山东建筑大学

本科毕业设计

（本科毕业论文）

外文文献及译文

文献、资料题目：Web Services, Service-Oriented computing, and Service-Oriented Architecture :Separating Hype from reality

文献、资料来源：Journal of Database Management

文献、资料发表（出版）日期：2008.07.01

院　（部）：计算机科学与技术学院

专　　业：网络工程

班　　级：152

姓　　名：王斌

学　　号：201511102016

指导教师：刘刚、秦松

翻译日期：2019.02.05

外文文献：

Web Services, Service-Oriented computing, and Service-Oriented Architecture: Separating Hype from reality

John Erickson, University of Nebraska - Omaha, USA

Keng Siau, University of Nebraska - Lincoln, USA

Abstract

Service-oriented architecture (SOA), Web services, and service-oriented computing (SOC) have become the buzz words of the day for many in the business world. It seems that virtually every company has implemented, is in the midst of implementing, or is seriously considering SOA projects, Web services projects, or service-oriented computing. A problem many organizations face when entering the SOA world is that there are nearly as many definitions of SOA as there are organizations adopting it. Further complicating the issue is an unclear picture of the value added from adopting the SOA or Web services paradigm. This article attempts to shed some light on the definition of SOA and the difficulties of assessing the value of SOA or Web services via return on investment (ROI) or nontraditional approaches, examines the scant body of evidence empirical that exists on the topic of SOA, and highlights potential research directions in the area.

Keywords: service-oriented architecture; service-oriented computing; Web services

Background and History of Service-oriented Architecture

A minimum of nine formal definitions of SOA exist as of this writing, from sources such as the Organization for the Advancement of Structured Information Standards (OASIS), the Open Group, XML.com, Javaworld.com, Object Management Group (OMG), the World Wide Web Consortium (W3C), Webopedia, TechEncyclopedia, WhatIs.com, and Webopedia.org. In addition,

many other definitions put forth by numerous industry experts, such as those from IBM, further cloud the issue, and worse yet, other formal definitions might also exist. In other words, the concept of serviceoriented architecture appears in many ways to be a virtually content-free description of an IT-based architecture. It is not our intent here to add yet another definition to this already crowded arena of definitions, but to try tocull the common, base meanings from the various distinct definitions.

Prior to about 2003, the term service-oriented architecture was not in general use for the most part, according to Wikipedia ("SOA," 2007). However, since that time, SOA has exploded nearly everywhere in the business and technology world. SOA appears to derive or develop in many cases from more basic Web services. These services can include enabling technologies such as SOAP, CORBA, EJB (Enterprise Java Beans), DCOM (distributed component object model), and even SIP (session-initiated protocol) among many others; services may also include other middleware created with XML (Lee, Siau, & Hong, 2003; Siau & Tian, 2004; Sulkin, 2007; Walker, 2007).

Service-Oriented Architecture Definitions

The Open Group (2007) defines SOA as "an architectural style that supports service orientation." The definition goes on to also include descriptions of architectural style, service orientation, service, and salient features of SOA. OASIS defines SOA as "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains." The OASIS definition includes what they call a "reference model" in which the details of the definition are expanded and formalized. The Object Management Group (2007) defines SOA as "an architectural style for a community of providers and consumers of services to achieve mutual value." OMG adds that SOA allows technical independence among the community members, specifies the standards that the (community) members must agree to adhere to, provides business and process value to the (community) members, and "allows for a variety of technologies to facilitate (community) interactions" (OMG, 2007).

W3C (2007) defines SOA as "a form of distributed systems architecture that is typically characterized by…a logical view, a message orientation, a description orientation, granularity and platform neutrality." W3C adds details describing what it means by logical view, message and

description orientations, granularity, and platform neutrality. XML.com (2007) defines SOA as follows:

SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents. A service is a unit of work done by a service provider to achieve desired end results for a service consumer. Both provider and consumer are roles played by software agents on behalf of their owners.

The Javaworld.com SOA definition, composed by Raghu Kodali (2005), is as follows: “Service-oriented architecture (SOA) is an evolution of distributed computing based on the request/reply design paradigm for synchronous and asynchronous applications.” Kodali also goes on to describe four characteristics of SOA. First, the interfaces composed in XML, using WSDL (Web services description language), are used for self-description. Second, XML schema called XSD should be used for messaging. Third, a UDDI-based registry maintains a list of the services provided. Finally, each service must maintain a level of quality defined for it via a QoS (quality of service) security requirement.

