This article will discuss what Service Oriented Architecture (SOA) is, why and how it should be tested, and the challenges that SOA testers are likely to face. It will give you insight into the requirements of an SOA tester and the differences in SOA testing compared with traditional software testing.

**Asanka Vithanage**  
Senior Software Engineer-QA  
WSO2

Table of contents

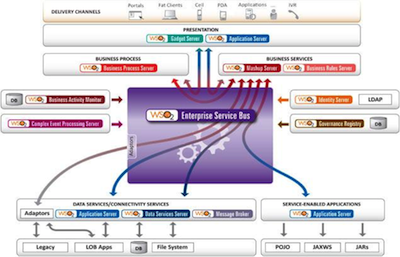
* What is SOA?
* What is SOA testing?
* Importance of SOA testing
* Challenges of SOA testing
* How SOA testing differs from traditional testing
* SOA testers and their testing psychology
* SOA testing lifecycle
* SOA testing tools
* Automation of SOA

What is SOA?

Modern day software solutions hardly work in a standalone mode and must have the ability to securely integrate with disparate data sources and different ecosystems. Handling increasing and changing user expectations can be considered as the key problem that modern enterprise software solutions are trying to resolve.

SOA has emerged as a solution and is considered as an architectural design pattern that can overcome these problems. Although misinterpreted by most people, SOA is not a synonym for web services. However, most SOA-based implementations are built with web services.

Building an enterprise SOA solution from scratch requires a lot of time, effort, and money. Therefore, reusing already developed, third-party components can be cost effective and saves time. Due to demand for such SOA components, software vendors have implemented the required SOA components and some vendors have managed to build entire SOA stacks. WSO2 is one such vendor. The diagram below shows how WSO2 products are used to build a full SOA stack. It will help you get an overall understanding of an SOA solution.



For more information on SOA testing read our white paper on

[SOA Solutions & Middleware Testing >>](http://wso2.com/whitepapers/soa-solutions-and-middleware-testing/)

or read our customer use case of SOA implementation in our case study on

[University Psychiatric Clinics Basel Modernizes Integration of its Applications with WSO2 Enterprise Service Bus >>](http://wso2.com/casestudies/university-psychiatric-clinics-basel-Modernizes-integration-of-its-applications-with-wso2-enterprise-service-bus/)

What is SOA testing?

Every software product is supposed to go through a quality assurance cycle to make sure the product that is delivered lives up to its quality standards. Therefore, testing is a major effort required in any software development project.

Most people consider web service testing as SOA testing. However, as mentioned before, SOA is not only about the web services. It is about the overall architecture. In other words, SOA testing should not be restricted to just web services testing.

An SOA solution is generally an integrated set of products that can be a collection of legacy applications, third-party components, or custom developed components, among others. Testers are expected to test not only the individual products and its functionality, but also the overall integrated solution. Therefore, an SOA tester has to consider the bigger picture of SOA for testing, including service providers, service directories, service consumers, communication between SOA components, and authentication providers.

Importance of SOA testing

Businesses today completely depend on their IT systems for their day-to-day business operations, and SOA can be considered as the backbone of the enterprise system. There is no doubt that smooth functioning of enterprise systems is really important for modern businesses and a single error can result in the enterprise losing large sums of money and damaging the business’ brand.

SOA solutions are really complex in nature and due to the challenges, which will be discussed later in this article, the testing phase of an SOA solution is crucial.

Challenges of SOA testing

Testing is undoubtedly a challenging task and can become increasingly challenging due to reasons like growing customer expectations, rapid changes in systems, limited time frames, etc.

The following points can be identified as some of the more unique challenges when in SOA testing:

* Complex and heterogeneous nature of SOA solutions
* SOA testing being more data driven than traditional testing
* The need to check product compliance against complex technical specs and standards
* The larger set of testing combinations with different types of products that can be integrated
* Difficulty to simulate testing environments
* Difficulty to reproduce issues
* The need to focus on complete business use cases instead of just a product feature
* High technical competency required from testers
* The need for more attention on load and performance testing
* Inclusion of different vendor products
* Limited time for testing
* Continuous growth of the platform with new features, new products, etc.

How SOA testing differs from traditional testing

When testing normal applications, testers highly depend and interact on the respective application GUIs. However, SOA testers do not have that luxury as most SOA components do not come with GUIs and normally expose programmatic interfaces like APIs and WSDLs.

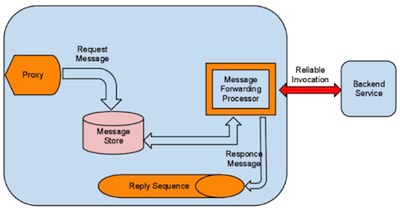
Normal web applications can be tested on general user perspective as, most times, real users of considered applications are non-technical. Yet, when it comes to SOA testing, end-users of most SOA solutions/components are highly technical people like solution architects, CIOs, and software developers. Therefore, testers need to test the applications from a developer point of view. SOA testers need to possess expertise in both the business and technology domains to effectively test it.

SOA testing requires both service consumers and service providers to be tested. The SOA platform mostly provides just the section for the service providers, so the testers need to implement matching SOA client apps in order to perform testing. As a result, the testers need to play the role of a developer as well.

Given rapid release cycles, less time is allocated for testing, so it is very important to automate all possible testing scenarios. GUI-based automation plays a major role when automating normal web applications; however, for an SOA solution, automation needs to be done through programmatic interfaces.

In traditional testing, the tester has to wait until the developers complete the features of the system; however, in SOA testing, they can start testing at earlier stages by creating similar mock services.

To highlight the exact differences between traditional testing and SOA testing, we have used the [Message Store and Message Processor JMS Use Case](https://docs.wso2.org/display/ESB481/Store+and+Forward+Using+JMS+Message+Stores#StoreandForwardUsingJMSMessageStores-UseCaseScenario2) scenario that is documented in the WSO2 ESB documentation section.



**Traditional approach**

The tester will publish the messages through the proxy service and checks if he/she receives the expected response at the Reply Sequence. The tester only executes simple test cases and generates the test results depending on them.

**SOA approach**

Here, the tester needs to test all the scenarios listed below.

* Make sure the Reply Sequence receives the respective response message accordingly
* Find out how the system behaves in case of Backend Service and Message Store down time
* Find out how the system behaves when there are high loads
* Check whether there is any message duplication, modification, or loss in the system
* Find out how the system behaves with a slow backend service
* Find out how the system behaves to the different types of message formats sent through the proxy service
* Test the individual Message Store and the Message Forwarding Processor feature

SOA testing strategies

As discussed above, the nature of SOA-based solutions call for different testing strategies and a different toolset.

SOA testers need to have a specific set of skills and a different mindset because they are required to identify a different approach.

There are a few identified approaches for SOA testing. They include bottom-up testing, top-down testing, and system testing. Testers can use any or all of these approaches. Most successful SOA testing projects use all approaches and place emphasis on one approach based on the requirements of the architecture.

SOA testers and their testing psychology

The following qualities are generally considered essential for a software tester. SOA testers should posses the same qualities. These include

* Attention to detail and keen observation
* Ability to detect possible problems quickly
* Passion and willingness to break the software systems
* Understand the product and integration of its parts
* Be able to look into issues from the perspective of customers/users
* Willingness to object to and question every aspect of the software
* Willingness to complete everything perfectly and in a well-ordered manner

SOA testers need to specifically possess certain unique qualities and approaches, such as as the following:

* Test the application with a developer-oriented mindset and not from the traditional user perspective
* Have a wider knowledge of all aspects in IT from OS administration and database administration to programming languages and networking
* Have fair knowledge in software development
* Have a thorough understanding of SOA paradigms

SOA testing lifecycle

In the traditional software testing lifecycle, the following phases are executed and each phase has its own entry criteria and deliverables. These processes may vary according to the software development lifecycle that is being used as well as management decisions.

1. Requirements/design review
2. Test planning
3. Test designing
4. Test environment setup
5. Test execution
6. Test reporting

When performing SOA testing, a similar process can be followed, but testers need to put more effort into configuring required test environments due to the complex nature of SOA solutions.

During the test execution phase, SOA testers need to put in more effort into and focus more attention on the areas mentioned in the sections below.

**Performance**

Any middleware platform should perform swiftly to support the business transactions. Therefore, identifying performance bottlenecks is a key aspect that an SOA tester would need to focus on.

**Load**

With the rapid growth of technology, enterprise applications require the capacity to handle more data and more transactions. As a result, it is very important to consider load testing.

**QoS**

Each SOA service has a Quality of Service (QoS) associated with it. Some of the key QoS elements are security (i.e. authentication and authorization), reliable messaging, caching, and throttling. By using different policies, who can access what, when and how often can be restricted. The testers will have to test for all these factors.

Furthermore, SOA testers need to make sure that the SOA solution and its components work according to the standard specification. For example, with regard to security, components should follow standards such as Security Assertion Markup Language (SAML) and WS-Security.

**Reliability**

SOA has a loosely coupled nature and several messages usually need to be exchanged between the service consumers and the service providers to accomplish a business activity. As a result, it is extremely important to ensure that the messages flow between the systems as expected. SOA testers need to make sure that the system behaves in a reliable manner.

**Clustering**

Due to the increasing transactions and data handling loads, a single SOA component may not be able to handle the expected loads. Hence, those components are required to run with multiple nodes to support the required scalability. In addition, for high availability and failover scenarios, running an SOA solution with multiple nodes is important. Therefore, testers have to make sure that the required components run smoothly in clustered environments.

**Integration**

As we highlighted in previous sections of the article, a key advantage of SOA is its ability to integrate different kinds of products. The tester will have to focus on possible integration scenarios and make sure that the SOA solution integrates accurately.

**Platform support**

SOA platform components are supposed to run on different operating systems. Ensuring that the products are running on all major operating systems is also a key part of the SOA testing cycle.

SOA testing tools

Having a proper tool set is very important for SOA testing. Effective and relevant tools will maximize productivity and produce accurate outcomes with the expected quality.

When selecting a testing tool for SOA testing the following factors should be considered:

* The ability to generate request messages automatically
* The ability to validate responses using assertions
* QoS-enabled service invocation
* Service simulation
* Support for multiple transport protocols
* Multiple message support

**Testing tools**

**SoapUI -** SoapUI is an open-source tool that can be used for web services testing. In addition to functional testing, SoapUI can also be used for performance testing. One of the advantages of using SoapUI is that you can use it to perform security testing.

**Apache Jmeter -** JMeter is an open-source tool that can be used for analyzing and measuring performance and performing REST/SOAP invocations. This tool can be used to automate most user stories. It allows the tester to run the same test with different users and also has the ability to perform parameterization. One of the drawbacks of this tool is that it cannot be used to automate security-related scenarios.

**cURL -**cURL is a command line tool that is used to send or retrieve information with the use of URL syntax. This tool supports many internet protocols such as HTTP, HTTPS, FTP, IMAP, and many more. cURL is not heavily used for load testing and is mainly used for functional testing.

Refer to [WSO2 Library article about SOA tools](http://wso2.com/library/articles/2013/10/productivity-acceleration-tools-for-soa-testers/#chap2) to learn more about SOA testing tools.

**Monitoring tools**

**JConsole -** JConsole is a graphical monitoring tool that is used to monitor Java Virtual Machine (JVM) and Java applications, both on local and on remote machines.

**JProfiler -** The JProfiler can be used to identify performance bottlenecks, pin down memory leaks, and understand threading issues.

Traditional tools that are used with the testing life cycle can also be used for SOA testing. SOA testers can use tools like Microsoft Test Manager and TestLink for test case management. Other tools like Atlassian JIRA, Bugzilla, Mantis, and Redmine can be used for issue tracking and issue management.

Test automation of SOA

Maintaining the quality of the SOA platform is a tedious job. Increasing the manual test team is not practical or feasible in the long run. Therefore, automating the SOA as much as possible is the only solution.

However, SOA automation is quite different from normal web application automation. In web automation, testers depend heavily on user interfaces; however, in SOA automation, testers have to focus on programmatic interfaces and services.

The following areas can be identified as key areas that require consideration for automation.

* Test environment setup/clustering
* Unit/module testing
* Product level feature automation
* Integration scenario automation

Summary

Having an idea about SOA and its uniqueness will significantly help to effectively carry out SOA testing. Throughout this article, we have explained the nature of an SOA solution and how it differs from a traditional software solution. With that understanding, it’s also key to identify the unique challenges in SOA testing and the required approaches to overcome these challenges.

**What is service-oriented architecture?**

**MORE LIKE THIS**

* [**The power behind the SOA repository**](http://www.javaworld.com/article/2072287/soa/the-power-behind-the-soa-repository.html)
* [**Get familiar with ebXML Registry**](http://www.javaworld.com/article/2071871/soa/get-familiar-with-ebxml-registry.html)
* [**SOA for the real world**](http://www.javaworld.com/article/2076191/data-storage/soa-for-the-real-world.html)

**An introduction to SOA**

**By Raghu R. Kodali**

JavaWorld | Jun 13, 2005 2:00 AM PT

RELATED TOPICS

* [Cloud Computing](http://www.javaworld.com/category/cloud-computing)
* [Learn Java](http://www.javaworld.com/category/learn-java)
* [Enterprise Java](http://www.javaworld.com/category/enterprise-java)
* [Development Tools](http://www.javaworld.com/category/development-tools)
* [Java App Dev](http://www.javaworld.com/category/application-development)
* [Web Services](http://www.javaworld.com/category/web-services)
* [Java Language](http://www.javaworld.com/category/java-language)

[24COMMENTS](http://www.javaworld.com/article/2071889/soa/what-is-service-oriented-architecture.html#comments)

*Get a primer on SOA, starting with the basic question of what a service-oriented architecture is and what comprises a Java-based SOA infrastructure at the core, platform, and quality-of-services level.*

Service-oriented architecture (SOA) is an evolution of distributed computing based on the request/reply design paradigm for synchronous and asynchronous applications. An application's business logic or individual functions are modularized and presented as services for consumer/client applications. What's key to these services is their loosely coupled nature; i.e., the service interface is independent of the implementation. Application developers or system integrators can build applications by composing one or more services without knowing the services' underlying implementations. For example, a service can be implemented either in .Net or J2EE, and the application consuming the service can be on a different platform or language.

[The Six Roles of the Interface](http://www.javaworld.com/article/3044050/java-language/discover-the-six-roles-that-interfaces-play-in-the-java-language.html)

Java's interface language feature often puzzles newcomers to this language. I eliminate this mystery by

[READ NOW](http://www.javaworld.com/article/3044050/java-language/discover-the-six-roles-that-interfaces-play-in-the-java-language.html)

Service-oriented architectures have the following key characteristics:

* SOA services have self-describing interfaces in platform-independent XML documents. Web Services Description Language ([WSDL](http://www.javaworld.com/#resources)) is the standard used to describe the services.
* SOA services communicate with messages formally defined via XML Schema (also called [XSD](http://www.javaworld.com/#resources)). Communication among consumers and providers or services typically happens in heterogeneous environments, with little or no knowledge about the provider. Messages between services can be viewed as key business documents processed in an enterprise.
* SOA services are maintained in the enterprise by a registry that acts as a directory listing. Applications can look up the services in the registry and invoke the service. Universal Description, Definition, and Integration ([UDDI](http://www.javaworld.com/#resources)) is the standard used for service registry.
* Each SOA service has a quality of service (QoS) associated with it. Some of the key QoS elements are security requirements, such as authentication and authorization, reliable messaging, and policies regarding who can invoke services.

**Why SOA?**

The reality in IT enterprises is that infrastructure is heterogeneous across operating systems, applications, system software, and application infrastructure. Some existing applications are used to run current business processes, so starting from scratch to build new infrastructure isn't an option. Enterprises should quickly respond to business changes with agility; leverage existing investments in applications and application infrastructure to address newer business requirements; support new channels of interactions with customers, partners, and suppliers; and feature an architecture that supports organic business. SOA with its loosely coupled nature allows enterprises to plug in new services or upgrade existing services in a granular fashion to address the new business requirements, provides the option to make the services consumable across different channels, and exposes the existing enterprise and legacy applications as services, thereby safeguarding existing IT infrastructure investments.

