可能会锁定执行服务的资源，但如果调用方永远不完成事务（没有提交或回滚），提供者将很难干净地释放资源锁定。正如这种情况所表明的，服务的范围以及谁具有控制权有时候并不太容易确定。

服务部署生命周期

服务和任何软件一样，需要进行规划、设计、实现、部署、维护，并最后退役。应用程序生命周期可以为公开性的，从而影响组织的很多方面；但服务的生命周期的影响甚至可能更大，因为多个应用程序可能会依赖于单个服务。

服务开发生命周期通常都具有五个主要的阶段：

- **已计划。**已标识了新服务并正在设计中，不过尚未实现或正在实现中。
- **测试。**实现后，必须对服务进行测试（稍后将对测试进行更详细的说明）。有些测试可能需要在生产环境中执行，此环境会将服务作为活动服务处理。
- **活动。**这是服务可供使用的阶段，我们通常所谈说的服务实际是处于此阶段的服务。这是一个服务，处于可用状态，在实际运行并且确实可完成相应的工作，而且尚未退役。
- **已弃用。**此阶段描述仍然处于活动状态但不会再存在很长时间的服务。这将警告使用者停止使用此服务。
- **已退役。**这是服务的最后一个阶段，表示一个不再提供的服务。注册中心可以保存有关曾经处于活动状态但不再可用的服务的记录。此阶段是不可避免的，不过很多提供者或使用者都未将此阶段纳入计划中。

退役可以有效地关闭服务版本，应该事先对退役数据进行计划和公布。在退役前，服务应在一段合适的时间内处于已弃用状态，以便以编程方式警告用户，从而使他们据此进行相应的计划。弃用和退役计划应在 SLA 中指定。

服务版本治理

- 提供服务后不久，这些服务的用户就开始需要进行一些相应的更改。需要对问题进行修复，需要添加新功能，需要重新设计接口，还需要删除不需要的功能。服务反映业务的情况，因此，随着业务发生变化，服务也需要进行相应的更改。
- 不过，对于服务的现有用户，需要采用巧妙的方式进行更改，以便不会干扰他们的成功操作。同时，现有用户对稳定性的需求不能对用户希望使用其他功能的需求造成障碍。
- 服务版本治理可满足这些相互矛盾的目标。版本治理允许满足服务现有功能的用户继续以不变的方式使用服务，并同时允许对服务进行改进，以满足具有新需求的用户。当前服务接口和行为将保留为一个版本，而同时会将较新的服务作为另一个版本引入。版本兼容性允许使用者调用不同但兼容的服务版本。

服务迁移

- 即使使用版本治理，使用者也不能期望永远提供和支持某个服务（或更准确地说，希望使用的一个服务版本）。服务提供者最终一定会停止提供此服务。版本兼容性可以帮助延迟这个“最后审判日”，但却不能消除这个问题。版本治理并不会使服务开发生命周期过时，而会允许生命周期扩展到多个连续的代。
- 当使用者开始使用服务时，将会创建对该服务的依赖关系，必须对此依赖关系进行管理。管理技术可用于有计划地周期性迁移到服务的较新版本。此方法还允许使用者利用添加到服务的新功能。
- 不过，即使在采用了最佳治理的企业中，服务提供者也不能仅依赖于使用者迁移。由于各种原因（遗留代码、人力、预算、优先级），一些使用者可能无法及时地进行迁移。这是否意味着提供者必须永远支持相应的服务版本？提供者是否可以在所有使用者已进行了迁移后直接禁用相应的服务版本？
- 这两个极端都不甚合意。一个不错的折衷办法是为每个服务版本采用有计划的弃用和退役计划，如服务部署生命周期中所述。