Finally, IBM proposes that SOA “describes a style of architecture that treats software components as a set of services” (UNL-IBM System in Global Innovation Hub, 2007). Furthermore, it insists that business needs should “drive definition” of the services, and that the value proposition be centered on the reusability and flexibility of the defined services.

Service-Oriented Architecture

We begin the SOA discussion with an overview of SOA provided by Krafzig, Banke, and Slama (2005). They proposed a three-level hierarchical perspective on SOA in which Level 1 includes the application front end, the service, the service repository, and the service bus (SB). Accordingly, only the service child has children, consisting of the contract, implementation, and interface. Finally, the last level of the proposed hierarchy is composed of business logic and data, children of implementation. The next subsections will discuss the general ideas of the elements included in the hierarchy proposed by Krafzig et al. described previously. This is not to recommend adoption of the hierarchy and description as the final description of SOA, but rather as a framework for discussing the meaning of SOA for the remainder of this article.

Application Front end

This part of SOA comprises a source-code interface, and in SOA terminology, it is referred to as the application programming interface (API). In accordance with most commonly accepted design principles, the underlying service requests, brokerage (negotiation), and provision should be transparent to the end user.

Service repository

The service repository could be thought of as the library of services offered by a particular SOA. This would likely consist of an internal system that describes the services, and provides the means in the user interface to call a particular service. UDDI could be seen as a realization of the service repository idea. UDDI is a global registry that allows businesses to list themselves on the Internet. UDDI is platform independent and XML based. The point of UDDI is for businesses to list the Web or SOA-type services that they provide so that other companies searching for such services can more easily locate and arrange to use them.

Service Bus

The SB, more commonly referred to as the enterprise service bus (ESB), provides a transportation pathway between the data and the end-user application interface. Using an ESB does not necessarily mean SOA is being implemented, but ESB or some sort of SB use is almost always part of an SOA deployment. According to Hicks (n.d.), Oracle's idea of anESB includes multiple protocols that "separate integration concerns from applications and logic." What this means is that ESBs have now become commercialized, and can be licensed for use much like other UDDI-based services. So, companies searching for ESB solutions as part of an SOA effort now have multiple choices and do not necessarily have to re-create the wheel by building their own ESB.

Common Services

It seems apparent from many of the SOA definitions that many of the technologies included in an SOA definition, and by default SOA implementations, are established and conventional protocols. To better understand the services provided in many SOA definitions, a brief explanation of some of the more commonly used underlying technologies is provided. A particular service may or may not be explicitly Web based, but in the end it matters little since the services provided by the

architecture should be transparently designed, implemented, and provided. The general consensus from most involved in Web services is that the services are meant to be modular. This means that no single document encompasses all of them, and furthermore, that the specifications are multiple and (more or less) dynamic. This results in a small number of core specifications. Those core services can be enhanced or supported by other services as “the circumstances and choice of technology dictate” (“Web Service,” 2007).

XML allows users to define and specify the tags used to capture and exchange data, typically between distinct and usually incompatible systems from different companies or organizations. This means that XML is a good example of middleware; it also means that XML enables Web services. XML was one of the initial drivers that provided the ability to conduct e-business for many businesses in the Internet era. XML cannot really be considered a service, but as the language used to write many of the Web services or service stack protocols.

SOAP, like all protocols, consists of a set list of instructions detailing the action(s) to be taken in a given circumstance. SOAP is designed to call, access, and execute objects. The original SOAP was typically for communications between computers, and usually involved XML-based messages. SOAP and its underlying XML programming comprised one of the first Web service communication stacks. One of the original Web services that SOAP provided was called remote procedure call (RPC), which allowed a remote computer to call a procedure from another computer or network. More recently, SOAP has taken on a somewhat modified meaning so that the acronym now means service-oriented architecture protocol. In both cases, what SOAP does is to use existing communications protocols to provide its services. The more common early SOAP contracts included XML applications written for HTTP (hypertext transfer protocol), HTTPS (HTTP over secure socket layer), and SMTP (simple mail transfer protocol), among others. It should be apparent from these that many early SOAP implementations involved e-commerce or e-business applications, which means that the concern at the time when many applications were first developed was to move sales and other data collected in Web portals to back-end data stores.