**SOA for cloud computing**

Learn more about service-oriented architecture in the Java enterprise:

* [Logically SOA: An architecture overview](http://www.javaworld.com/javaworld/jw-01-2007/jw-01-soa.html)
* [SOA for the real world](http://www.javaworld.com/javaworld/jw-11-2006/jw-1129-soa.html)
* [Secure SOA: What you need to know](http://www.javaworld.com/javaworld/jw-04-2006/jw-0410-webservices.html)
* [Enterprise patterns for a lean SOA](http://www.javaworld.com/javaworld/jw-04-2009/jw-04-lean-soa-with-javaee6.html)

As in Figure 1's example, an enterprise employing SOA could create a supply chain composite application using a set of existing applications that expose the functionality via standard interfaces.

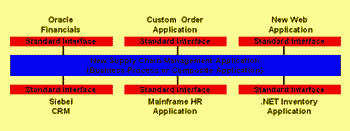
[[](http://www.javaworld.com/carticimagjw-0613-soa1.gif)](http://www.javaworld.com/carticimagjw-0613-soa1.gif" \o "Figure 1. Supply chain application. Click on thumbnail to view full-sized image.)

Figure 1. Supply chain application. Click on thumbnail to view full-sized image.

**Service architecture**

To implement SOA, enterprises need a service architecture, an example of which is shown in Figure 2.

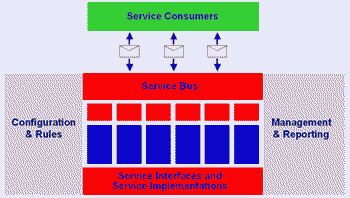
[[](http://www.javaworld.com/carticimagjw-0613-soa2.gif)](http://www.javaworld.com/carticimagjw-0613-soa2.gif" \o "Figure 2. A sample service architecture. Click on thumbnail to view full-sized image.)

Figure 2. A sample service architecture. Click on thumbnail to view full-sized image.

In Figure 2, several service consumers can invoke services by sending messages. These messages are typically transformed and routed by a service bus to an appropriate service implementation. This service architecture can provide a business rules engine that allows business rules to be incorporated in a service or across services. The service architecture also provides a service management infrastructure that manages services and activities like auditing, billing, and logging. In addition, the architecture offers enterprises the flexibility of having agile business processes, better addresses the regulatory requirements like Sarbanes Oxley ([SOX](http://www.javaworld.com/#resources)), and changes individual services without affecting other services.

**RECENT JAVA HOW-TOs**

* [](http://www.javaworld.com/article/3030214/big-data/open-source-java-projects-apache-phoenix.html)

[**Open source Java projects: Apache Phoenix**](http://www.javaworld.com/article/3030214/big-data/open-source-java-projects-apache-phoenix.html)

* [](http://www.javaworld.com/article/3008117/development-tools/jump-into-java-microframeworks-part-2-ninja.html)

[**Jump into Java microframeworks, Part 2: Ninja**](http://www.javaworld.com/article/3008117/development-tools/jump-into-java-microframeworks-part-2-ninja.html)

* [](http://www.javaworld.com/article/2995526/development-tools/jump-into-java-micro-frameworks-part-1.html)

[**Jump into Java microframeworks, Part 1**](http://www.javaworld.com/article/2995526/development-tools/jump-into-java-micro-frameworks-part-1.html)

**SOA infrastructure**

To run and manage SOA applications, enterprises need an SOA infrastructure that is part of the SOA platform. An SOA infrastructure must support all the relevant standards and required runtime containers. A typical SOA infrastructure looks like Figure 3. The following sections discuss the infrastructure's individual pieces.

[[](http://www.javaworld.com/native?prx_t=EQsCA5KAFAqikMA&ntv_fr)](http://www.javaworld.com/native?prx_t=EQsCA5KAFAqikMA&ntv_fr" \o "Unified Communications: the Backbone of the Intelligent Enterprise)

***BrandPost*** Sponsored by Crestron

[Unified Communications: the Backbone of the Intelligent Enterprise](http://www.javaworld.com/native?prx_t=EQsCA5KAFAqikMA&ntv_fr)

Making it easy to optimize workflow, collaboration, and information Unified Communications in an intelligent, high-performance enterprise not only integrates voice, chat, email, and video, but also includes video...

[[](http://www.javaworld.com/carticimagjw-0613-soa3.gif)](http://www.javaworld.com/carticimagjw-0613-soa3.gif" \o "Figure 3. A typical SOA infrastructure. Click on thumbnail to view full-sized image.)

Figure 3. A typical SOA infrastructure. Click on thumbnail to view full-sized image.

**SOAP, WSDL, UDDI**

WSDL, UDDI, and SOAP are the fundamental pieces of the SOA infrastructure. WSDL is used to describe the service; UDDI, to register and look up the services; and SOAP, as a transport layer to send messages between service consumer and service provider. While SOAP is the default mechanism for Web services, alternative technologies accomplish other types of bindings for a service. A consumer can search for a service in the UDDI registry, get the WSDL for the service that has the description, and invoke the service using SOAP.

**POPULAR ON JAVAWORLD**

* [](http://www.javaworld.com/article/2900842/mobile-java/java-the-once-and-future-king-of-internet-programming.html)

[**Java: The once and future king of Internet programming**](http://www.javaworld.com/article/2900842/mobile-java/java-the-once-and-future-king-of-internet-programming.html)

* [](http://www.javaworld.com/article/2916548/java-web-development/http-2-for-java-developers.html)

[**HTTP/2: A jump-start for Java developers**](http://www.javaworld.com/article/2916548/java-web-development/http-2-for-java-developers.html)

* [](http://www.javaworld.com/article/2076075/learn-java/core-java-learn-java-from-the-ground-up.html)

[**Learn Java from the ground up**](http://www.javaworld.com/article/2076075/learn-java/core-java-learn-java-from-the-ground-up.html)

**WS-I Basic Profile**

[WS-I Basic Profile](http://www.javaworld.com/#resources), provided by the Web services Interoperability Organization, is turning into another core piece required for service testing and interoperability. Service providers can use the Basic Profile test suites to test a service's interoperability across different platforms and technologies.

**J2EE and .Net**

Though the J2EE and .Net platforms are the dominant development platforms for SOA applications, SOA is not by any means limited to these platforms. Platforms such as J2EE not only provide the framework for developers to naturally participate in the SOA, but also, by their inherent nature, bring a mature and proven infrastructure for scalability, reliability, availability, and performance to the SOA world. Newer specifications such as Java API for XML Binding (JAXB), used for mapping XML documents to Java classes, Java API for XML Registry (JAXR), used for interacting with the UDDI registries in a standard manner, and Java API for XML-based Remote Procedure Call (XML-RPC), used for invoking remote services in J2EE 1.4 facilitate the development and deployment of Web services that are portable across standard J2EE containers, while simultaneously interoperating with services across other platforms such as .Net.

**Quality of services**

Existing mission-critical systems in enterprises address advanced requirements such as security, reliability, and transactions. As enterprises start adopting service architecture as a vehicle for developing and deploying applications, basic Web services specifications like WSDL, SOAP, and UDDI aren't going to fulfill these advanced requirements. As mentioned previously, these requirements are also known as quality of services. Numerous specifications related to QoS are being worked out in standards bodies like the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS). Sections below discuss some of the QoS artifacts and related standards.

***Security***

The [Web Services Security](http://www.javaworld.com/#resources) specification addresses message security. This specification focuses on credential exchange, message integrity, and message confidentiality. The attractive thing about this specification is it leverages existing security standards, such as Security Assertion Markup Language (SAML), and allows the usage of these standards to secure Web services messages. Web Services Security is an ongoing OASIS effort.

***Reliability***

In a typical SOA environment, several documents are exchanged between service consumers and service providers. Delivery of messages with characteristics like once-and-only-once delivery, at-most-once delivery, duplicate message elimination, guaranteed message delivery, and acknowledgment become important in mission-critical systems using service architecture. [WS-Reliability](http://www.javaworld.com/#resources) and [WS-ReliableMessaging](http://www.javaworld.com/#resources)are two standards that address the issues of reliable messaging. Both these standards are now part of OASIS.

***Policy***

Service providers sometimes require service consumers to communicate with certain policies. As an example, a service provider may require a Kerberos security token for accessing the service. These requirements are defined as *policy assertions.* A policy may consist of multiple assertions. WS-Policy standardizes how policies are to be communicated between service consumers and service providers.

***Orchestration***

As enterprises embark on service architecture, services can be used to integrate silos of data, applications, and components. Integrating applications means that the process requirements, such as asynchronous communication, parallel processing, data transformation, and compensation, must be standardized. BPEL4WS or [WSBPEL](http://www.javaworld.com/#resources) (Web Services Business Process Execution Language) is an OASIS specification that addresses service orchestration, where business processes are created using a set of discrete services. WSBPEL is now part of OASIS.

***Management***

As the number of services and business processes exposed as services grow in the enterprise, a management infrastructure that lets the system administrators manage the services running in a heterogeneous environment becomes important. Web Services for Distributed Management ([WSDM](http://www.javaworld.com/#resources)) will specify that any service implemented according to WSDM will be manageable by a WSDM-compliant management solution.

Other QoS attributes such as coordination between partners and transactions involving multiple services are being addressed in the WS-Coordination and WS-Transaction specifications, respectively, which are OASIS efforts as well.

**SOA is not Web services**

There seems to be general confusion about the relationship between SOA and Web services. In an April 2003 Gartner report, Yefim V. Natis makes the distinction as follows: "Web services are about technology specifications, whereas SOA is a software design principle. Notably, Web services' WSDL is an SOA-suitable interface definition standard: this is where Web services and SOA fundamentally connect." Fundamentally, SOA is an architectural pattern, while Web services are services implemented using a set of standards; Web services is one of the ways you can implement SOA. The benefit of implementing SOA with Web services is that you achieve a platform-neutral approach to accessing services and better interoperability as more and more vendors support more and more Web services specifications.

**Benefits of SOA**

While the SOA concept is fundamentally not new, SOA differs from existing distributed technologies in that most vendors accept it and have an application or platform suite that enables SOA. SOA, with a ubiquitous set of standards, brings better reusability of existing assets or investments in the enterprise and lets you create applications that can be built on top of new and existing applications. SOA enables changes to applications while keeping clients or service consumers isolated from evolutionary changes that happen in the service implementation. SOA enables upgrading individual services or services consumers; it is not necessary to completely rewrite an application or keep an existing system that no longer addresses the new business requirements. Finally, SOA provides enterprises better flexibility in building applications and business processes in an agile manner by leveraging existing application infrastructure to compose new services.

Raghu R. Kodali is consulting product manager and SOA evangelist for Oracle Application Server. Kodali leads next-generation SOA initiatives and J2EE feature sets for Oracle Application Server, with particular expertise in EJB, J2EE deployment, Web services, and BPEL. Prior to product management, Kodali held presales and technical marketing positions in Oracle Asia-Pacific, based in Singapore. Prior to Oracle, he worked as software developer with National Computer Systems, Singapore. He holds a master's degree in computer science and is a frequent speaker at technology conferences. Kodali maintains an active blog at Loosely Coupled Corner

service-oriented architecture (SOA)

Service-oriented architecture (SOA) is an approach used to create an [architecture](http://whatis.techtarget.com/definition/architecture)based upon the use of services. Services (such as RESTful Web services) carry out some small function, such as producing data, validating a customer, or providing simple analytical services.

Download Our Guide: Application Integration for SaaS Adoption



More and more organizations looking to hop on the cloud wagon are facing application and data integration roadblocks. Getting around them is difficult but possible—and a step-by-step approach can help.

窗体顶端

* 
* 

窗体底端

By submitting your email address, you agree to receive emails regarding relevant topic offers from TechTarget and its[partners](http://www.techtarget.com/html/privacy_partners_0-c.html). You can withdraw your consent at any time. Contact [TechTarget](http://www.techtarget.com/html/about_contact_directions.html) at 275 Grove Street, Newton, MA.

You also agree that your personal information may be transferred and processed in the United States, and that you have read and agree to the [Terms of Use](http://searchsoa.techtarget.com/about/copyright) and the [Privacy Policy](http://www.techtarget.com/html/privacy_policy.html).

Safe Harbor

In addition to building and exposing services, SOA has the ability to leverage these services over and over again within [applications](http://searchsoftwarequality.techtarget.com/definition/application) (known as composite applications). SOA binds these services to [orchestration](http://searchsoa.techtarget.com/definition/service-orchestration), or individually leverages these services. Thus, SOA is really about fixing existing architectures by addressing most of the major systems as services, and abstracting those services into a single [domain](http://searchsoa.techtarget.com/definition/domain) where they are formed into solutions.

One of the keys to SOA architecture is that interactions occur with [loosely coupled](http://searchnetworking.techtarget.com/definition/loose-coupling) services that operate independently. SOA architecture allows for service reuse, making it unnecessary to start from scratch when upgrades and other modifications are needed. This is a benefit to businesses that seek ways to save time and money.

SOA is known to provide both time-to-market advantages, as well as [business agility](http://searchcio.techtarget.com/definition/business-agility-BA). The use of orchestration engines, or leveraging development environments that leverage services and SOA, allow those who build applications to do so quickly, since the services provide much of what the application requires. This provides the time-to-market advantage.

PRO+

Content

Find more PRO+ content and other member only offers,[here.](http://pro.techtarget.com/ProLP?Offer=PROContentBox)

* **E-Handbook**

[Mobile goes back to the future with SOA principles](http://searchsoa.techtarget.com/ehandbook/Mobile-goes-back-to-the-future-with-SOA-principles)

* **E-Handbook**

[Embedded software, IoT development demand careful scrutiny](http://searchsoa.techtarget.com/ehandbook/Embedded-software-IoT-development-demand-careful-scrutiny)

Placing volatility into a domain (such as an orchestration engine) allows SOA-built applications to quickly adapt around changing business requirements. In many instances, it's just a matter of re-sequencing the services invoked, or reconfiguring the orchestrations to alter the application.

Simple in concept, SOA is also a best practice to fix broken architectures. With the wide use of standards such as [Web services](http://searchsoa.techtarget.com/definition/Web-Services-Glossary), SOA is being promoted as the best way to bring architectural agility to your enterprise, that is, if you do SOA correctly. The problem has been that the ways that enterprises leverage SOA as an architectural pattern varies greatly from enterprise-to-enterprise. Thus, the [ROI](http://searchcio.techtarget.com/definition/ROI) from moving to SOA has ranged from great successes, to outright failures.

SOA is a valid approach to solve many of the architectural problems that enterprises face today. However, those who implement SOA typically look at it as something you buy, not something you do. Thus, many SOA projects are about purchasing some technology that is sold as 'SOA-in-a-box.' You get something-in-a-box, but not SOA, and that only adds to the problems.

SOA, as the "A" implies, is architecture. And thus it is the orderly arrangement of systems that best serve the needs of the business. Taken in its literal context, [enterprise IT](http://searchcio.techtarget.com/definition/enterprise-IT-enterprise-class-IT) can succeed with SOA. However, most do not succeed and much of that failure is due to the fact that the SOA implementers view SOA as something other than architecture, and most often those implementers are not architects.

While SOA enjoyed varying success in the past, the movement to [cloud computing](http://searchcloudcomputing.techtarget.com/definition/cloud-computing)provides some renewed value to SOA. [Clouds](http://searchnetworking.techtarget.com/definition/cloud) are typically [API](http://searchexchange.techtarget.com/definition/application-program-interface)- or service-driven, and thus are service-oriented. As cloud computing becomes more popular, more enterprises will rethink the use of SOA, which includes the use of service directories, [service governance](http://searchsoa.techtarget.com/definition/SOA-governance), orchestration, and other technologies related to SOA.

**To explore how SOA is used in the enterprise, here are some additional resources:**

[The principles of service orientation:](http://searchsoa.techtarget.com/feature/SOA-Advisor-The-principles-of-service-orientation) SOA guru Thomas Erl explains the fundamentals of service-oriented architecture, including loose coupling, service abstraction and statelessness.

[How do SOAP and REST stack up as Web services?](http://searchsoa.techtarget.com/tip/REST-vs-SOAP-How-to-choose-the-best-Web-service) REST and SOAP each have their own benefits. Learn when it makes to use one over the other.

[Get back to the basics of service-oriented architecture with this SOA Overview:](http://searchsoa.techtarget.com/feature/SOA-Overview) Learn about implementation, registry and repository, governance and management of SOA.