服务注册中心

- 服务提供者如何提供和宣传其服务？服务使用者如何查找其希望调用的服务？这些都在服务注册中心的职责范围内。服务注册中心将担当可用服务清单的角色，并提供调用服务的地址。
- 服务注册中心还可以帮助进行服务版本的协调工作。使用者和提供者可以指定它们需要或提供的版本，注册中心将随后确保仅列举出使用者所需版本的提供者。注册中心可以管理版本兼容性，跟踪版本间的兼容性并列举出使用者所需的版本或兼容版本的提供者。注册中心还可以支持服务状态，如测试状态和（如前面提到的）已弃用状态，且仅向希望使用服务的使用者提供具有这些状态的服务。
- 当用户开始使用服务时，将在此服务上创建一个依赖关系。每个使用者清楚地知道其所依赖的服务，但在企业全局范围内，这些依赖关系可能很难检测，更谈不上管理了。注册中心不仅列出服务和提供者，还可以跟踪使用者和服务间的依赖关系。这个跟踪功能可以帮助回答一个很古老的问题：*谁在使用此服务？*可识别依赖关系的注册中心能向使用者通知提供者方面的更改情况，如某个服务变成了已弃用状态。
- IBM 的 WebSphere Service Registry and Repository 是一款用于实现服务注册中心的产品。它担当服务定义的存储库以及这些服务的提供者的注册中心角色。它提供了一个中心目录，供开发人员查找可重用服务，并且在运行时使用，以便服务使用者和企业服务总线 (ESB) 查找服务提供者及调用服务的地址。

服务消息模型

- 在服务调用中，使用者和提供者必须就消息格式达成一致。当独立开发团队分开设计两个部分时，他们很容易陷入难于就公共消息格式达成一致的困境。再加上数十个使用典型服务的应用程序和使用数十个服务的典型应用程序，您就不难理解直接协商消息格式如何会变成需要全力投入来完成的任务。
- 用于避免消息格式混乱的一种常见方法是使用规范数据模型。规范数据模型是一组公共数据格式，独立于任何应用程序，由所有应用程序共享。这样，应用程序就不必就消息格式达成一致，可以直接同意使用现有规范数据格式。规范数据模型处理消息中的数据格式，因此仍然需要对消息格式的其他部分达成一致（如 Header 字段、消息有效负载包含什么数据以及数据如何安排），但规范数据模型可极大地促进协议的达成。
- 中央治理委员会可以充当中立方，负责开发规范数据模型。作为应用程序调查和服务设计工作的一部分，还可以设计在服务调用中使用的公共数据格式。

服务监视

- 如果服务提供者停止工作，您如何知道呢？是否要等到使用这些服务的应用程序停止，使用这些应用程序的人开始抱怨，才会知道服务已停止了呢？
- 结合使用多个服务的组合应用程序的可靠性只与其依赖的服务的可靠性相当。由于多个组合应用程序可能共享一个服务，单个服务出现故障可能会影响很多应用程序。必须定义 SLA，以描述使用者可以依赖的可靠性和性能。必须监视服务提供者，以确保其达到了所定义的 SLA。
- 一个与此相关的方面是问题确定。当组合应用程序停止工作时，确定为什么会这样。这可能是由于应用程序用于与用户进行交互的 UI 停止了运行造成的。但也可能 UI 运行正常，但所使用的其他服务或这些服务使用的某些服务未正常运行。因此，务必注意，不仅要监视每个应用程序的运行状况，还要监视每个服务（提供者整体）和各个提供者的运行状况。单个业务事务中服务间的事件的相关性非常重要。
- 此类监视可以帮助在问题出现前检测和防止问题。可以检测不均衡和停机状况，能在这些情况造成重大影响前发出警告，甚至可以尝试自动纠正问题。可以对长时间的使用情况进行度量，以帮助预测使用率会变得更高的服务，以便为其提供更高的容量。