CORBA is an OMG-developed standard that allows different software components that are usually written in different languages and nstalled on different computers to work together (Zhao & Siau, 2007). CORBA was developed in the early 1990s, and while not overtly an SOA at the time, it actually performs many of the functions in an SOA, using an IIOP- (Internet inter-orb

protocol) based service stack.

EJB is a component typically situated on the server that “encapsulates the business logic of an application” (“EJB,” 2007). EJB enables the creation of modular enterprise (and other) applications. The intent of EJB is to facilitate the creation of middleware that acts as a go-between tying front-end applications to back-end applications or data sources.

SIP is a signaling protocol designed for use in telecommunications at the application layer. It has generally become one of the primary protocols used in VoIP (voice over Internet protocol), H.323, and other communications standards. SIP can be seen as a primary provider of Web services for Internet-based voice communications such as VoIP (Sulkin, 2007).

Contract (Services)

Components of a service contract typically include primary and secondary elements. The primary elements consist of the header, functional requirements, and nonfunctional requirements. Subelements for the header consist of the name, version, owner, RACI, and type. Under functional requirements are functional requirement descriptions, service operations, and invocation. Nonfunctional requirements include security constraints, QoS, transactional requirements (the service part of a larger transaction), service-level agreement, and process (“SOA,” 2007). The contract generally includes metadata about itself, who owns it, and how it is brokered, bound, and executed.

Interface

At this level of service provision, the interface referred to is a segment of code that connects the service with the data and/or business logic (process). The interface describes how data will be moved into and out of the data source by the service, and must be designed to comply with the physical (data, data structures, etc.) and process (business logic) requirements of the existing and/or legacy system.

Implementation

The implementation specifies the contract and interface to be used for each service requested, and contains the direct pathway into the data and business logic.

Architecture

The service component of SOA has been discussed, though admittedly at a high level. However, the architecture component has not yetbeen addressed and it will be helpful to speak briefly about the architecture segment of SOA. Architecture in general refers to the art (or science) behind the design and building of structures. Alternatively, an architecture may refer to a method or style of a building or a computer system. So, if SOA is taken literally as a description of its function, it could be taken to mean a structured way of organizing or arranging the services in a business or organization.

SOA Framework

It is apparent from the existing definitions and models that service-oriented architecture is commonly seen as an architecture or way of assembling, building, or composing the information technology infrastructure of a business or organization. As such, SOA is not a technology in itself; rather, it is a way of structuring or arranging other technologies to accomplish a number of other tasks. Thisnaturally leads to the problem of a multiplicity of definitions of SOA since many relatively similar structural arrangements of services are possible. Many of the definitions also indicate that the arrangement and relationships between modules

Undefined
Service Oriented Architecture SOA
Services： 1.physical Transport Schema 2.API (Application programming Interface) 3. Source (Service Library or repository; UDDI, internal or external)

Undefined		
BusinessLogic (Process or Activity)		Data (Connection to datasources:internal or extranal)

should be loosely coupled rather than tightly coupled. This allows for customization of services based on need, and on-demand rather than some predetermined structure, but the downside is that it also leads toward a plethora of definitions and approaches to SOA implementation.

Some of the common features that seem sensible to include in a formal definition of SOA would relate to a common framework, such as that specified by Krafzig et al. (2005) or one of the other standards bodies. In other words, a framework would include metadata describing the various important features of SOA, how those features can be arranged, and the libraries or location of services that allow adopting organizations to arrange bindings or contracts between themselves and the service provider, independent of whether the service provider is internal or external. We propose the framework depicted in Figure 1 as a starting point forSeveral of the standards bodies have taken a stance in creating or calling for a metamodel, at least in some form. Among them are the Open Group, OASIS, OMG,W3C, and to a lesser extent industry-related bodies such as Javaworld. com, XML.com, IBM, and Oracle.

UDDI has become a very well-known structured repository for services and service components, which speaks to the universality of the library or centralized database of services. However, more standardization efforts will be necessary to enhance the interoperability of UDDI.