* http://cdn.ttgtmedia.com/ITKE/uploads/avatar/21703.gif

Margaret Rouse asks:

**Does your organization use SOAP for Web services, or an alternative?**

4  Responses

[Join the Discussion](http://itknowledgeexchange.techtarget.com/discussions/discussion/does-your-organization-use-soap-for-web-services-or-an-alternative/)

[Learn about the different components of service oriented architecture:](http://searchdatamanagement.techtarget.com/answer/What-are-the-components-of-service-oriented-architecture-SOA) Take a look at a short list of they key pieces of SOA architecture.

[A guide to using SOAP for Web services:](http://searchsoa.techtarget.com/tutorial/Simple-Object-Access-Protocol-SOAP-Tutorial) Learn about the SOAP standard and when it's best to use it.

This was first published in December 2014

Next Steps

Enterprises can manage applications [using Talend ESB](http://searchsoa.techtarget.com/feature/For-data-oriented-app-integration-look-toward-Talend-ESB) and its open source version, Open Studio for ESB.

Continue Reading About serv

SOA Testing Tools for Black, White and Gray Box

Black, White & Gray Box SOA Testing Tools are essential for deploying robust, scalable, interoperable and secure Web Services.

I. Introduction

|  |  |  |
| --- | --- | --- |
| Web Services are the foundations of modern Service Oriented Architecture (SOA).  Typical Web Services include message exchange between a consumer and a producer using SOAP request and responses over the ubiquitous HTTP protocol. A Web service producer advertises its services to potential consumers through Web Services Description Language (WSDL) – an XML file that contains details of available operations, execution endpoints and expected SOAP request-response structures.    Many testing techniques and methodologies developed over the years apply to Web Services-based SOA systems as well. Through functional, regression, unit, integration, system and process level testing, the primary objective of testing methodologies is to increase confidence that the target system will deliver functionality in a robust, scalable, interoperable and secure manner.    Techniques such as Black, White and Gray Box testing applied to traditional systems map well into Web Services deployments.  However, the following characteristics of a Web Services deployments introduce unique testing challenges:    §          Web Services are intrinsically distributed and are platform and language agnostic.  §          Web Services can be chained with dependencies on other 3rd party Web Services that can change without notice.  §          Web Services ownership is shared across various stakeholders.  §          Web Services client developers typically only have access to interfaces (WSDLs) and lack access to code.    In this paper, we will investigate testing techniques and their application to Web Services.  We will use a simple sample Web service to illustrate each of these techniques and the relative strengths and weaknesses of such techniques.  Finally, a novel approach that extends Gray Box’s reach into realm of White Box testing by leveraging the rich information provided in the WSDL file will be described. |  | Key Concepts    Essential Testing Techniques  Using Black, White, & Gray Box Testing techniques are key to deploying robust SOA.      Gray Box Testing is ideal for Web Services Consumers  Published Web Services operations in WSDLs provide rich enough information for consumers to derive benefits of Gray Box Testing.      Extending Gray Box’s Reach into the Domain of  White Box Testing  Using Web Services WSDLs, intelligent auto-generated tests can elevate the Gray Box Testing closer to White Box Testing without the associated burdens.      Auto-Generated Tests  Mutations based on Web Services WSDLs enable users to auto-generate tests that exercise multiple code-paths extensively in published operations. |

II. Black, White and Gray Box Testing for Web Services

A.  Black Box Testing

Definition:  Black Box testing refers to the technique of testing a system *with no knowledge* of the internals of the system.  Black Box testers do not have access to the source code and are oblivious of the system architecture.  A Black Box tester typically interacts with a system through a user interface by providing inputs and examining outputs without knowing where and how the inputs were operated upon. In Black-Box testing, target software is exercised over a range of inputs and the outputs are observed for correctness. How those outputs are generated or what is inside the Box doesn't matter to the tester.

Web Service Example:  To illustrate Black Box testing for a sample Web Services, an operation **Divide** that simply divides to integers *a*and *b*.  The Black Box tester is unaware of what operating system, programming language, 3rd party libraries or other Web Services are being used under the hood to perform the**Divide** operation.

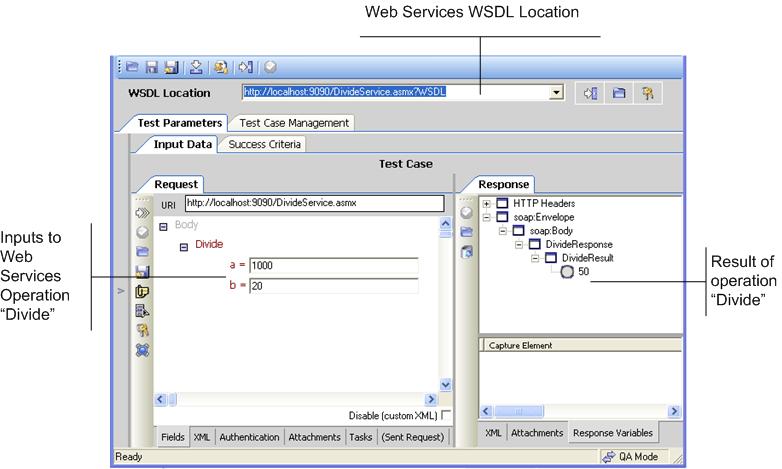


Figure 1:  Black Box Testing Example for Divide Operation.

As shown in **Figure 1**, the Black Box tester has the ability to insert inputs to the operation and look at outputs. The tester may know that the Web Services WSDL is located but is completely oblivious of implementation details, program execution states, and internal exception handling.  The tester has a specification and goes through a rigorous, time-consuming and oftentimes redundant exercise of trying values and ensuring that the operation **Divide** functions as expected.

Advantages

§          Efficient Testing – Well suited and efficient for large code segments or units.

§          Unbiased Testing – clearly separates user’s perspective from developer’s perspective through separation of QA and Development responsibilities.

§          Non intrusive – code access not required.

§          Easy to execute – can be scaled to large number of moderately skilled testers with no knowledge of implementation, programming language, operating systems or networks.

Disadvantages

§          Localized Testing – Limited code path coverage since only a limited number of test inputs are actually tested.

§          Inefficient Test Authoring – without implementation information, exhaustive input coverage has unknown additional benefits to the actual code paths exercised and can require tremendous resources.

§          Blind Coverage – cannot control targeting code segments or paths which may be more error prone than others.

Black Box testing is best suited for rapid test scenario testing and quick Web Service prototyping.  This testing technique for Web Services provides quick feedback on the functional readiness of operations through quick spot checking.  Black Box testing is also better suited for operations that have enumerated inputs or tightly defined ranges or facets so that broad input coverage is not necessary.

B.  White Box Testing

Definition:  White Box testing refers to the technique of testing a system *with knowledge* of the internals of the system.  White Box testers have access to the source code and are aware of the system architecture.  A White Box tester typically analyzes source code, derives test cases from knowledge about the source code, and finally targets specific code paths to achieve a certain level of code coverage.

Web Service Example:  To illustrate White Box testing, Figure 2 presents a simple set of rudimentary operations in C#.  The first Operation **Divide** takes in 2 integers and returns *a* divide-by *b* with  no checks on the input values.  The second operation **safeDivide**takes in 2 string parameter inputs, but in contrast to the first operation **Divide**, the **safeDivide** operation has a broad exception handling mechanism that safely catches errors for cases where either bad data types of bad data values are sent.

A White Box tester with access to such details about both operations can readily craft efficient test cases that exercise boundary conditions.  Just by observing the code, a White Box tester can immediately try:

§          Divide-by-Zero scenario by setting the denominator *b* to zero

§          Integer Overflow scenario by setting either integer a value > ± 2,147,483,647

§          Orthogonal Data types for example floats, date, decimal data types.

§          Special characters.

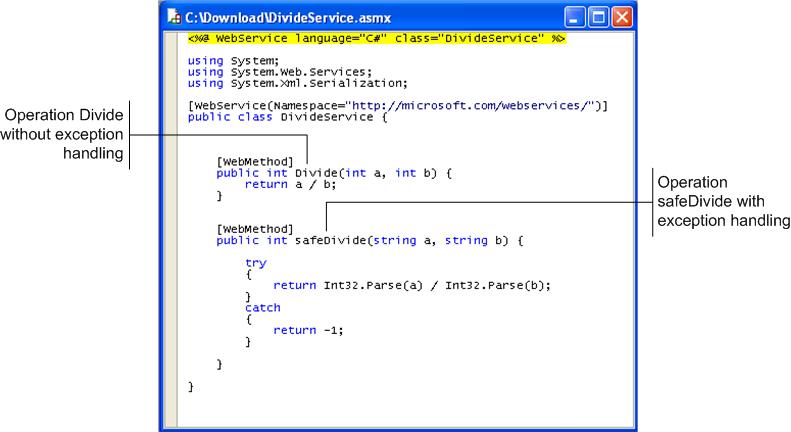


Figure 2:  White Box Testing Example for 2 Web Services Operations.

A simple Divide-by-Zero test, executed for the **safeDivide** operation returns a rudimentary error code -1 as expected.  However, performing the same Divide-by-Zero test on the **Divide** operation results in the following verbose stack trace in the SOAP Response:

1.       <soap:Fault>

2.            <faultcode>soap:Server</faultcode>

3.             <faultstring>System.Web.Services.Protocols.SoapException: Server was unable to process request. --- System.DivideByZeroException: Attempted to divide by zero. atDivideService.Divide(Int32 a, Int32 b)End of inner exception stack trace ---

4.             </faultstring>

5.       </soap:Fault>

In the **Divide** operation, a White Box tester can quickly identify and subsequently verify that the target operation has no custom exception handling but rather relies on the vendor container running the Web Service to handle the exception. The tester would then point this lack of exception handling to the development teams and bring this operation inline with the **safeDivide** operation where*try-catch* flow is used for exception handling.  A Black Box tester may perhaps identify the weakness through blind testing, however, the level of effort and the number of iterations would be large and the probability of stumbling on to such defects would be low, especially with increasing program complexity.

Advantages

§          Increased Effectiveness – Crosschecking design decisions and assumptions against source code may outline a robust design, but the implementation may not align with the design intent.

§          Full Code Pathway Capable – all the possible code pathways can be tested including error handling, resource dependencies, and additional internal code logic/flow.

§          Early Defect Identification – Analyzing source code and developing tests based on the implementation details enables testers to find programming errors quickly.

§          Reveal Hidden Code Flaws – access to source code improves understanding and uncovering unintended hidden behavior of program modules.

Disadvantages

§          Difficult To Scale – requires intimate knowledge of target system, testing tools and coding languages, and modeling.  It suffers for scalability of skilled and expert testers.

§          Difficult to Maintain – requires specialized tools such as source code analyzers, debuggers, and fault injectors.

§          Cultural Stress – the demarcation between developer and testers starts to blur which may become a cultural stress.

§          Highly intrusive – requires code modification has been done using interactive debuggers, or by actually changing the source code. This may be adequate for small programs; however, it does not scale well to larger applications.  Not useful for networked or distributed systems.

White-Box testing is most suited for Web Services early in the development cycle where the developer and the tester may collaborate to identify defects.  White-Box testing is problematic for large SOA deployments where the distributed nature of services makes it easy for 3rd party Web Services to be invoked from within other Web Services.  This results in the lack of knowledge of programming language, operating systems and hardware platforms.  Unlike calling functions from a shared library running in the same memory space, distributed Web Services provide additional access challenges making White-Box testing across a SOA next to impossible.

C.  Gray Box Testing

Definition:  Gray Box testing refers to the technique of testing a system with *limited* *knowledge* of the internals of the system.  Gray Box testers have access to detailed design documents with information beyond requirement documents.  Gray Box tests are generated based on information such as state-based models or architecture diagrams of the target system.

Web Service Example:  To illustrate Gray Box testing, **Figure 3** presents a Web Services Definition Language (WSDL) file for the simple **Divide** and **safeDivide** operations.  Lines 3-36 show the data types for the messages.  These lines as well as Lines 23-24 and Lines 43-45 point to **safeDivide** request message using unbounded strings.  Even without access to source code or binaries, this would indicate to a tester to try buffer overflow type boundary conditions.

Without access to source code or binaries, a web service tester can only consume and invoke Web Services through WSDL files.  With a rich array of information available through such WSDLs, and the inability to modify code or binaries for White Box testing, a Web Services tester can use details such as the location of the web service and the transport protocol (Line 82), data types (Lines 3-36), etc. provide significant leverage for authoring intelligent, efficient and highly targeted test cases.



Figure 3: WSDL file (partially collapsed) for**Divide** and**safeDivide** operations.

Advantages

§          Offers Combined Benefits – Leverage strengths of both Black Box and White Box testing wherever possible.

§          Non Intrusive – Gray Box does not rely on access to source code or binaries.  Instead, based on interface definition, functional specifications, and application architecture.

§          Intelligent Test Authoring – Based on the limited information available, a Gray Box tester can author intelligent test scenarios, especially around data type handling, communication protocols and exception handling.

§          Unbiased Testing – The demarcation between testers and developer is still maintained.  The handoff is only around interface definitions and documentation without access to source code or binaries.

Disadvantages

§          Partial Code Coverage – Since the source code or binaries are not available, the ability to traverse code paths is still limited by the tests deduced through available information.  The coverage depends on the tester authoring skills.

§          Defect Identification – Inherent to distributed application is the difficult associated in defect identification.   Gray Box testing is still at the mercy of how well systems throw exceptions and how well are these exceptions propagated with a distributed Web Services environment.

The inherent distributed nature of Web Services and lack of source code or program binaries access makes White Box testing impossible within a SOA.  With WSDLs as the de facto contract between consumers and producers in a Web Services-based SOA, significant information is available to construct intelligent and efficient gray Box tests.  WSDLs provide rich information to construct and automate such tests to improve Web Services deployments.

III. Pushing Gray towards White Box Testing

Through WSDLs – the Web Services API –  testers have significant insight into the protocol, data types, operation expectations and error handling capabilities of a Web Services.  Intelligent and efficient Gray Box tests can be authored and run to determine defects based on the information available in WSDLs. However, there exists a strong need to automate the test generation process based on the available information for increased testing efficiency.

XSD-Mutation™ is one such patent-pending automation technique by Crosscheck Networks’ SOAPSonar™Enterprise product.  Using information available in the WSDL, a set of test cases both positive and negative can be generated to discover defects in target Web Services.  The test mutations may occur at the data type, data value, message structure or protocol binding level.  Although the WSDL does not reveal internal programmatic information such as relative exception handling capabilities of the two operations, through mutation generated tests; the application code exception handling logic and robustness is quickly revealed.

By using such techniques, Web Services can be thoroughly exercised without source code or binary access. Along the Black-White testing spectrum, such testing techniques push the middle ground gray Box testing more towards the White Box testing end of the spectrum without the associated expense or intrusiveness. Furthermore, given that White Box testing is not even an option in distributed Web Services-based SOAs, the only option available is to start with Gray Box testing and use automation tools such as SOAPSonar™ to push the gray towards White Box testing.  Mutation techniques add newer test “frequencies” to the testing spectrum driving the test cases closer towards a complete White Box set of “frequencies.”

IV. Summary & Recommendations

Web Services-based SOA plays an important role in facilitating the integration of disparate applications from various departments or trading partners and thus increasing business productivity.  The distributed nature of Web Services makes Gray Box testing ideal for detecting defects within a SOA.  Black Box testing provides rapid functional testing that can be used across distributed services; however, owing to the “blind” nature of Black Box testing, test coverage is limited, inefficient and redundant.  White Box testing is not practical for Web Services since access to source code or binaries in a Web Services deployment is usually impossible.  By leveraging the rich information presented in WSDL files, intelligent and efficient Gray Box test can be generated.  Further state-of-the-art techniques such as message mutation can be used to auto-generate a large array of test that can extract program internals – exception handling, states, flows – without having access to source or binaries.  Such techniques push the Gray Box testing closer to the results of White Box testing without dealing with its expense or intrusive characteristics.

V.  About Crosscheck Networks SOAPSonar™

Crosscheck Networks has built SOAPSonar™ to provide you comprehensive, code-free testing that is extremely easy to set up and run.  You will be generating Functional, Performance, Interoperability and Vulnerability Reports in minutes and leveraging non-intrusive and efficient Gray Box Testing Techniques through SOAPSonar™ and its patent-pending XSD-Mutation™ technology.

Contact Information

Website: [www.crosschecknet.com](http://www.crosschecknet.com/)

Email:     [support@crosschecknet.com](mailto:support@crosschecknet.com)

Phone:    1-888-CROSSCK (276-7725)

              1 617-938-3956 (from outside US)

[<Crosscheck your Web Services/>™](http://www.crosschecknet.com/)

SOA Testing Methodologies & Best Practices - A Torry Harris Whitepaper

This paper outlines an approach to SOA testing in an effective, reliable manner & describes various testing strategies, test cases & tools.