服务所有权

- 当多个组合服务使用一个服务时，谁负责此服务？是那个人或组织负责它们吗？如果是其中一个，是哪一个？其他人是否认为他们拥有这个服务的所有权？
- 无论是社区公园、可重用 Java 框架，还是服务提供者，任何共享资源都很难获得和得到相应的照顾。不过，所需的采用池化技术的资源可提供远远超过任何参与者成本的价值：公路系统就是这样。
- 企业通常围绕其业务操作组织员工报告结构和财务报告结构。SOA 采用相同的方式围绕相同的操作组织企业的 IT，负责特定操作的部门也可以负责这些操作的 IT 的开发和运行。该部门拥有这些服务。SOA 中的服务和组合应用程序经常并不遵循企业严格的层次报告和财务结构，从而在 IT 责任方面造成空白和重叠。
- 一个相关的问题就是用户角色。由于 SOA 的重点是保持 IT 和业务的一致，而另一个重点是企业重用，组织中很多不同的人员对什么将成为服务、这些服务如何工作以及将如何使用都有发言权。这些角色包括业务分析人员、企业架构师、软件架构师、软件开发人员和 IT 管理员。所有这些角色在确保服务正确为企业需求和工作服务方面都负有责任。
- SOA 应该能反映其业务。这通常意味着要更改 SOA 来适应业务，但在这种情况下，可能有必要更改业务来与 SOA 匹配。如果不可能，则需要提高多个部门间的合作水平，以分担开发公共服务的任务。这样的合作可以通过跨组织的独立委员会来实现，此委员会实际上拥有服务，并对其进行管理。

服务测试

- 服务部署生命周期包括测试阶段，在此阶段，团队将在激活服务前确认服务能正确工作。如果测试了服务提供者，且表明其工作正常，使用者是否需要也对其进行重新测试？是否采用同样的严格要求对服务的所有提供者进行测试？如果服务更改，是否需要对其进行重新测试？
- SOA 增加了以独立的方式测试功能的机会，并提高了对其按预期工作的期望值。不过，SOA 也为每个不一定信任服务的新使用者提供了重新测试相同功能的机会，以便确定服务一致地工作。同时，由于组合应用程序共享服务，单个存在错误的服务就可以对一系列看起来不相关的应用程序造成负面影响，从而扩大这些编程错误的后果。
- 为了利用 SOA 的重用好处，服务使用者和提供者需要就提供者合理的测试级别达成一致，并需要确保测试按照双方达成的协议执行。然后，服务使用者只需要测试自己的功能以及到服务的连接，并假定服务将按照预期的方式工作。

服务安全

- 是否允许任何人调用任何服务？具有一系列用户的服务是否允许其所有用户访问所有数据？服务使用者和提供者之间交换的数据是否需要进行保护？服务是否需要足够的安全，以满足其最偏执的用户或最懒散的用户的需求？
- 对于任何应用程序，安全是一个困难但必要的命题。功能需要限制为授权的用户，需要对数据进行保护，以防止被窃听。通过提供更多的功能访问点（即服务），SOA 有可能会大幅度增加组合应用程序中的漏洞。
- SOA 可创建非常容易重用的服务，甚至哪些不应该重用这些服务的使用者也能对其进行重用。即使在授权用户中，并非所有用户都应该具有服务能够访问的所有数据的访问权。例如，用于访问银行帐户的服务应该仅提供特定的用户帐户（尽管其代码也具有访问其他用户的其他帐户的权限）。一些服务使用者比相同服务的其他使用者具有更大的数据保密性、完整性和不可否认性。
- 服务调用技术必须能够提供所有这些安全功能。必须对服务的访问进行控制，且将访问权仅限于授权使用者。用户标识必须传播到服务中，并用于授权数据访问。数据保护的质量必须表示为相应范围内的策略。这就允许使用者说明最低级别的保护和最大功能，并能与可能实际包含其他保护的相应提供者进行匹配。