It also appears, especially with the industry definitions of SOA, that the contracts, bindings, interfaces, service buses, and other implementation-related portions of SOA are important elements to be considered when attempting to give an overall definition of SOA. This unfortunately could easily represent a stumbling block in garnering consensus on a definition of SOA since each of these companies has invested significant time, human, and other likely resources toward development of their specific pieces of the SOA pie. Each company has invested heavily and thus will likely be less willing to risk that investment and any potential return and customer lock-in in order to simply agree on standards. We observed a similar

occurrence of this type of behavior in the recently ended format war in the high-definition DVD market. Similarly, if the standards bodies have political or industry leanings, agreement on a common SOA definition and standards could be difficult to achieve.

Another more recent development comes from Shah and Kalin (2007). They proposed that organizations adopting SOA follow a specific path based on an analysis of business challenges, including SOA business drivers and IT barriers. This led them to speculate that a specific adoption model be used to guide the SOA implementation process. They indicated that an ad hoc SOA model is better where the benefits of new services are specific to each individual service, where the technologies may be inconsistently applied (different implementations for the same service in different projects), where services cannot be reused, and where the increases in technological complexity translate into decreased system response times. Shah and Kalin ended with a call for a strategy- or program-based SOA adoption model that is situational.

We propose that a common definition of SOA is possible and necessary, and call for negotiations among interested bodies with the aim of reaching a common definition of SOA. We realize that in practice it might prove difficult or even nearly impossible to expect such a consensus to be arrived at, but a common definition and structure of SOA would go a long way toward dealing with some of the confusion, misinformation, and hype regarding the entire subject. Difficult though it might be to expect this, a realization that SOAP, CORBA, RPC, and XML among many other technological tools have reached a point of relative agreement amongst users if not ubiquity, at least related to their underlying standards, should provide some evidence that agreements can be reached. Next, we will examine SOA from the research perspective.

Possibilities for research

Research into SOA is extremely limited at this point in time. What studies exist can be classified into several distinct categories. The first includes exploratory or recommendation-type efforts that propose various means to approach SOA implementation. These investigations may or may not include proprietary industry software, but most of these research efforts propose the use of patterns or blueprints and a metamodel of SOA as a means to understanding the SOA perspective. Second, in this category are research proposals that examine company-specific technologies or tools (i.e., IBM proposing the use of Rational Software, including the Rational

Unified Process) in relation to SOA design and implementation. Neither of the first two types of SOA research generally involve ideas on how to measure SOA in terms of success or failure, or even suggest metrics. Finally, the third type of research articles focus on empirical research.

中文译文：

Web 服务、面向服务的计算和面向服务的体系结构：将宣传与现实分开

约翰.埃里克森，内布拉斯加大学-奥马哈分校,美国

耿秀，内布拉斯加大学-林肯分校,美国

摘要

面向服务的体系结构(SOA)、Web 服务和面向服务的计算(SOC)已经成为当今商界许多人的热门词汇。似乎几乎每个公司都已经实现了 SOA 项目、Web 服务项目或面向服务的计算，或者正在实现 SOA 项目、Web 服务项目或面向服务的计算。许多组织在进入 SOA 世界时面临的一个问题是，采取 SOA 的组织几乎和定义 SOA 的组织一样多。进一步使问题复杂画的是，采取 SOA 或 Web 服务范式所带来的附加值前景并不明确。本文试图阐明的是 SOA 的定义以及解决 SOA 的价值评估的困难和 Web 服务的投资回报的非传统方法，并且强调了存在于 SOA 中潜在的研究方向。

关键词：面向服务的体系结构；面向服务的计算；Web 服务

面向服务的体系结构的背景和历史

在撰写本文的时候，至少已经有了 SOA 的正式定义，他们分别来自于结构化信息标准促进组织(OASIS)、厂家技术中立联合协会、XML 网、Java 世界网、项目管理组织、万维网联盟。此外，许多行业专家(例如 IBM 的专家)提出的定义更是进一步使概念变得模糊，而且更糟糕的是可能存在其他正式定义。换句话说，面向服务的体系结构的概念在很多方面