**SOA Test Methodology**

**Abstract**

[**Service-Oriented Architecture (SOA)**](http://www.thbs.com/services/integration-and-soa) promises significant benefits to today's organizations. Successfully delivering SOA benefits, especially Business Agility and Component Reuse, will be dependent on the Test Approach that your organization adopts to implement your SOA.

This white paper will provide a comprehensive guidance on best practices for testing SOA Solutions. This document includes a review of the following topics that will need to be addressed to ensure a successful SOA implementation:

* Why more testing effort will be required at the service level and not at the system level
* Why Security testing moves from an end of project activity to one that spans the entire project life cycle
* A SOA Test team will not only require a detailed technical understanding of your SOA, but they must be experts of domains within your business
* SOA Test Approach demands an appropriate tool strategy

**Introduction**

'To SOA or not to SOA' is not the question anymore. It is 'When to SOA?' With the maturity in SOA Implementations and realization of the associated benefits and challenges, increasingly Enterprises are including SOA Adoption in their Road Map.

**Service-Oriented Architecture, or SOA, enables IT departments to make the transition from an application-centric view of the world to a process-centric view.** Today, IT departments have the freedom to combine business services from multiple applications to deliver true end-to-end support for business processes. And, because the integration mechanism of SOA (usually Web Services) enables loosely coupled integration, IT departments can upgrade or change applications without impacting other applications.

Though SOA is being increasingly implemented both as green field (top down) and legacy modernization (bottom up), there is a clear lack of testing methodologies designed specifically for SOA applications. New approaches and methodologies are necessary to verify and validate applications based on SOA concepts.

**Why is SOA testing different?** The answer has many dimensions, but the bottom line is agility and flexibility. Yes, what makes SOA a very attractive, business friendly IT paradigm is the same reason why a different testing approach is required in SOA Implementations.

When it comes to [testing SOA applications](http://www.thbs.com/services/integration-and-soa/soa-testing), one has to look beyond functionality and performance (load) testing. SOA testing requires testing of interfaces and services that might bring together diverse systems and platforms, along with other performance (latency) and security related aspects.

One of the other **challenges** to be tackled in SOA Testing is the availability of the environment with the dependent underlying services and/or applications. For instance, an SOA Implementation might bring together two or more autonomous internal applications/services when composing a business process.

The availability of these internal applications/services becomes highly important during integration testing in parts as well as during end-to-end testing of the business process.

Starting with a brief introduction to SOA, this paper outlines an approach to testing a [**SOA Implementation**](http://www.thbs.com/knowledge-zone/a-guide-to-soa-implementation) in an effective and reliable manner. The paper further describes the various testing categories, suggested test strategy and an introduction to the available tools in the market that can be used to complement the overall Testing Approach.

**Service-Oriented Architecture Overview**

**What is SOA?**

A concise functional definition is provided below:

Oasis ([www.oasis-open.org](http://www.oasis-open.org/)) defines SOA as -- "An architectural style whose goal is to achieve loose coupling among interacting software agents / services."

Arasanjani, Borges and Holley define SOA as follows:

*“SOA is the architectural style that supports loosely coupled services to enable business flexibility in an interoperable, technology-agnostic manner. SOA consists of a composite set of business-aligned services that support a flexible and dynamically re-configurable end-to-end business processes realization using interface-based service descriptions.”*

**SOA Benefits & Implementation Principles**

SOA provides benefits in four basic categories:

* reducing integration expense
* increasing asset reuse
* increasing business agility
* reduction of business risk

These four core benefits actually offer return at many different levels and parts of the organization, depending on the set of business problems the company is applying SOA to.

**How should companies calculate the expected returns that tangible benefits provide the organization?** Only by understanding the full range of SOA value propositions, can companies begin to get a hand on calculating the ROI on SOA. Even then, it may not be possible to understand SOA's true ROI before the project is complete, because SOA addresses issues of fundamentally unpredictable business changes.

**The following key principles are recommended when implementing SOA:**

* Document the Business Processes. Be it bottom up or top down, availability of these Business
* Process documentation is critical in delivering SOA in its true self instead of say, Web Services based applications
* SOA Implementation is an evolution – start with a pilot, deliver business value and incrementally add on
* The SOA Implementation must be based on loosely coupled services that provide the highest flexibility and ongoing cost reduction due to reuse and lower maintenance
* The services should have standards compliant interfaces to enable seamless integration and interoperability with other services
* The business drives the services, and the services drive the technology
* Business agility is a fundamental to SOA

**Key Terms of Service-Oriented Architecture**

A service is representative of a repeatable business process/task. Services are used to encapsulate the functional units of an application by providing an interface that is well defined and implementation independent. Services can be invoked by other services or applications.

**Service orientation** defines a method of integrating business applications and processes as linked services.

**Service-oriented architecture (SOA)** can mean different things to different people depending on the person's role and context. From a business perspective, SOA defines a set of business services composed to capture the business design that the enterprise wants to expose internally, as well as its customers and partners. From an architecture perspective, SOA is an architectural style that supports service orientation. At an implementation level, SOA is fulfilled using a standards based infrastructure, programming model and technologies such as Web Services. From an operational perspective, SOA includes a set of agreements between service consumers and providers that specify the quality of service, as well as reporting on the key business and IT metrics.

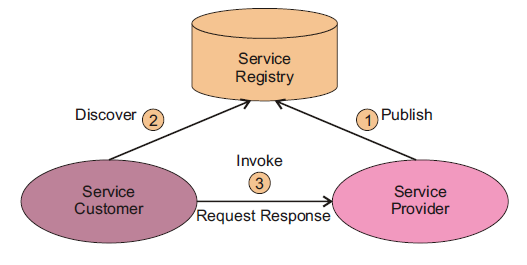
A **composite application** is a set of related and integrated services that support a business process built on SOA.

**Basic Components of SOA**

SOA consists of the following three components:

* Service provider
* Service consumer
* Service registry

Each component can also act as one of the two other components. For instance, if a service provider needs additional information that it can only acquire from another service, it acts as a service consumer. Figure 1-1 shows the operations each component can perform.



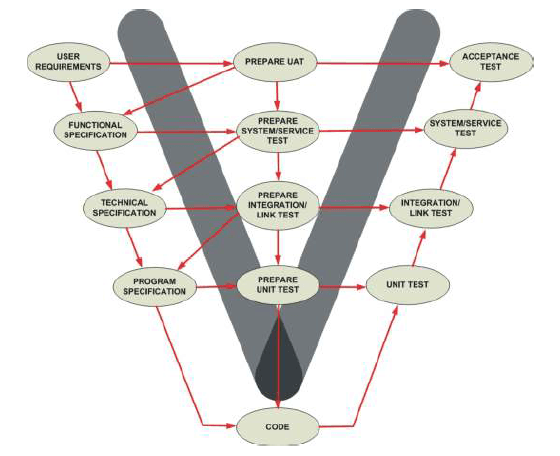
The service provider creates a service and in some cases publishes its interface and access information to a service registry.

The service registry is responsible for making the service interface and implementation access information available to service consumers.

The service consumer locates entries in the service registry and then binds to the service provider in order to invoke the defined service.

**Test Model**

Testing activities such as design, analysis, planning and execution must happen throughout the entire SOA project life cycle. The [**V-Model**](http://en.wikipedia.org/wiki/V-Model_(software_development)) is a suitable **testmethodology** that enforces testing discipline throughout the project life cycle. The project starts with defining the Business User Requirements. The V-Model would recommend that the **Business Acceptance Test Criteria** for these defined requirements are defined and agreed before moving to the start of the technical design phase. Before moving to the next phase/level of technical design, the model recommends test criteria defined for that level of technical requirements, and so on. The V-Model is simply encouraging the project team to continually determine how they would successfully test the project deliverables.



**The V-Model is a suitable test methodology to deliver SOA projects for the following reasons:**

* It encourages a top-down approach with respect to defining the business process requirements, high-level functional technical design, security etc.
* It then encourages a bottom-up test approach – test individual functions within a service, test an individual service then test a composite set of services through to testing an integrated process and finally testing a complete business system. SOA is 'loosely' coupled services and that is why a bottom up test approach is recommended.
* The levels reflect different viewpoints of testing on different levels of detail.
* The V-model encourages testing throughout the entire Software Development Life Cycle.

**SOA Governance**

[SOA Governance](http://www.thbs.com/knowledge-zone/soa-governance) is about ensuring that each new and existing service conforms to the standards, policies and objectives of an organization for the entire life of that service.

**Why is SOA Governance needed?**

SOA Governance plays an increasingly important role in today's challenging business environment. It provides structure, commitment and support for the development, implementation and management of SOA, as necessary, to ensure it achieves its objectives.

**SOA Governance provides the following benefits:**

* Realize business benefits of SOA
* Business process flexibility
* Improved time to market
* Maintaining Quality of Service (QoS)
* Ensuring consistency of service
* Measuring the right things
* Communicating clearly between businesses

**SOA Test Methodology**

**Traditional Test Approach**

[Traditional software testing](http://en.wikipedia.org/wiki/Software_testing) that focused on code-level testing has evolved with Distributed and Web Service architectures. Web application testing has introduced more testing of business logic through the application's user interface, which has proved to be critical when deploying new solutions. With SOA, the need to test the business logic still exists; however, many SOA services will not have a user interface, which is one of the new challenges to your test organization.

**Some of the new SOA Testing challenges are:**

* Services that do not have a user interface
* Data driven business logic within services
* External services to the organization
* The quality of 'service' software will be vital to promote reuse and facilitate business agility. Services that have known bugs and quality issues will not be reused by the development teams. A significant increase in testing activities and test assets (functional, performance and security regression suites that include sophisticated harnesses and stubs) will be required at a service (program) level
* Predicting the future usage of services to assist with performance, load, stress, scalability
* As your SOA evolves, security testing will have a higher priority and profile within your organizations test strategy

In SOA, Services are based on heterogeneous technologies. No longer can we expect to test an application that was developed by a unified group, as a single project, sitting on a single application server and delivering through a standardized browser interface. The ability to string together multiple types of components to form a business process requires unconstrained thinking from an architect's perspective, and test planning and scheduling complexities from a tester's perspective.

In SOA, application logic is in the middle-tier, operating within numerous technologies, residing outside the department, or even outside the company.

We know that to test SOA, you need to go far beyond merely testing a user interface or browser screen. Web Services (WSDL/SOAP) will be an important component for many SOAs, but if you're only testing Web Services, you are not likely to test the entire technology stack that makes up the application. What are the transactions happening at the messaging layer? Is the right entry being reflected in the database? In fact, many perfectly valid SOA applications will house business logic entirely outside of the Web Services.

To address the above challenges, organizations need to review and enhance their current test methodology. Many Test Tool vendors have now recognized the new SOA test challenges and have developed a new breed of tools to help organizations to plan, manage and automate SOA functional, performance and security testing.

**Revised Test Approach for SOA**

The governing and design principles of SOA, coupled with the benefits that SOA claims to deliver, will force changes to organizations' approaches to Software Testing and Quality Assurance. The following are some of the main reasons for change:

* The Test team will require broader set of technical skills: SOA testing will require test teams to have a detailed understanding of the underlying SOA architecture and technologies. This must include an understanding of Network Security
* A new breed of tools will be required to assist the test team quickly and accurately link planned test coverage to:
  + Business domain process rules
  + Service usage (other processes that reuse the service)
  + Technical design specifications
  + Configuration and version control
  + Security Risk Analysis

The new automated test tools will also have to facilitate non-GUI functional and performance testing at the service level.

* The Test team structure will require alignment to business domains (processes) and not by technologies. This will promote an effective and efficient business risk prioritized test approach at both the service and business process levels, to ensure that agility and speed to market is not compromising quality.
* Organizations will have to invest in developing and maintaining 'Test Assets' for key services – services should have functional and non-functional regression packs, sophisticated harnesses and stubs. Services will have to be delivered to the Integration and the UAT test phases with a statement of Quality that will guarantee performance and security, with well-defined and understood functional test coverage.

A higher level of quality will be required to actively promote and encourage service reuse.

**SOA Test Approach**

**Purpose**

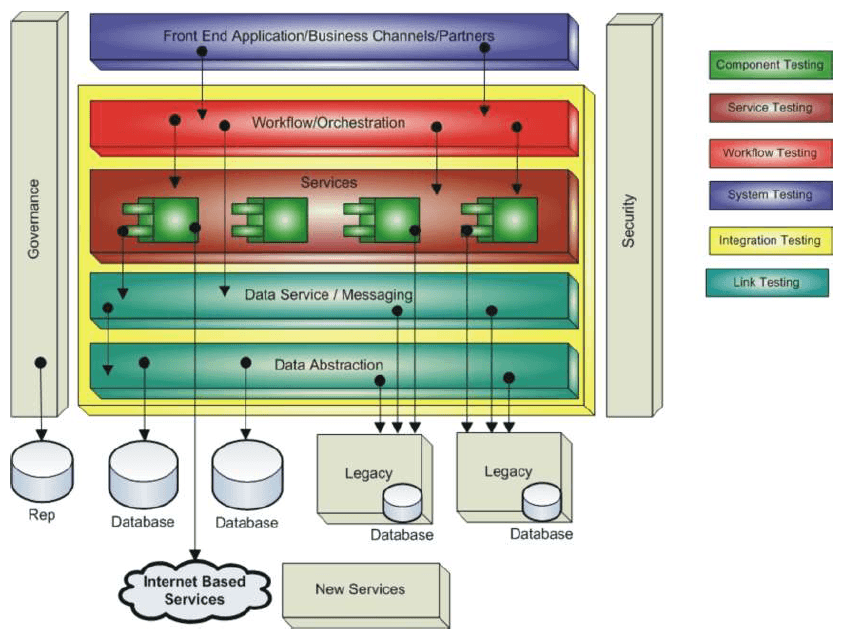
Testing SOA could be viewed as a complex computing problem. With any complex problem, the key is to break it down into smaller, more manageable components and build quality into these deliverables. The foundations to successful SOA testing are as follows:

* Equal weighting of testing effort throughout the project life cycle. Many organizations still fail to recognize the real benefits of static and formal review techniques during the early stages of the project. Most or all of the testing effort comes too late at the end of the project life cycle. More testing effort will be required at a service (program) level.
* The SOA test team is a blend of business domain and technology experts.
* Design the project test approach alongside the project business and technical requirements. Budget for the Test team to be involved from the start of the project.
* Implement Quality Controls throughout the project life cycle.
* Security Testing is not an end of project activity! Design and Plan Security testing from the start of the project.
* Test tools are a must!

**How Do You Test SOA Architecture?**

How do you test SOA architecture? You don't. Instead, you learn how to break down the architecture to its component parts, working from the most primitive to the most sophisticated, testing each component, then the integration of the holistic architecture. In other words, you have to divide the architecture into domains, such as services, security, and governance and test each domain separately using the recommended approach and tools.

SOA is loosely coupled with complex inter-dependencies and a SOA testing approach must follow the same pattern.



**Figure 2.1 represents a model of SOA components and how they're interrelated.** The Test team designing the Project Test approach and plans must have a macro understanding of how all of the components work independently and collectively.

**You can categorize SOA testing into the following phases:**

* Governance Testing
* Service-component-level testing
* Service-level testing
* Integration-level testing
* Process/Orchestration-level testing
* System-level testing
* Security Testing

**Governance Testing**

[SOA Governance](http://www.thbs.com/knowledge-zone/soa-governance) is a key factor in the success of any [SOA Implementation](http://www.thbs.com/knowledge-zone/a-guide-to-soa-implementation). It is also the most 'loosely' used term, as it covers the entire lifecycle of SOA Implementation – from design to run time to ongoing maintenance. SOA Governance refers to the Standards and Policies that govern the design, build and implementation of a SOA solution and the Policies that must be enforced during runtime.

Organizations must have well defined Design, Development, Testing and Security Standards that will guide and direct SOA implementations. Quality controls and reviews must be implemented throughout the entire Project life cycle to and processes, to ensure compliance. The appropriate peers must conduct these reviews and deviations from recommended standards must be agreed by the organization's Governance team.