参考资料

SOA 治理

IBM SOA 治理生命周期。
IBM SOA 治理和管理方法。
Advancing SOA Governance and Service Lifecycle Management。
IBM SOA Governance 和 New IBM Software and Consulting Services Help Organizations Reach Business Goals 新闻稿（2006 年 3 月 22 日）
访问 developerWorks SOA and Web Services 专区，包括 SOA and Web services 新手入门。
推荐阅读清单：面向服务的体系结构和 WebSphere Process Server（IBM developerWorks，2006 年 4 月）
阅读文章“什么是 IT 治理，为什么应该对其加以注意？”（IBM developerWorks，2006 年 4 月），了解关于治理的更多信息。
阅读 Tilak Mitra 的文章“SOA 治理案例”（IBM developerWorks，2005 年 8 月），了解关于 SOA 治理的更多信息。
评论专栏: Scott Simmons：SOA 治理与预防面向服务的混乱。
Rational Method Composer plug-in for SOA Governance。
IBM SOA/Web Services Center of Excellence (PDF)。
阅读 Ali Arsanjani 和 Kerrie Holley 的文章“Increase flexibility with the Service Integration Maturity Model (SIMM)”（IBM developerWorks，2005 年 9 月）。
访问 Ali Arsanjani 的 Blog：“Practices in Service-Oriented Architecture”
访问 Bobby Woolf 的 Blog：“WebSphere SOA and J2EE in Practice”
访问 developerWorks Architecture 专区，阅读更多有关治理的文章。

服务定义

阅读“如果刚刚采用 SOA，最好将哪些软件作为服务实现？”，了解有关服务定义的更多信息。（IBM developerWorks，2006 年 2 月）
“SOA 实现：服务设计原则”，（IBM developerWorks，2006 年 2 月，作者：David J.N. Artus）
“迈向面向服务的体系结构和集成的模式语言，第 1 部分：构建服务生态系统”（IBM developerWorks，2005 年 7 月，作者：Ali Arsanjani）

服务版本治理

“WebSphere Process Server 的版本治理和动态性”（IBM developerWorks，2006 年 2 月，作者：Richard G. Brown）和“SOA programming model, Part 5:Managing change in Web services components and applications”（IBM developerWorks 技术讲座，2005 年 11 月，作者：Richard G. Brown）
“Web 服务版本治理最佳实践”（IBM developerWorks，2004 年 1 月，作者：Kyle Brown 和 Michael Ellis）
“ Web 服务的局部向后兼容 (minor backward-compatible)”（IBM developerWorks，2004 年 11 月，作者：Russell Butek）

服务注册中心

WebSphere Service Registry and Repository
IBM WebSphere Service Registry and Repository 简介。
“选择适合您的业务模型的 ESB 拓扑”（IBM developerWorks，2006 年 7 月，作者：Chris Nott 和 Marcia Stockton）
“SOA 反模式：成功采用和实现面向服务的体系结构的障碍”（IBM developerWorks，2005 年 11 月，作者：Jenny Ang、Luba Cherbakov 和 Mamdouh Ibrahim）

服务消息模型

Enterprise Integration Patterns（Addison-Wesley，2003 年，作者：Gregor Hohpe 和 Bobby Woolf）

服务监视

“Monitor business IT services using IBM Tivoli Monitoring for Transaction Performance”（IBM developerWorks，2005 年 6 月，作者：Wilfred Jamison 和 Richard Duggan）

服务所有权

“用于实现 Web 服务的 SOA 编程模型，第 10 部分：SOA 用户角色”（IBM developerWorks，2006 年 2 月，作者：Mandy Chessell 和 Birgit Schmidt-Wesche）

服务测试

“通过服务模拟来简化 SOA 开发”（IBM developerWorks，2005 年 12 月，作者：Bobby Woolf）

服务安全

“通过 WebSphere Application Server V6 实现 Web 服务安全，第 1 部分：安全体系结构介绍”（IBM developerWorks，2006 年 4 月，作者：Tony Cowan）a