相对于 IT 的体系结构并没有十分明确的内容描述。根据维基百科的说法，在 2003 年之前，面向对象的服务体系结构这个术语在大多数情况下并不常用。然而，从那时起，SOA 在商业和技术诸多方面都呈现出了爆炸式的增长。在多数情况下，SOA 似乎是从更基本的 Web 服务派生或开发的。这些服务包括诸如 SOAP、CORBA、EJB(企业 Java Bean)、DCOM(分布式组件对象模型)甚至是 SIP(会话协议)等技术；服务也可能包括用 XML 创建的其他中间件。

面向服务的体系结构的定义

Open Group(2007)将 SOA 定义为“支持面向服务的体系结构样式”。该定义还包括对体系结构风格、面向服务、服务和 SOA 的显著特征的描述。OASIS 将 SOA 定义为一个组织和利用在不同所有权控制下的分布式功能的一个典例。OASIS 的定义包括所谓的“参考模型”，并扩展和形式化了其中定义的细节。对象管理组(2007)将 SOA 定义为“服务提供者和服务使用者实现相互价值的体系结构样式”。OMG 补充说 SOA 允许社区成员之间存在技术独立性，并且“允许各种技术来促进社区进行交互”。

W3C 将 SOA 定义为“一种分布式系统架构的形式，其中典型特征就是逻辑视图、消息方向、描述方向、粒度和平台中立性”。W3C 添加了通过逻辑视图、消息和描述方向、粒度和平台中性来描述其含义的细节。XML 网站对 SOA 的定义如下：SOA 是一种体系结构风格，其目标是实现交互软件代理之间的松散耦合。服务是服务提供者为实现服务使用者所需的最终结果而完成的工作单元。提供者和使用者都是软件代理代表其所有者所扮演的角色。

由 Raghu Kodali(2005)编写的 Javaworld.com SOA 的定义如下：“面向服务的体系结构(SOA)是基于同步异步应用程序的请求/应答设计规范式的分布式计算的发展”。Kodali 还描述了 SOA 的四个特征。首先，使用 WSDL(Web 服务描述语言)组成的 XML 接口用于自我描述。其次，应该使用名为 XSD 的 XML 模式进行消息传递。第三，基于 uddi 的用户注册中心维护所提供的服务列表。最后这个服务必须维护通过 QoS(服务质量)安全需求为其定义的质量级别。

面向服务的体系结构

我们从 Krafzig、Banke 和 Slama(2005)提供的 SOA 概述开始 SOA 讨论。他们对 SOA 提

出了一个三层的分层透视图，其中第一层包括应用程序前端、服务、服务存储库和服务总线(SB)。因此，只有服务子节点有子节点，有协议、实现和接口组成。最后，层次结构的最后一层由业务逻辑和数据(实现子层)组成。下一小节将讨论 Krafzig 等人提出的层次结构中所包含的思想。这并不是建议将层次机构和描述作为 SOA 的最终描述，而是将其作为一个框架，在本文的其余部分将讨论 SOA 的含义。

前端应用程序

SOA 的这一部分包括一个源代码接口，在 SOA 的术语中，它被称作应用程序编程接口(application programming interface,API)。根据最普遍接受的设计原则，底层服务请求、中介(协商)和规定应该对最终用户透明。

服务存储库

可以将服务存储库看作 特定的 SOA 提供的服务库。这包括描述服务的内部系统，并在用户界面中提供调用它的服务的方法。UDDI 可以看作是服务存储库思想的实现。UDDI 是一个全球注册中心，允许企业在因特网上展示它们自己。UDDI 是独立于平台和 XML 的。UDDI 的目的是让企业列出它们提供的 WEB 或 SOA 类型的服务，以便其他搜索此类服务的公司能够更加容易的定位和使用它们。

服务总线

SB，通常称为企业服务总线(enterprise service bus,ESB),提供了数据和最终用户应用程序接口之间的传输路径。使用 ESB 并不意味着就实现了 SOA，但是 ESB 或者说某种 SB 的使用几乎总是 SOA 部署的一部分。根据 Hicks 的说法，Oracle 的 ESB 概念包括多个协议，“将关注点从应用程序和逻辑中分离出来”。这意味着 ESB 现在已经商业化可以像其他基于 UDDI 的服务一样获得使用许可。因此，搜寻 ESB 解决方案作为 SOA 工作的公司现在有多种选择，不必通过构建自己的 ESB 重新创建。