**The following are examples of SOA Governance Policy types:**

* Quality of Service policies on Performance, Security and Transactions
* Regulatory policies – Sarbanes-Oxley
* Business policies – rules
* Audit policies – what events need to be logged, how long must events be kept, etc
* Infrastructure policies – access, backups, disaster recovery and failover

Test cases will be constructed and executed in all of the project test phases to determine if SOA Policies are being enforced. SOA policies can be enforced at runtime, by using technologies and/or monitoring tools.

SOA Governance testing will not be a separate test phase. Testing that SOA Governance is enforced will take place throughout the project life cycle, through formal peer reviews and different test scenarios that will be executed during the separate test phases.

**Service-component-level Testing**

Service-component level testing or Unit testing is normally performed by the developers to test that the code not only successfully compiles, but the basic functionality of the components and functions within a service are working as specified.

The primary goal of Component testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as you expect. Each Component is tested separately before integrating it into a service or services.

The following quality and test activities are recommended in this phase/level of testing:

* Formal peer reviews of the code to ensure it complies with organization standards and to identify any potential performance and security defects or weaknesses
* Quality entry and exit criteria are not only defined for this level of testing, but are achieved before moving to the next level of testing

**Service-level Testing**

Service testing will be the most important test level/phase within your SOA Test approach. Today, many organizations build a program or Web service, perform limited unit testing and accelerate its delivery to the integration test phase, to allow the test team to evaluate its quality. Service reuse will demand each service is delivered from this level/phase of testing with a comprehensive statement of quality and even a Guarantee!

The following quality and test activities are recommended in this phase/level of testing:

* Formal peer reviews of the code to ensure it complies with organization standards and to identify any potential interoperability, performance and security defects or weaknesses
* Functional, performance and security regression suites to be executed against the service. This will require the help of automated test tools and the development of sophisticated harnesses and stubs
* Quality entry and exit criteria are not only defined for this level of testing, but are achieved before delivering the service to the next level of testing

Service Level testing must ensure that the service is not only meeting the requirements of the current project, but more importantly, is still meeting the business and operational requirements of the other processes that are using that service.

**Integration-level Testing**

The integration test phase will focus on service interfaces. This test phase aims to determine if interface behaviour and information sharing between the services, are working as specified. The test team will ensure that all the services delivered to this test phase comply with the defined interface definition, in terms of standards, format and data validation. Integration testing test scenarios should also 'work' the layers of communications, the network protocols. This test phase may include testing external services to your organization.

**Process/Orchestration-level Testing**

Process/Orchestration testing ensures services are operating collectively as specified. This phase of testing would cover business logic, sequencing, exception handling and process decomposition (including service and process reuse).

**System-level Testing**

System Level testing will form the majority, if not all of the User Acceptance Test phase. This test phase will test that the SOA technical solution has delivered the defined business requirements and has met the defined business acceptance criteria. To ensure that this phase/level of testing is targeting only the key business scenarios of the solution, the business stakeholders and testers must fully understand the quality and test coverage that has been achieved in previous test phases.

**Security Testing**

As SOA evolves and grows within your organization, the profile and necessity of Security testing will increase. Today, many organizations perform an inadequate amount of penetration testing at the very end of a project. SOA combined with Government and Regulatory compliance, will require Security testing activities to be incorporated into the entire project life cycle. Section 4 discusses SOA Security testing in more detail.

Deliverables based on disciplines and key documentation

|  |  |
| --- | --- |
| **Life Cycle processes** | **Deliverables** |
| Test Planning and Control | Test Plan/Test Strategy |
| Test analysis and design | Requirements Traceability Document |
| Test implementation and execution | Test cases, Test Scripts, test data and test logs |
| Evaluating exit criteria and reporting | Status Report/Metrics and Defect summary Report |
| Test closure activities | Post-mortem document |

**Test Phases and Test Types**

SOA testing will span a number of test phases and test types. The following table summarizes test types that may be required in the different test phases.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **TEST PHASES** | **Guiding Documents** | **Functional** | **Performance** | **Interoperability** | **Backward Compatibility** | **Compliance** | **Security** |
| Component- Level Testing Phase | Technical Design, Program specification & Standards | √ | √ | √ | √ | √ | √ |
| Service- Level Testing Phase | Business Requirement s, Technical Design & Governance Standards and Policies | √ | √ | - | - | √ | √ |
| Integration & Orchestration Testing Phase | Business Requirement s, Technical Design & Governance Standards and Policies | √ | - | √ | √ | - | √ |
| System/ Process (User Acceptance) Testing Phase | Business Requirement s, Technical Design & Governance Standards and Policies | √ | √ | √ | √ | √ | √ |

**Functional Testing**

Functional or Black Box testing will determine if a component or a service or the whole system is operating to specification without reference to the internal technical workings or design. The Business Requirements and the Higher Level Technical Design definitions are the main inputs to Functional test case design.

**Performance**

As SOA grows and evolves over time, and more of the SOA components and services are reused, it will be critical that each of these components and services have a known performance and capacity under production loads and how they are scalable.

Organizations must move away from the misguided thought 'that you can only do performance, load and stress test on a fully integrated technical solution', to performance testing individual services and components of the SOA architecture. Many of the new SOA test tools support performance testing of services and components. Services must be delivered to the Integration test phase with tested and defined performance and capacity statements.

**Security**

SOA requires security testing to be designed and planned right from the start of the project. Security testing should be executed throughout the project test phases and not just when the complete system has been delivered at the end of the life cycle. Section 4 discusses SOA Security testing in more detail.

**Interoperability**

Interoperability is the ability of a system or a product to work with other systems or products without special effort on the part of the customer. Services will achieve interoperability by either strictly adhering to published interface standards or by using a broker service that will convert the data to the format of the other service interface on the 'fly'.

Design-time interoperability testing is not enough. Run-time Interoperability testing is also necessary for SOA. Comprehensive design-time testing combined with active run-time interoperability behaviour testing ensures that IT assets can integrate independent of platform, operating system, and programming language.

**Backward Compatibility**

Backward compatibility testing will determine if changes to an interface will affect existing users (otherwise known as service consumers) of the interface. If existing users are unaffected then the change is backward compatible. If existing users are affected then the change is not backward compatible, and a solution or strategy will be needed to manage the impact of the change. A service's interface will at some point have to change. Any change made to an interface must be assessed and tested for backward compatibility.

**Compliance**

Governance will be a key factor in successfully implementing SOA. SOA Governance is bound by Organization Standards and Policies. Compliance testing must be implemented throughout the entire project life cycle to ensure these Standards and Policies are enforced. Government and Regulatory Compliance will also demand this type of testing.

**Regression Strategy**

Regression testing is also known as validation testing and provides a consistent, repeatable validation of each change to services under development or being modified. Each time a defect is fixed, the potential exists to inadvertently introduce new errors, problems, and defects. An element of uncertainty is introduced about ability of the service to repeat everything that went right up to the point of failure. Regression testing is the selective retesting of a service or SOA system that has been modified to ensure that no previously working services fail as a result of the repairs.

Regression testing doesn't test that a specific defect has been fixed. The purpose of Regression testing is to ensure the service or system up to the point of repair, has not been adversely affected by the fix. Organizations must invest in the development and maintenance of functional and non-functional (performance and security) regression suites. It is recommended that a significant weighting of such suites should be aimed at key individual services and not just at the fully integrated systems. This will be essential if true service reuse is to be achieved. Automated test tools will be a key dependency on the cost effectiveness of executing and maintaining such regression suites.

**Security Testing**

SOA will raise the profile and necessity of Security testing. Many business processes within an organization will consist of several services, physically located in different parts of the corporate WAN, updating a number of databases and potentially sharing sensitive data with external organizations. This will raise the obvious question of **“How safe is the data as it navigates a complex internal and external network?”**

Today, many organizations perform Security Penetration testing at the very end of the project life cycle to cover all software security. Penetration testing is an authorized attempt to breach the security of a system using intruder and/or worm access techniques. Performing Penetration testing at the end of a project runs a significant risk of not only finding severe security bugs very late in the day, but also delivering a system that has an inadequate security design.

As your SOA evolves, your organizations networks will become more complex and it will not be possible to protect all assets. Security prioritization will be required to protect the most valuable company assets.

Government and Regulatory Compliance, Sarbanes-Oxley, FDIC and FISMA to name but a few, mandate that organizations regularly test the security of their networks and to provide audit findings. For these reasons, security analysis and testing must be incorporated into the entire Project Life Cycle.

* The Business Requirement definition must include security requirements.
* A security risk assessment should be performed during the technical design phases to prioritize and justify the required security testing.
* Formally review all technical deliverables have been built in accordance to your organizations defined security standards.
* Penetration security tests can be planned and executed at the component/service level and not just when a fully integrated system has been delivered.

Today, there are many commercial and open source Security test tools available. These tools have evolved from scanners that report potential security weaknesses to tools that actually execute specific types of penetration tests. Security tools are required if your organization desires a creditable and repeatable approach to security testing.

**User Acceptance Testing**

User Acceptance Testing (UAT) is a given when implementing new systems or processes. It is the formal means by which the new system or process does actually meet the essential business and operational requirements.

Today, many organizations, for a variety of reasons, only involve the key business stakeholders at the beginning of the project to define the business requirements and at the end of the project to perform the Acceptance Testing prior to production implementation. This increases the risk of the users requiring many tests to be repeated and duplicated during this phase, and finding high severity defects and missing requirements very late in the project life cycle.

Key Business Stakeholders and Users will need to be more actively involved throughout the entire project lifecycle. They need to understand how quality and testing is built into the entire project and not just at the end. They will need to be actively involved and fully understand in detail the increased testing effort surrounding delivering an individual service into the UAT phase. The User Acceptance Test phase should be short and targeted to determine if the solution is fit for purpose and not 'let us test everything again'.

**A fundamental principle of SOA is that Business will drive SOA not technology!**

The main challenge to the Test team will be to construct 'bridges' to the key Business Users and Operational stakeholders to ensure their active participation throughout the entire project.

**Risk Based Testing**

As SOA grows and evolves within your organization, with many business processes reusing many services, it will be possible to define infinite numbers of test scenarios for each SOA project. This is a key challenge that must be overcome if your organization is to successfully implement SOA and deliver on the promises.

Many organizations are now adopting a Risk Based Test approach. **What is Risk Based Testing?** A simple definition would be that the test scripts and cases planned to be executed, are justified and prioritized by agreed and understood financial implications to the business on failure and the likelihood of that failure occurring.

There are many articles and information available on Risk Based Testing. This paper does not intend to discuss the topic in detail. This paper is simply recommending that when designing and planning the test scripts and cases within each of the test phases outlined in this paper, that they are justified and prioritized by a defined risk to the project delivery and/or to the business when the solution is implemented.

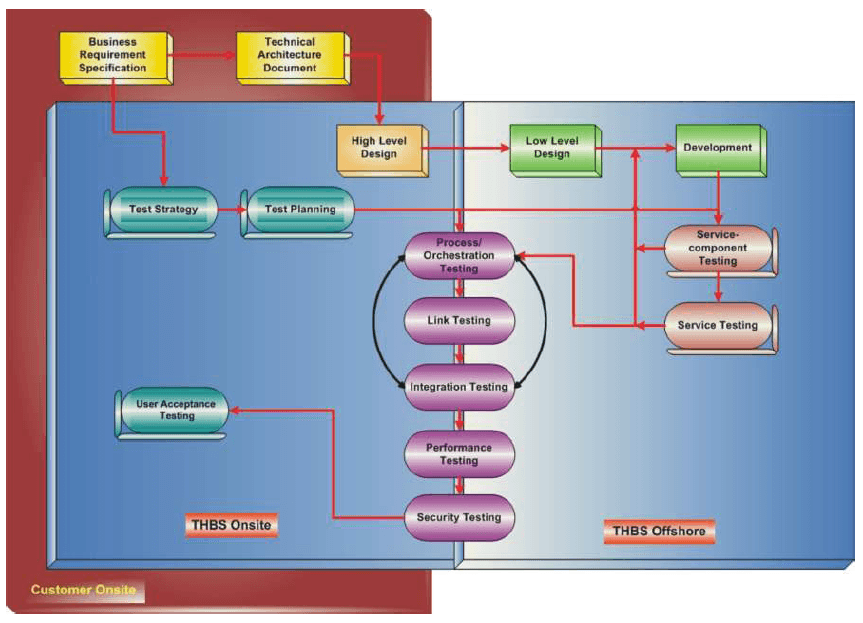
**Offshore onsite model**

The Onsite-Offshore Test Model is the most frequently used model, to deliver offshore cost savings to clients, without compromising on the quality of project deliverables. This model will offer real solutions to the new challenges of SOA testing. As stated in earlier sections, SOA will require significant test effort and activity at a service level. The principle reason behind this statement is that the service is to be reused. If a service has known defects and quality issues, then it will probably not be selected for reuse by the development teams.

SOA will demand that individual services are delivered to the Integration and User Acceptance Test phases with statements of qualities and guarantees on business functionality, performance and security. The following lists the main reasons why offshore testing could be the simple answer to this challenge:

* The significant cost savings offered by the offshore model promotes long term partnerships that will improve the quality and efficiencies of the testing service
* The abundance of professionally certified and technically skilled testing professionals that will be essential to building all the required test assets to perform all aspects of service level testing
* Offshore companies operate at SEI CMM level 5
* The offshore model can offer 'follow the sun' engagements that could help with Time to Market and business agility demands

The following diagram provides an overview of the typical onsite offshore test model. Typically, the project est. planning and requirement gathering are executed onsite at the client's location.



A small onsite team will work at the customer's location while a larger team is deployed at the offshore centre. The onsite team provides an interface with the client and coordinates the offshore work. The offshore team acts as a virtual extension of the onsite project team to execute test cases and report defects.

**Test tools and usage**

As stated previously in this paper, SOA requires a comprehensive test tool strategy. The following section gives a brief overview of the main Vendor and Open Source tools that offer benefits to SOA testing.

**Commercial Products**

**Green Hat GH Tester**

[Green Hat GH Tester](http://en.wikipedia.org/wiki/Green_Hat_(software_company)) is a graphical tool for testing message-based systems. It can be used to publish and subscribe easily to a wide range of protocols including JMS, SOAP and products like SONIC MQ and TIBCO RV. Its powerful test suite can be used to quickly create test stubs for adapters still under construction, and enable users to continue design and implementation of workflows without waiting for the real adapters.

**Mercury**

[Mercury](http://en.wikipedia.org/wiki/Mercury_Interactive) provides a suite of products - Quick Test, Service Test, Load Runner and Mercury Quality Centre. These together provide a solution for SOA functional test and regression test automation as well as performance testing. Built with Mercury's industry-leading Mercury Load Runner technology, it can greatly reduce testing time and help ensure that services will meet the functional and performance requirements of the business before being deployed into production.

More recently Mercury has acquired the Systinet Corporation. Systinet has brought significant expertise in SOA technologies and governance to the Mercury product offering.

**Parasoft SOAtest**

[Parasoft SOAtest](http://www.parasoft.com/soatest) is an automated Web Services testing product that allows users to verify all aspects of a Web Service. SOAtest supports WSDL validation, client/server unit and functional testing and performance testing. SOAtest addresses key Web Services and SOA development issues such as interoperability, security, change management, and scalability.

**AdventNet QEngine**

This is a complete [Web-based test automation tool](http://www.manageengine.com/products/qengine/) that supports functional and performance testing of Web applications and Web Services. The tool is developed using Java, which facilitates portability and multiple platform support (Windows and Linux).

**Borland SilkPerfomer SOA edition**

[SilkPerformer SOA edition](http://silkperformer-soa-edition.software.informer.com/) is an automated Web Services test tool that supports functional and performance testing of services and interoperability functional and performance testing between services.

**LISA WS – Testing**

[LISA WS-Testing](http://lisa-ws-testing.software.informer.com/) is a fully functional, no-code Web Services test authoring and execution solution that both developers and QA/Business teams can use. The solution supports all of the current protocols and unit/functional/regression tests you need to build and launch thorough test workflows against your WSDL and SOAP objects.

**Open Source Products**

**SOAP UI**

[SoapUI](http://www.soapui.org/)is a desktop application for inspecting, invoking and developing Web services over HTTP. The tool also supports functional, load and compliance testing. Functional and Load Testing can be performed manually using the soapUI and automatically using the soapUI command-line features.

**PushToTest TESTMAKER**

TestMaker is an open-source utility and framework to build and use intelligent test agents to check Webenabled applications for scalability, functionality and performance. TestMaker test agents implement user behaviour to drive a Web-enabled application as a real user would. TestMaker is a flexible, powerful central place to measure an application's ability to enable a user to achieve their goals.

**Conclusion**

The Service-Oriented Architecture (SOA) vision is everywhere, garnering almost universal acceptance among vendors and customers alike. The promise of business and IT alignment has undeniable appeal, and organizations are in various stages of SOA planning and adoption.

This paper has outlined a number of recommendations that will ensure a successful approach to SOA testing. The following summarizes the key recommendations:

* Design your SOA project test approach alongside the definition of business requirements and high level technical design. Ensure the test teams are involved right from the start.
* Static test techniques such as formal inspections are a must to ensure the business requirements and technical designs are defined to standards and are complete. Strong governance and disciplines are essential to successfully delivering SOA.
* SOA is driven by business and not technology. The test team will have to ensure the key business stakeholders and users are actively involved throughout the project life cycle and not just at the start and the end.
* The Test team must be aligned to business domains not technologies. This will enable a successful implementation of a Risk Based approach to testing. The test team must also be technical and be able to 'white box' test the entire SOA platform.
* Security assessment and testing must take place throughout the entire project life cycle.
* Many organizations will need to perform a 'leap of faith'. The majority of testing activities now need to be performed during the business requirements analysis, design and service build phases. To achieve delivery agility and true service reuse, each individual service must be delivered to the Integration and UAT test phases with well-defined functional test coverage and guarantees on performance, scalability and security. Simply put, services must come with a Quality statement!
* Test Tools are now a must. Organizations must invest appropriately in a test tool strategy.

You should not be surprised that many of the recommendations outlined in this paper are not new. Many of them are already defined as best practices. SOA demands strong governance, well-defined standards, processes and disciplines. If Quality is at the heart of your SOA, then the promises of SOA will be delivered.

[**How to access Amazon AWS S3?**](https://towardssoa.wordpress.com/2015/08/22/how-to-access-amazon-aws-s3/)

Posted by Vivek on August 22, 2015

Here are the steps to access Amazon S3 to access files stored on cloud.

1. Download AWS CLI Installer from [**https://s3.amazonaws.com/aws-cli/AWSCLI64.msi**](https://s3.amazonaws.com/aws-cli/AWSCLI64.msi). It can be 32 bit or 64 bit depending your system’s setting.
2. Once downloaded, Install the AWS CLI
3. Open the Command Prompt and navigate to the directory where AWS client is installed.
4. Enter the command from the bin directory- **aws configure**

Provide the Access Key and Secret Access Key when asked. You can leave the region name and output format fields blank. Example below:

C:\windows\system32>aws configure

AWS Access Key ID [None]: [Access Key]

AWS Secret Access Key [None]: [Secret Access Key]

Default region name [None]:

Default output format [None]:

You can use clients such as S3 Browser to connect and access the files.

If you want to change the credentials, go to user home. You should be able to find .aws directory. It has two files – config and credentials. Credentials file should have access key id and secret access key.

Posted in [**SOA**](https://towardssoa.wordpress.com/category/soa/) | [**Leave a Comment »**](https://towardssoa.wordpress.com/2015/08/22/how-to-access-amazon-aws-s3/#respond)

[**Testing/ Mocking WebServices using browser based plugins**](https://towardssoa.wordpress.com/2015/06/11/testing-mocking-webservices-using-browser-based-plugins/)

Posted by Vivek on June 11, 2015

REST has rapidly become a part of almost all SOA or integration architectures. While there are tools like SOAPUi that can be used to test REST services, there are plugins which are more light-weight and provide good features

1. **SOA Client** (Firefox plugin)

It can be used to access SOAP based web services as well as UDDI registries. It can be added using the following link:

2. **Postman** (Chrome plugin)

It is an advanced REST service testing tool.

3. **Mockable.io**

It is not a plugin but a site that allows user to mock services and provides them with a unique URL which can be shared with other users.

Not just REST, it also allows to mock SOAP services.

Posted in [**SOA**](https://towardssoa.wordpress.com/category/soa/) | [**Leave a Comment »**](https://towardssoa.wordpress.com/2015/06/11/testing-mocking-webservices-using-browser-based-plugins/#respond)

[**Ruby: Working with Postgres Database**](https://towardssoa.wordpress.com/2015/06/10/ruby-working-with-postgres-database/)

Posted by Vivek on June 10, 2015

Firstly, install a ruby gem pg using the following command

gem install  pg

Then write a class to connect to database and retrieve values from a table

**class *PostGresClient***

*#!/usr/bin/env ruby*

*require***‘pg’**

*# Output a table of current connections to the DB  
conn*= ***PG***.*connect*(**“localhost”**, 5432, **”**, **”**, **“”**, **“”**, **“”**)  
*res*= *conn*.exec( **“SELECT \* FROM public.\”Employee\””**)  
*res*.each **do**|*row*|  
*row*.each **do**|*column*|  
puts *column***end  
end**

end

Posted in [**SOA**](https://towardssoa.wordpress.com/category/soa/) | [**Leave a Comment »**](https://towardssoa.wordpress.com/2015/06/10/ruby-working-with-postgres-database/#respond)

[**Ruby: Writing message to ActiveMQ**](https://towardssoa.wordpress.com/2015/06/10/ruby-writing-message-to-activemq/)

Posted by Vivek on June 10, 2015

The following piece of code can be used to write a message to ActiveMQ queue

require **‘rubygems’**require **‘jms’**

*config*= {  
**:factory***=>***‘org.apache.activemq.ActiveMQConnectionFactory’**,  
**:broker\_url***=>***‘tcp://localhost:61616’**,  
**:require\_jars***=>*[**“~/Applications/apache-activemq-5.4.2/activemq-all-5.4.2.jar”**]  
}

***JMS***::***Connection***.*session*(*config*) **do**|*session*|  
*session*.producer(**:q\_name***=>***‘myQueue’**) **do**|*producer*|  
*producer*.*send*(*session*.message(**“Helloworld”**))  
**end  
end**

Posted in [**SOA**](https://towardssoa.wordpress.com/category/soa/) | [**Leave a Comment »**](https://towardssoa.wordpress.com/2015/06/10/ruby-writing-message-to-activemq/#respond)

[**Ruby: Reading message from ActiveMQ**](https://towardssoa.wordpress.com/2015/06/10/ruby-reading-message-from-activemq/)

Posted by Vivek on June 10, 2015

The following piece of code can be used to read a message from queue using Ruby.

*# Include JMS after ActiveMQ*require **‘rubygems’**

*require***‘jms’**

*# Connect to ActiveMQ  
config*= {  
**:factory***=>***‘org.apache.activemq.ActiveMQConnectionFactory’**,  
**:broker\_url***=>***‘tcp://localhost:61616’**,  
**:require\_jars***=>*[**“~/Applications/apache-activemq-5.4.2/activemq-all-5.4.2.jar”**]  
}  
***JMS***::***Connection***.*session*(*config*) **do**|*session*|  
*session*.consume(**:q\_name***=>***‘myQueue’**, **:timeout***=>*1000) **do**|*message*|  
p *message***end  
end**

Posted in [**SOA**](https://towardssoa.wordpress.com/category/soa/) | [**Leave a Comment »**](https://towardssoa.wordpress.com/2015/06/10/ruby-reading-message-from-activemq/#respond)

[**Ruby: Reading property file**](https://towardssoa.wordpress.com/2015/06/10/ruby-reading-property-file/)

Posted by Vivek on June 10, 2015

1. Create a yml file containing properties:

Consider the following properties file with a property

**inputfile:**C:\example\myfile.xml

1. Load the properties file using the following command:

yml = ***YAML***::*load*(***File***.open(**‘properties.yml’**))

1. Fetch the property using the following statement:

yml[**‘inputfile’**]

Posted in [**SOA**](https://towardssoa.wordpress.com/category/soa/) | [**Leave a Comment »**](https://towardssoa.wordpress.com/2015/06/10/ruby-reading-property-file/#respond)

[**Call for papers – Hadoop**](https://towardssoa.wordpress.com/2015/06/05/call-for-papers-hadoop/)

Posted by Vivek on June 5, 2015

The call for papers is open for [**Strata+Hadoop World 2015 in Singapore**](http://strataconf.com/big-data-conference-sg-2015), Dec 1-3.  
If you’re up for presenting and would like to share your ideas, case studies, best practices and technical knowledge [**submit your presentation proposals here**](http://strataconf.com/big-data-conference-sg-2015/public/cfp/391) before the June 18th deadline

Posted in [**SOA**](https://towardssoa.wordpress.com/category/soa/) | [**Leave a Comment »**](https://towardssoa.wordpress.com/2015/06/05/call-for-papers-hadoop/#respond)

[**Free Mulesoft ESB Training**](https://towardssoa.wordpress.com/2015/06/05/free-mulesoft-esb-training/)

Posted by Vivek on June 5, 2015

Register now !!

[**https://training.mulesoft.com/mulesoft\_u/developer\_essentials**](https://training.mulesoft.com/mulesoft_u/developer_essentials)

This is an **8-week self-study course** for developers, architects, and technical managers who want **to achieve ‘Associate MuleSoft Developer’ certification**.

Posted in [**SOA**](https://towardssoa.wordpress.com/category/soa/) | [**Leave a Comment »**](https://towardssoa.wordpress.com/2015/06/05/free-mulesoft-esb-training/#respond)

[**Register for the Mulesoft Forum 2015 (July 14, 2015).. Free!!**](https://towardssoa.wordpress.com/2015/06/05/register-for-the-mulesoft-forum-2015-july-14-2015-free/)

Posted by Vivek on June 5, 2015

If you are a CIO, Project Manager, an Integration Architect, or Developer, don’t miss the opportunity to engage in an interactive discussions with your peers around digital transformation. You will walk away with a number of innovative ways to improve and transform projects and business initiatives with API-led connectivity.

At MuleSoft Forum, you’ll have the opportunity to:

* Learn how Mule ESB Enterprise Edition offers reliability, performance, scalability, security – all out-of-the box.
* See MuleSoft’s connectivity platform in action in a short live demo
* Discover how API-Led Connectivity enables major business initiatives
* Attend the exclusive MuleSoft ecosystem networking reception

[**http://www.cvent.com/events/forum-2015-mumbai/event-summary-1684136a44bc44508de58022bc59c63c.aspx?lang=en&sms=2**](http://www.cvent.com/events/forum-2015-mumbai/event-summary-1684136a44bc44508de58022bc59c63c.aspx?lang=en&sms=2)

Posted in [**SOA**](https://towardssoa.wordpress.com/category/soa/) | [**Leave a Comment »**](https://towardssoa.wordpress.com/2015/06/05/register-for-the-mulesoft-forum-2015-july-14-2015-free/#respond)

[**ClassNotFOundException? – Command to find the required jar**](https://towardssoa.wordpress.com/2015/02/02/classnotfoundexception-command-to-find-the-required-jar/)

Posted by Vivek on February 2, 2015

It is irritating when one gets ClassNotFoundException. One can go and find the jar and add it to the classpath to resolve the problem. Jars can be searched on sites like [**http://www.findjar.com**](http://www.findjar.com/). However, sometimes jars are within the system and only thing needed to fix the problem is to set the classpath appropriately. Here is the unix command to find the jar:  
$ find . -name “\*.jar” -print -exec jar -tf ‘{}’ \; | grep -E “jar$|String\.class”

Posted in [**SOA**](https://towardssoa.wordpress.com/category/soa/) | Tagged: [**java**](https://towardssoa.wordpress.com/tag/java/) | [**Leave a Comment »**](https://towardssoa.wordpress.com/2015/02/02/classnotfoundexception-command-to-find-the-required-jar/#respond)

《[我所理解的RESTful Web API [Web标准篇]](http://www.cnblogs.com/artech/p/restful-web-api-01.html)》

《[我所理解的RESTful Web API [Web标准篇]](http://www.cnblogs.com/artech/p/restful-web-api-01.html)》Web服务已经成为了异质系统之间的互联与集成的主要手段，在过去一段不短的时间里，Web服务几乎清一水地采用SOAP来构建。构建REST风格的Web服务是最近两三年风行的潮流，所以很多人以为REST是一个事物。而事实却是：REST自其诞生之日起到现在（2014年）已经有14年了，它为什么叫这么一个“奇怪”的名字呢？

目录   
一、为什么叫这个“奇怪”的名字？二、采用URI标识资源   
二、采用URI标识资源   
三、使用“链接”关联相关的资源   
四、使用统一的接口   
五、使用标准的HTTP方法   
六、支持多种资源表示方式   
七、无状态性

# 一、为什么叫这个“奇怪”的名字？

[](http://images.cnitblog.com/blog/19327/201401/06074820-3b08e822599440c28ea3f402a7fac67e.jpg)

2000年，Roy Thomas Fielding博士在他那篇著名的博士论文《Architectural Styles and the Design of Network-based Software Architectures》中提出了几种软件应用的架构风格，REST作为其中的一种架构风格在这篇论文的第5章中进行了概括性的介绍。我个人建议本书的读者都能读读这篇论文，原文和中文译文都可以从网络上找到。

REST是“REpresentational State Transfer”的缩写，可以翻译成“表现状态转换”，但是在绝大多数场合中我们只说REST或者RESTful。为什么会起这么一个奇怪的名字呢？我们可以从上述这篇论文中找到答案。Fielding在论文中将REST定位为“分布式超媒体应用（Distributed Hypermedia System）”的架构风格，它在文中提到一个名为“HATEOAS（Hypermedia as the engine of application state）”的概念。

我们利用一个面向最终用户的Web应用来对这个概念进行简单阐述：这里所谓的应用状态（Application State）表示Web应用的客户端的状态，简单起见可以理解为会话状态。资源在浏览器中以超媒体的形式呈现，通过点击超媒体中的链接可以获取其它相关的资源或者对当前资源进行相应的处理，获取的资源或者针对资源处理的响应同样以超媒体的形式再次呈现在浏览器上。由此可见，超媒体成为了驱动客户端会话状态的转换的引擎。

借助于超媒体这种特殊的资源呈现方式，应用状态的转换体现为浏览器中呈现资源的转换。如果将超媒体进一步抽象成一般意义上的资源呈现（Representation ）方式，那么应用状态变成了可被呈现的状态（REpresentational State）。应用状态之间的转换就成了可被呈现的状态装换（REpresentational State Transfer），这就是REST。

REST在我看来是一种很笼统的概念，它代表一种架构风格。对于多个Web应用采用的架构，我们只能说其中某一个比其它的更具有REST风格，而不能简单粗暴地说：“它采用了REST架构而其它的没有”。为了将REST真正地落地，Lenoard Rechardson & Sam Ruby在《RESTful Web Services》一书中提出了一种名为“面向资源的架构（ROA： Resource Oriented Architecture）”。该书中介绍了一些采用ROA架构的Web服务应该具备的基本特征，它们可以指导我们如果构架具体的RESTful Web API。

# 二、采用URI标识资源

SOAP Web API采用RPC风格，它采用面向功能的架构，所以我们在设计SOAP Web API的时候首相考虑的是应高提供怎样的功能（或者操作）。RESTful Web API采用面向资源的架构，所以在设计之初首先需要考虑的是有哪些资源可供操作。

资源是一个很宽泛的概念，任何寄宿于Web可供操作的“事物”均可视为资源。资源可以体现为经过持久化处理保存到磁盘上的某个文件或者数据库中某个表的某条记录，也可以是Web应用接受到请求后采用某种算法计算得出的结果。资源可以体现为一个具体的物理对象，它也可以是一个抽象的流程。

一个资源必须具有一个或者多个标识，既然我们设计的Web API，那么很自然地应该采用URI来作为资源的标识。作为资源标识的URI最好具有“可读性”，因为具有可读性的URI更容易被使用，使用者一看就知道被标识的是何种资源，比如如下一些URI就具有很好的可读性。

* http://www.artech.com/employees/c001（编号C001的员工）
* http://www.artech.com/sales/2013/12/31（2013年12月31日的销售额）
* http://www.artech.com/orders/2013/q4（2013年第4季度签订的订单）

除了必要的标志性和可选的可读性之外，标识资源的URI应该具有“可寻址性（Addressability）”。也就是说，URI不仅仅指明了被标识资源所在的位置，而且通过这个URI可以直接获取目标资源。通过前面的介绍 我们知道URI具有URL和URN两种主要的表现形式，只要前者具有可寻址性，所以我们最好采用一个URL作为资源的标识。