公共服务

从许多 SOA 定义中可以明显看出，SOA 定义和默认的 SOA 实现中包含着许多技术都是建立和传统的协议。为了更好的理解许多 SOA 中提供的服务，本文提供了一些更为常用的底层技术的一些说明。一个特定的服务可能是基于 WEB 的，也有可能不是，但这并不重要，

因为体系结构提供的服务应该是透明的。大多数从事 WEB 服务相关人员的普遍共识是，服务应该是模块化的。这意味着并没有一个文档能够包含所有的规范，而且规范也不是一成不变的。这就导致了仅有少量的核心规范。这些核心服务可以被其他服务增强或者支持，如“环境和技术选择要求”。

XML 允许用户定义和指定用于捕获和交换的数据标记，这些标记通常在来自不同公司或者组织的不同系统之间使用，通常不兼容。这意味着 XML 是中间件的一个很好的例子；这还意味着 XML 支持 WEB 服务。XML 是最初的驱动程序之一，它为 Internet 时代的许多企业提供了电子商务的能力。XML 不能被认为是一种真正的服务，而是一种用于编写许多 WEB 服务或服务堆栈协议的语言。

SOAP 和所有协议一样，由一组指令组成，这些指令详细的描述了在给环境中要采取的操作。SOAP 被设计用来调用、访问和执行对象。最初的 SOAP 通常用于计算机之间的通信，通常涉及基于 XML 的消息。SOAP 及其底层的 XML 编程组了首批 WEB 服务通信栈之一。SOAP 提供的原始 WEB 服务之一称为远程过程调用(RPC),它允许远程计算机从另一台计算机或者网络进行调用。最近，SOPA 进行了一些协议的修改，所以这个缩写现在意味着面向服务的体系结构协议。在这两种情况下，SOAP 所做的就是使用现有的通信协议来为其提供服务。早期更为常见的 SOAP 协议包括 HTTP、HTTPS 和 SMTP 编写的 XML 应用程序。从这些方面可以明显看出，许多早期的 SOAP 实现都涉及电子商务或者电子商务应用程序，这意味着在最初开发许多应用程序的时候，所关心的是将 WEB 门户中收集的销售和其他数据转移到后端数据存储。

CORBA 是一个由 OMG 开发的标准，它允许使用不同的语言编写不同的软件组件安装在不同的计算机上一起工作。CORBA 是在 20 世纪 90 年代早期开发的，虽然当时还不是 SOA，但它实际上使用时基于 IIOP 的服务堆栈来执行 SOA 中的许多功能。

EJB 通常位于服务器上，“封装应用程序的业务逻辑”。EJB 支持创建模块化企业和其他应用程序。EJB 的目的是加快中间件的创建，中间件充当将前端应用程序绑定到后端应用程序或数据源的中间人。

SIP 是一种为在应用层的电信中使用而设计的信令协议。它已普遍成为 VoIP、H.323 和其他通信标准中使用的主要协议之一。SIP 可以被视为基于 Internet 的语音通信的 WEB 服务提供者。

合同(服务)

服务的契约组件通常包括主要和次要元素。主要元素包括头、功能需求和非功能需求。标题的子元素包括名称、版本、所有者、RACI 和类型。在功能需求之下是功能需求描述、服务操作和调用。非功能需求包括安全约束、QoS、事物需求、服务级别协议和流程。协议通常包括关于本身、谁拥有它以及如何代理、绑定和执行它的元数据。

接口

在服务提供的这个级别上，所引用的接口是将服务与数据和/或业务逻辑(流程)连接起来的一段代码。接口描述了服务如何将数据迁入和数据迁出，并且必须设计成符合现有和/或者遗留系统的元素(数据、数据结构等)和流程(业务逻辑)需求。