URI除了可以标识某个独立的资源外（比如“http://www.artech.com/employees/c001”），还可以标识一组资源的集合或者资源的容器（比如“http://www.artech.com/orders/2013/q4”）。当然，一组同类资源的集合或者存放一组同类资源的容器本身也可以视为另一种类型的复合型（Composite）资源，所以“URI总是标识某个资源”这种说法是没有问题的。

# 三、使用“链接”关联相关的资源

在绝大多数情况下，资源并不会孤立地存在，必然与其它资源具有某种关联。既然我们推荐资源采用具有可寻址性的URL来标识，那么我们就可以利用它来将相关的资源关联起来。比如我们采用XML来表示一部电影的信息，那么我们采用如下的形式利用URL将相关的资源（导演、领衔主演、主演、编剧以及海报）关联在一起。实际上这可以视为一份超文本/超媒体文档。当用户得到这样一份文档的时候，可以利用自身的内容获得某部影片基本的信息，还可以利用相关的“链接”得到其它相关内容的详细信息。

1: <movie>

2: <name>魔鬼代言人</name>

3: <genre>剧情|悬疑|惊悚</genre>

4: <directors>

5: <add ref="http://www.artech.com/directors/taylor-hackford">泰勒.海克福德</add>

6: </directors>

7: <starring>

8: <add ref = "http://www.artech.com/actors/al-pacino">阿尔.帕西诺</add>

9: <add ref = "http://www.artech.com/actors/keanu-reeves ">基诺.李维斯</add>

10: </starring>

11: <supportingActors>

12: <add ref = "http://www.artech.com/actors/charlize-theron ">查理兹.塞隆</add>

13: <add ref = "http://www.artech.com/actors/jeffrey-jones ">杰弗瑞.琼斯</add>

14: <add ref = "http://www.artech.com/actors/connie-nielsen">康尼.尼尔森</add>

15: </supportingActors>

16: <scriptWriters>

17: <add ref = "http://www.artech.com/scriptwriters/jonathan-lemkin">乔纳森•莱姆金</add>

19: <add ref = "http://www.artech.com/scriptwriters/tony-gilroy">托尼•吉尔罗伊 </add>

20: </scriptWriters>

21: <language>英语</language>

22: <poster ref = "http://www.artech.com/images/the-devil-s-advocate"/>

23: <story>...</story>

24: </movie>

Fielding在他的论文中将REST定位为“分布式超媒体应用”的架构风格，而超媒体的核心就是利用“链接”相关的信息结成一个非线性的网，所以从一点也可以看出REST和“使用链接关联相关的资源”这个特性使吻合的。

# 四、使用统一的接口

由于REST是面向资源的，所以一个Web API旨在实现针对单一资源的操作。我们在前面已经说个，针对资源的基本操作唯CRUD而已，这是使我们可以为Web API定义标准接口成可能。所谓的标准接口就是针对不同资源的Web API定义一致性的操作来操作它们，其接口可以采用类似于下面的模式。

1: public class ResourceService

2: {

3: public IEnumerable<Resource>[] Get();

4: public void Create(Resource resource);

5: public void Update(Resource resource);

6: public void Delete(string id);

7: }

能否采用统一接口是RESTful Web API和采用RPC风格的SOAP Web服务又一区别。如果采用RPC风格的话，我们在设计Web API的时候首先考虑的是具体哪些功能需要被提供，所以这样的Web API是一组相关功能的集合而已。

以一个具体的场景为例。现在我们需要设计一个Web API来管理用于授权的角色，它只需要提供针对角色本身的CRUD的功能以及建立/解除与用户名之间的映射关系。如果我们将其定义成针对SOAP的Web服务，其服务接口具有类似于如下的结构。

1: public class RoleService

2: {

3: public IEnumerable<string> GetAllRoles();

4: public void CreateRole(string roleName);

5: public void DeleteRole(string roleName);

6:

7: public void AddRolesInUser(string userName, string[] roleNames);

8: public void RemoveRolesFromUser(string userName, string[] roleNames);

9: }

如下我们需要将其定义成一个纯粹的RESTful的Web API，只有前面三个方法在针对角色的CRUD操作范畴之内，但是后面两个方法却可以视为针对“角色委派（Role Assignment）”对象的添加和删除操作。所以这里实际上涉及到了两种资源，即角色和角色委派。为了使Web API具有统一的接口，我们需要定义如下两个Web API。

1: public class RolesService

2: {

3: public IEnumerable<string> Get();

4: public void Create(string roleName);

5: public void Delete(string roleName);

6: }

7:

8: public class RoleAssignmentsService

9: {

10: public void Create(RoleAssignment roleName);

11: public void Delete(RoleAssignment roleName);

12: }

# 五、使用标准的HTTP方法

由于RESTful Web API采用了同一的接口，所以其成员体现为针对同一资源的操作。对于Web来说，针对资源的操作通过HTTP方法来体现。我们应该将两者统一起来，是Web API分别针对CRUD的操作只能接受具有对应HTTP方法的请求。

我们甚至可以直接使用HTTP方法名作为Web API接口的方法名称，那么这样的Web API接口就具有类似于如下的定义。对于ASP.NET Web API来说，由于它提供了Action方法名称和HTTP方法的自动映射，所以如果我们采用这样的命名规则，就无需再为具体的Action方法设定针对HTTP方法的约束了。

1: public class ResourceService

2: {

3: public IEnumerable<Resource>[] Get();

4: public void Post(Resource resource);

5: public void Put(Resource resource);

6: public void Patch (Resource resource);

7: public void Delete(string id);

8:

9: public void Head(string id);

10: public void Options();

11: }

上面代码片断提供的7个方法涉及到了7个常用的HTTP方法，接下来我们针对资源操作的语义对它们作一个简单的介绍。首先GET、HEAD和OPTIONS这三个HTTP方法旨在发送请求以或者所需的信息。对于GET，相应所有人对它已经非常熟悉了，它用于获取所需的资源，服务器一般讲对应的资源置于响应的主体部分返回给客户端。

HEAD和OPTIONS相对少见。从资源操作的语义来讲，一个针对某个目标资源发送的HEAD请求一般不是为了获取目标资源本身的内容，而是得到描述目标资源的元数据信息。服务器一般讲对应资源的元数据置于响应的报头集合返回给客户端，这样的响应一般不具有主体部分。OPTIONS请求旨在发送一种“探测”请求以确定针对某个目标地址的请求必须具有怎样的约束（比如应该采用怎样的HTTP方法以及自定义的请求报头），然后根据其约束发送真正的请求。比如针对“跨域资源”的预检（Preflight）请求采用的HTTP方法就是OPTIONS。

至于其它4中HTTP方法（POST、PUT、PATCH和DELETE），它们旨在针对目标资源作添加、修改和删除操作。对于DELETE，它的语义很明确，就是删除一个已经存在的资源。我们着重推荐其它三个旨在完成资源的添加和修改的HTTP方法作一个简单的介绍。

通过发送POST和PUT请求均可以添加一个新的资源，但是两者的不同之处在于：对于前者，请求着一般不能确定标识添加资源最终采用的URI，即服务端最终为成功添加的资源指定URI；对于后者，最终标识添加资源的URI是可以由请求者控制的。也正是因为这个原因，如果发送PUT请求，我们一般直接将标识添加资源的URI作为请求的URI；对于POST请求来说，其URI一般是标识添加资源存放容器的URI。

比如我们分别发送PUT和POST请求以添加一个员工，标识员工的URI由其员工ID来决定。如果员工ID由客户端来指定，我们可以发送PUT请求；如果员工ID由服务端生成，我们一般发送POST请求。具体的请求与下面提供的代码片断类似，可以看出它们的URI也是不一样的。

1: PUT http://www.artech.com/employees/300357 HTTP/1.1

2: ...

3:

4: <employee>

5: <id>300357</id>

6: <name>张三</name>

7: <gender>男<gender>

8: <birthdate>1981-08-24</birthdate>

9: <department>3041</department>

10: </employee>

1: POST http://www.artech.com/employees HTTP/1.1

2: ...

3:

4: <employee>

5: <name>张三</name>

6: <gender>男<gender>

7: <birthdate>1981-08-24</birthdate>

8: <department>3041</department>

9: </employee>

POST和PUT请求一般将所加资源的内容置于请求的主体。但是对于PUT请求来说，如果添加资源的内容完全可以由其URI来提供，这样的请求可以不需要主体。比如我们通过请求添加一个用于控制权限的角色，标识添加角色的URI由其角色名称来决定，并且不需要指定除角色名称的其它信息，那么我们只要发送如下一个不含主体的PUT请求即可。

1: PUT http://www.artech.com/roles/admin HTTP/1.1

2:

3: ...

除了进行资源的添加，PUT请求还能用于资源的修改。由于请求包含提交资源的标识（可以放在URI中，也可以置于保存在主体部分的资源内容中），所以服务端能够定位到对应的资源予以修改。对于POST和PUT，也存在一种一刀切的说法：POST用于添加，PUT用于修改。我个人比较认可的是：如果PUT提供的资源不存在，则做添加操作，否则做修改。

对于发送PUT请求以修改某个存在的资源，服务器一般会将提供资源将原有资源整体“覆盖”掉。如果需要进行“局部”修改，我们推荐请求采用PATCH方法，因为从语义上讲“Patch”就是打补丁的意思。

## 安全性与幂等性

关于HTTP请求采用的这些个方法，具有两个基本的特性，即“安全性”和“幂等性”。对于上述7种HTTP方法，GET、HEAD和OPTIONS均被认为是安全的方法，因为它们旨在实现对数据的获取，并不具有“边界效应（Side Effect[[1]](file:///E:\\????????????/ASP.NET%20Web%20API????????????/???1???%20ASP.NET%20Web%20API??????/" \l "_ftn1_9470)）”。至于其它4个HTTP方法，由于它们会导致服务端资源的变化，所以被认为是不安全的方法。

幂等性（Idempotent）是一个数学上的概念，在这里表示发送一次和多次请求引起的边界效应是一致的。在网速不够快的情况下，客户端发送一个请求后不能立即得到响应，由于不能确定是否请求是否被成功提交，所以它有可能会再次发送另一个相同的请求，幂等性决定了第二个请求是否有效。

上述3种安全的HTTP方法（GET、HEAD和OPTIONS）均是幂等方法。由于DELETE和PATCH请求操作的是现有的某个资源，所以它们是幂等方法。对于PUT请求，只有在对应资源不存在的情况下服务器才会进行添加操作，否则只作修改操作，所以它也是幂等方法。至于最后一种POST，由于它总是进行添加操作，如果服务器接收到两次相同的POST操作，将导致两个相同的资源被创建，所以这是一个非幂等的方法。

当我们在设计Web API的时候，应该尽量根据请求HTTP方法的幂等型来决定处理的逻辑。由于PUT是一个幂等方法，所以携带相同资源的PUT请求不应该引起资源的状态变化，如果我们在资源上附加一个自增长的计数器表示被修改的次数，这实际上就破坏了幂等型。

不过就我个人的观点来说，在有的场合下针对幂等型要求可以不需要那么严格。举个例子，我对于我们开发的发部分应用来说，数据表基本上都有一个名为LastUpdatedTime的字段表示记录最后一次被修改的时间，因为这是为了数据安全审核（Auditing）的需要。在这种情况下，如果接收到一个基于数据修改的PUT请求，我们总是会用提交数据去覆盖现有的数据，并将当前服务端时间（客户端时间不可靠）作为字段LastUpdatedTime的值，这实际上也破坏了幂等性。

可能有人说我们可以在真正修改数据之前检查提交的数据是否与现有数据一致，但是在涉及多个表链接的时候这个“预检”操作会带来性能损失，而且针对每个字段的逐一比较也是一个很繁琐的事情，所以我们一般不作这样的预检操作。

# 六、支持多种资源表示方式

资源和资源的表示（Representaion）是两个不同的概念，资源本身是一个抽象的概念，是看不见摸不着的，而看得见摸得着的是资源的表现。比如一个表示一个财年销售情况的资源，它既可以表示为一个列表、一个表格或者是一个图表。如果采用图表，又可以使用柱状图、K线图和饼图等，这一切都是针对同一个资源的不同表示。

我们说“调用Web API获取资源”，这句话其实是不正确的，因为我们获取的不是资源本身，仅仅是资源的某一种表示而已。对于Web来说，目前具有两种主流的数据结构，XML和JSON，它们也是资源的两种主要的呈现方式。在多语言环境下，还应该考虑描述资源采用的语言。

我们在设计Web API的时候，应该支持不同的资源表示，我们不能假定请求提供的资源一定表示成XML，也不能总是以JSON格式返回获取的资源，正确的做法是：根据请求携带的信息识别提交和希望返回的资源表示。对于请求提交的资源，我们一般利用请求的Content-Type报头携带的媒体类型来判断其采用的表示类型。对于响应资源表示类型的识别，可以采用如下两种方式。

* 让请求URI包含资源表示类型，这种方式使用的最多的是针对多语言的资源，我们一般讲表示语言（也可以包含地区）的代码作为URI的一部分，比如“http://www.artech.com/en/orders/2013”表示将2013年的订单以英文的形式返回。
* 采用“内容协商（Content Negotiation）”根据请求相关报头来判断它所希望的资源表示类型，比如“Accept”和“Accept-language”报头可以体现请求可以接受的响应媒体类型和语言。

对于上述两种资源表示识别机制，我们很多人会喜欢后者，因为第一种不够“智能”。实际上前者具有一个后者不具有的特性：“浏览器兼容型”[[2]](file:///E:\\????????????/ASP.NET%20Web%20API????????????/???1???%20ASP.NET%20Web%20API??????/" \l "_ftn2_9470)。对于Web API开发来说，浏览器应该成为一种最为常用的测试工具。在不借助任何插件的情况下，我们利用浏览器访问我们在地址栏中输入的URI时对生成的请求内容不能作任何干预的，如果与资源表示相关的信息（比如语言、媒体类型）被直接包含到请求的URI中，那么所有的情况都可以利用浏览器直接测试。

有人从另一方面对“URI携带资源表示类型”作了这样的质疑：由于URI是资源的标识，那么这导致了相同的资源具有多个标识。其实这是没有问题的，URI是资源的唯一标识，但不是其“唯一的唯一标识“，相同的资源可以具有多个标识。

# 七、无状态性

RESTful只要维护资源的状态，而不需要维护客户端的状态。对于它来说，每次请求都是全新的，它只需要针对本次请求作相应的操作，不需要将本次请求的相关信息记录下来以便用于后续来自相同客户端请求的处理。

对于上面我们介绍的RESTful的这些个特性，它们都是要求我们为了满足这些特征做点什么，唯有这个无状态却是要求我们不要做什么，因为HTTP本身就是无状态的。举个例子，一个网页通过调用Web API分页获取符合查询条件的记录。一般情况下，页面导航均具有“上一页”和“下一页”链接用于呈现当前页的前一页和后一页的记录。那么现在有两种实现方式返回上下页的记录。

* Web API不仅仅会定义根据具体页码的数据查询定义相关的操作，还会针对“上一页”和“下一页”这样的请求定义单独的操作。它自身会根据客户端的Session ID对每次数据返回的页面在本地进行保存，以便能够知道上一页和下一页具体是哪一页。
* Web API只会定义根据具体页码的数据查询定义相关的操作，当前返回数据的页码由客户端来维护。

第一种貌似很“智能”，其实就是一种画蛇添足的作法，因为它破坏了Web API的无状态性。设计无状态的Web API不仅仅使Web API自身显得简单而精炼，还因减除了针对客户端的“亲和度（Affinty）”使我们可以有效地实施负载均衡，因为只有这样集群中的每一台服务器对于每个客户端才是等效的。

[[1]](file:///E:\\????????????/ASP.NET%20Web%20API????????????/???1???%20ASP.NET%20Web%20API??????/" \l "_ftnref1_9470) 大部分计算机书籍都将Side Effect翻译成“副作用”，而我们一般将“副（负）作用”理解为负面的作用，其实计算机领域Side Effect表示的作用无所谓正负，所以我们觉得还是还原其字面的含义“边界效用”。除此之外，对于GET、HEAD和OPTIONS请求来说，如果服务端需要对它们作日志、缓存甚至计数操作，严格来说这也算是一种Side Effect，但是请求的发送者不对此负责。