结构

该结构实现为每个服务指定所使用的协议和接口，并包含进入数据和业务逻辑的直接路径。

体系结构

SOA 的服务组件已经被讨论过了，尽管不可否认是在一个较高层次上。然而，体系结构组件还没有完全得到解决，简单的介绍一下 SOA 体系结构部分是有意义的。建筑学一般是指建筑结构设计和建造背后的艺术(或科学)。或者体系结构又可以指建筑物或计算机系统的方法和风格。因此，如果将 SOA 字面上理解为对其功能的描述，则可以将其理解为在业务中女排服务的结构化方式。

SOA 框架

从现有的定义和模型中可以明显的看出，面向服务的体系结构通常被视为一种体系结构或组装、构建或组合业务或组织信息技术基础设施的方式。因此，SOA 本身并不是一种技术；相反他是一种构造或者安排其他技术来完成任务的方法。这自然会导致 SOA 定义的多样性问题，因为许多相类似的服务结构进行工作是可能的。许多定义还指出模块之间的关系应该是松散耦合的而不是紧密耦合的。这允许根据需要和需求随机应变定制服务，但

是这个缺点也会导致 SOA 出现过多的定义和方法。

<table>
<tr><td colspan="3">未定义</td></tr>
<tr><td colspan="3">面向服务的体系结构 SOA</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">服务包括：

1. 物理传输模式
2. API(应用程序编程接口)来源(服务库或资料库)；UDDI(内部或外部)</td></tr>
<tr><td colspan="3">未定义</td></tr>
<tr><td>业务逻辑
(过程或活动)</td><td></td><td>数据
(与数据源的连接；内部或外部)</td></tr>
</table>

图 1

在 SOA 正式定义中包含的一些公共特性似乎与公共框架有关，换句话说，一个框架包括元数据描述 SOA 的各个重要功能，也就是说，组织和选择协议与服务。

一些标准机构在构建元模型时采取了一些特定的立场。其中包括开放组、OASIS、OMG、W3C、以及较小程度上与行业有关的组织机构，如 JavaWorld.com、XML.com、IBM 和 Oracle。

UDDI 已经成为服务和服务组件的非常著名的结构化存储库，这说明了服务库或集中式数据库的普遍性。然而，仍然还需要更多的标准化工作来增强 UDDI 的互相操作性。

此外，特别是在 SOA 行业的定义中，协议、绑定、接口、服务总线和其他实现部分似乎是要在给出 SOA 定义时要好好考虑的因素。不幸的是，这却很容易成为在 SOA 定义上取得一致意见的绊脚石。因为这些公司都在 SOA 的开发上，投入了大量的时间和精力。每家公司都投入了大量的资源，因此他们可能不太愿意为了简单的标准达成一致，就拿投资、潜在回报和客户冒险。在最近结束的高清 DVD 市场斗争中，我们也观察到了类似的情况。类似的，如果有标准机构具有政治或行业倾向，则很难就公共 SOA 的定义和标准达成一致。

另外一个 SOA 的发展是来自 Shan 和 Kalin。他们建议采用 SOA 的组织遵循基于业务挑战分析的特定路径，这导致他们推测，应该使用特定的模型来实现 SOA。他们表示一个特别的 SOA 模型是能更好的为服务技术不同的应用，在服务不能被重用，技术复杂性的转化为减少系统响应时间。Shah 和 Kalin 最后呼吁使用基于策略或编程 SOA 采用的模型。

我们建议，SOA 的公共定义是必要的，并呼吁相关机构进行协商，以达成 SOA 的公共定义。我们意识到，现实中，几乎不可能期望达成这样的共识。但是 SOA 公共定义和结构对于处理关于整个主题的一些混淆、错误宣传有很大的帮助。尽管可能很难预料到这一点，但是，认识到 SOAP、CORBA、RPC 和 XML 等许多技术工具已经在用户之间达成了相对一致，应该提供一些证据，证明可能达成一致。

可能性研究

目前对于 SOA 的研究十分有限，现在的研究可以分为几个不同的类别。第一包括探索性的或建议类型的研究，这些研究提出各种方法来实现 SOA。这些调查可能包括也可能不包括专有的软件行业，但是大多数研究都建议使用模式或蓝图以及 SOA 的元模型作为理解 SOA 透视图的方法。其次，在这一类别中，是对于研究特定公司的技术和工具。前两种类型的 SOA 研究通常都不涉及如何根据成功或失败来衡量 SOA 的思想，甚至也不建议用度量标准。