[[2]](file:///E:\\????????????/ASP.NET%20Web%20API????????????/???1???%20ASP.NET%20Web%20API??????/" \l "_ftnref2_9470) 这里的“兼容”不是指支持由浏览器发送的请求，因为通过执行JavaScript脚本可以让作为宿主的浏览器发送任何我们希望的请求，这里的兼容体现在尽可能地支持浏览器访问我们在地址栏中输入的URI默认发送的HTTP-GET请求。

参考资料：   
[1] 《HTTP： The Definitive Guide》, By By David Gourley, Brian Totty, Marjorie Sayer, Anshu Aggarwal, Sailu Reddy   
[2] 《RESTful Web Services》, RESTful Web Services   
[3] 《A Brief Introduction to REST》，<http://www.infoq.com/articles/rest-introduction>   
[4] 《TCP/IP Illustrated (Volumn 1: The Protocol)》, by W. Richard Stevens

[我所理解的RESTful Web API [Web标准篇]](http://www.cnblogs.com/artech/p/restful-web-api-01.html)   
[我所理解的RESTful Web API [设计篇]](http://www.cnblogs.com/artech/p/restful-web-api-02.html)

分类: [[04] 架构思想](http://www.cnblogs.com/artech/category/219608.html)

标签: [REST](http://www.cnblogs.com/artech/tag/REST/), [RESTful](http://www.cnblogs.com/artech/tag/RESTful/), [Web API](http://www.cnblogs.com/artech/tag/Web%20API/)

[**好文要顶**](javascript:void(0);) [**关注我**](javascript:void(0);) [**收藏该文**](javascript:void(0);) **[http://common.cnblogs.com/images/icon_weibo_24.png](javascript:void(0);)** **[http://common.cnblogs.com/images/wechat.png](javascript:void(0);)**

[http://pic.cnblogs.com/face/u19327.jpg](http://home.cnblogs.com/u/artech/)

[Artech](http://home.cnblogs.com/u/artech/)  
[关注 - 52](http://home.cnblogs.com/u/artech/followees)  
[粉丝 - 6268](http://home.cnblogs.com/u/artech/followers)

荣誉：[推荐博客](http://www.cnblogs.com/expert/)

[+加关注](javascript:void(0);)

50

0

(请您对文章做出评价)

[«](http://www.cnblogs.com/artech/p/restful-web-api-01.html)上一篇：[我所理解的RESTful Web API [Web标准篇]](http://www.cnblogs.com/artech/p/restful-web-api-01.html)  
[»](http://www.cnblogs.com/artech/p/web-api-sample.html)下一篇：[在一个空ASP.NET Web项目上创建一个ASP.NET Web API 2.0应用](http://www.cnblogs.com/artech/p/web-api-sample.html)

posted @ 2014-01-06 07:48 [Artech](http://www.cnblogs.com/artech/) 阅读(54274) 评论(26) [编辑](http://i.cnblogs.com/EditPosts.aspx?postid=3506553) [收藏](http://www.cnblogs.com/artech/p/3506553.html)

**评论列表**

[#1楼](http://www.cnblogs.com/artech/p/3506553.html#2852536) 2014-01-06 08:13 [害怕飞的鸟](http://www.cnblogs.com/cmnh4/) 

老大加油

[支持(0)反对(0)](javascript:void(0);)

[#2楼](http://www.cnblogs.com/artech/p/3506553.html#2852575) 2014-01-06 09:14 [天朝码农](http://www.cnblogs.com/yxb88858/) 

哇，老大终于更新了

[支持(0)反对(0)](javascript:void(0);)

[#3楼](http://www.cnblogs.com/artech/p/3506553.html#2852600) 2014-01-06 09:27 [Json](http://www.cnblogs.com/xingxiaobai/) 

楼主，全新大作什么时候面世？

[支持(0)反对(0)](javascript:void(0);)

[#4楼](http://www.cnblogs.com/artech/p/3506553.html#2852639)[楼主] 2014-01-06 10:00 [Artech](http://www.cnblogs.com/artech/) 

[@](http://www.cnblogs.com/artech/p/3506553.html#2852600)Json  
[引用](http://www.cnblogs.com/artech/p/3506553.html#2852600)楼主，全新大作什么时候面世？  
已经完成，不过我还打算进行两轮修改！

[支持(1)反对(0)](javascript:void(0);)

[#5楼](http://www.cnblogs.com/artech/p/3506553.html#2852643)[楼主] 2014-01-06 10:02 [Artech](http://www.cnblogs.com/artech/) 

[@](http://www.cnblogs.com/artech/p/3506553.html#2852575)洛阳码农  
[引用](http://www.cnblogs.com/artech/p/3506553.html#2852575)哇，老大终于更新了  
不至于“终于“吧::)

[支持(0)反对(0)](javascript:void(0);)

[#6楼](http://www.cnblogs.com/artech/p/3506553.html#2852645)[楼主] 2014-01-06 10:02 [Artech](http://www.cnblogs.com/artech/) 

[@](http://www.cnblogs.com/artech/p/3506553.html#2852536)害怕飞的鸟  
[引用](http://www.cnblogs.com/artech/p/3506553.html#2852536)老大加油  
加油:)

[支持(0)反对(0)](javascript:void(0);)

[#7楼](http://www.cnblogs.com/artech/p/3506553.html#2852653) 2014-01-06 10:04 [王清培](http://www.cnblogs.com/wangiqngpei557/) 

蒋哥辛苦，小弟前来支持；

[支持(0)反对(0)](javascript:void(0);)

[#8楼](http://www.cnblogs.com/artech/p/3506553.html#2852803) 2014-01-06 11:30 [loncin](http://www.cnblogs.com/loncin/) 

Web API 和一般的web网站做的接口有什么区别？

[支持(0)反对(0)](javascript:void(0);)

[#9楼](http://www.cnblogs.com/artech/p/3506553.html#2852961)[楼主] 2014-01-06 13:54 [Artech](http://www.cnblogs.com/artech/) 

[@](http://www.cnblogs.com/artech/p/3506553.html#2852653)王清培  
[引用](http://www.cnblogs.com/artech/p/3506553.html#2852653)蒋哥辛苦，小弟前来支持；  
谢谢支持！

[支持(0)反对(0)](javascript:void(0);)

[#10楼](http://www.cnblogs.com/artech/p/3506553.html#2852962)[楼主] 2014-01-06 13:55 [Artech](http://www.cnblogs.com/artech/) 

[@](http://www.cnblogs.com/artech/p/3506553.html#2852803)loncin  
[引用](http://www.cnblogs.com/artech/p/3506553.html#2852803)Web API 和一般的web网站做的接口有什么区别？  
形式之一！

[支持(0)反对(0)](javascript:void(0);)

[#11楼](http://www.cnblogs.com/artech/p/3506553.html#2852972) 2014-01-06 13:59 [上校](http://www.cnblogs.com/zhuawang/) 

新瓶装旧酒，微软的传道者

[支持(0)反对(0)](javascript:void(0);)

[#12楼](http://www.cnblogs.com/artech/p/3506553.html#2853011)[楼主] 2014-01-06 14:27 [Artech](http://www.cnblogs.com/artech/) 

[@](http://www.cnblogs.com/artech/p/3506553.html#2852972)上校  
[引用](http://www.cnblogs.com/artech/p/3506553.html#2852972)新瓶装旧酒，微软的传道者  
这篇文章有说“瓶”吗？和微软有1毛钱关系？

[支持(0)反对(0)](javascript:void(0);)

[#13楼](http://www.cnblogs.com/artech/p/3506553.html#2857937) 2014-01-10 12:50 [vbfool](http://www.cnblogs.com/vbfool/) 

[@](http://www.cnblogs.com/artech/p/3506553.html#2853011)Artech  
唉，这世界就是这样，每个人都恨不得踩上微软一脚以表现自己牛逼，没了靶子，也要乱放几枪，起码留个响。  
没意思啊

[支持(1)反对(0)](javascript:void(0);)

[#14楼](http://www.cnblogs.com/artech/p/3506553.html#2914491) 2014-04-10 10:50 [Bellen](http://home.cnblogs.com/u/382965/) 

最近在开始学习使用，读后印象更加深刻了

[支持(0)反对(0)](javascript:void(0);)

[#15楼](http://www.cnblogs.com/artech/p/3506553.html#2918584) 2014-04-15 19:38 [CodeYu](http://www.cnblogs.com/codeyu/) 

今天看两篇。明天接着看。谢谢A大。  
  
**只要**前者具有可寻址性  
应为：只有  
  
所以从**一点**也可以看出REST和“使用链接关联相关的资源”这个特性**使**吻合的。  
分别应为：这一点，是  
  
由于RESTful Web API采用了**同一**的接口  
是否为：统一？  
  
发送请求以**或者**所需的信息。  
应为：获取  
  
请求**着**一般不能确定标识添加资源最终采用的URI，  
应为：者

[支持(1)反对(0)](javascript:void(0);)

[#16楼](http://www.cnblogs.com/artech/p/3506553.html#2920218) 2014-04-17 23:39 [疾风的流浪](http://home.cnblogs.com/u/625111/) 

[@](http://www.cnblogs.com/artech/p/3506553.html#2918584)CodeYu  
[引用](http://www.cnblogs.com/artech/p/3506553.html#2918584)今天看两篇。明天接着看。谢谢A大。  
  
只要前者具有可寻址性  
应为：只有  
  
所以从一点也可以看出REST和“使用链接关联相关的资源”这个特性使吻合的。  
分别应为：这一点，是  
  
由于RESTful Web API采用了同一的接口  
是否为：统一？  
  
发送请求以或者所需的信息。  
应为：获取  
  
请求着一般不能确定标识添加资源最终采用的URI，  
应为：者  
好仔细哦。上一篇也是这么仔细。

[支持(0)反对(0)](javascript:void(0);)

[#17楼](http://www.cnblogs.com/artech/p/3506553.html#2920807)[楼主] 2014-04-18 17:17 [Artech](http://www.cnblogs.com/artech/) 

[@](http://www.cnblogs.com/artech/p/3506553.html#2918584)CodeYu  
[引用](http://www.cnblogs.com/artech/p/3506553.html#2918584)今天看两篇。明天接着看。谢谢A大。  
  
只要前者具有可寻址性  
应为：只有  
  
所以从一点也可以看出REST和“使用链接关联相关的资源”这个特性使吻合的。  
分别应为：这一点，是  
  
由于RESTful Web API采用了同一的接口  
是否为：统一？  
  
发送请求以或者所需的信息。  
应为：获取  
  
请求着一般不能确定标识添加资源最终采用的URI，  
应为：者  
惭愧：）

[支持(0)反对(0)](javascript:void(0);)

[#18楼](http://www.cnblogs.com/artech/p/3506553.html#2988698) 2014-07-17 15:51 [korall](http://home.cnblogs.com/u/651724/) 

按博主的理解，是不是一句话：  
RESTful Web API 就是 以 URI 为 查询语句的一个数据查询系统的实现

[支持(0)反对(0)](javascript:void(0);)

[#19楼](http://www.cnblogs.com/artech/p/3506553.html#2988770)[楼主] 2014-07-17 16:49 [Artech](http://www.cnblogs.com/artech/) 

[@](http://www.cnblogs.com/artech/p/3506553.html#2988698)korall  
[引用](http://www.cnblogs.com/artech/p/3506553.html#2988698)按博主的理解，是不是一句话：  
RESTful Web API 就是 以 URI 为 查询语句的一个数据查询系统的实现  
为什么你只考虑到“查询”？

[支持(0)反对(0)](javascript:void(0);)

[#20楼](http://www.cnblogs.com/artech/p/3506553.html#2990161) 2014-07-18 17:45 [korall](http://home.cnblogs.com/u/651724/) 

哦 习惯了 SQL 的 Q，虽然SQL 也不仅仅是查询。  
似乎从这边文章里面，我能得到的信息就是 RESTful Web API 越 RESTful 就越像是数据查询(好吧，CRUD) API

[支持(0)反对(0)](javascript:void(0);)

[#21楼](http://www.cnblogs.com/artech/p/3506553.html#2990169) 2014-07-18 17:47 [korall](http://home.cnblogs.com/u/651724/) 

这也被当成我对RESTfull Web API 的理解，不知道偏了没有？

[支持(0)反对(0)](javascript:void(0);)

[#22楼](http://www.cnblogs.com/artech/p/3506553.html#3006205) 2014-08-12 18:09 [korall](http://home.cnblogs.com/u/651724/) 

似乎下面的例子正好印证了我的观点啊：  
[Linq to Amazon](http://weblogs.asp.net/fmarguerie/Introducing-Linq-to-Amazon)[http://weblogs.asp.net/fmarguerie/Introducing-Linq-to-Amazon](http://www.cnblogs.com/artech/p/null)  
  
var query =  
from book in new Amazon.BookSearch()  
where   
book.Title.Contains("ajax") &&  
(book.Publisher == "Manning") &&  
(book.Price <= 25) &&  
(book.Condition == BookCondition.New)  
select book;

[支持(0)反对(0)](javascript:void(0);)

[#23楼](http://www.cnblogs.com/artech/p/3506553.html#3129300) 2015-02-13 14:50 [慢慢飞](http://home.cnblogs.com/u/723521/) 

现在在用，写得不错

[支持(0)反对(0)](javascript:void(0);)

[#24楼](http://www.cnblogs.com/artech/p/3506553.html#3137147) 2015-03-08 14:27 [QthCN](http://home.cnblogs.com/u/728492/) 

您好，请教一个问题。  
比如设计一个论坛系统，每个用户都可以发表评论，因此对于评论可以由如下URI：  
查看所有评论：GET 127.0.0.1/users/11111/discussions  
查看所有用户： GET 127.0.0.1/users  
通过组合，我可以得到这个论坛的所有评论。  
  
但还有一种方法：  
查看所有评论：GET 127.0.0.1/discussions  
查看某个人的评论： GET 127.0.0.1/discussions?user=11111  
  
此时对于评论这个resource来说，就有两种URI了。那我我应该提供哪种URI，还是同时提供？如果同时提供，后端的controller也是分开写吗？如果同时提供，那我response body中返回的releated link应该包含哪种URI？  
  
多谢。

[支持(0)反对(0)](javascript:void(0);)

[#25楼](http://www.cnblogs.com/artech/p/3506553.html#3241289) 2015-08-04 21:43 [高海东](http://www.cnblogs.com/ghd258/) 

我有一个疑问 一般查询get一两个条件用url参数方式 但是在复杂查询条件可能有十几个 甚至更多的时候用url参数就有问题 安全性和字符过多的问题 请问这种情况怎么设计 用什么方式

[支持(0)反对(0)](javascript:void(0);)

[#26楼](http://www.cnblogs.com/artech/p/3506553.html#3341980) 2016-01-06 10:43 [zlphoenix](http://www.cnblogs.com/zlphoenix/) 

@Artech大哥 有两个问题想向您请教下：  
1.引入RESTFul风格的架构是为了解决什么样的问题的呢？它适用于什么样的场景，什么场景不适合？  
  
2.如你文中所讲，Restful通过7个HTTP 谓词GET/POST/PUT/DELETE/PATCH/HEAD/OPTIONS 表示动作，而且不应该通过API的URI部分表达动作，那对于一个资源来说，上述７个动作以外的动作应该怎样表达?比如电商系统的下达销售订单这个请求，API应该怎么设计？是salesOrder/booking/{id}这样？还是通过Post请求将订单状态更新。如果是后者相当于将后台业务实现暴露到前端来处理，是不是必须在领域模型SalesOrder之上设计一个用于发布服务的SalesOrderResource通过它的状态改变来转换成SalesOrder的动作？这一点我比较迷惑也是学习RESTFulApi一直以来的一个困惑，能否麻烦你帮我解释一下？谢谢！

[支持(0)反对(0)](javascript:void(0);)

[刷新评论](javascript:void(0);)[刷新页面](http://www.cnblogs.com/artech/p/3506553.html)[返回顶部](http://www.cnblogs.com/artech/p/3506553.html#top)

**注册用户登录后才能发表评论，请**[**登录**](javascript:void(0);)**或**[**注册**](javascript:void(0);)**，**[**访问**](http://www.cnblogs.com/)**网站首页。**

[【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库](http://www.ucancode.com/index.htm)  
[【推荐】融云即时通讯云－专注为 App 开发者提供IM云服务](http://www.rongcloud.cn/)  
[【推荐】UCloud开年大礼，充5000返1000；买云主机送CDN，详情点击](http://www.ucloud.cn/site/active/new_gift.html?utm_source=cost&utm_campaign=bokeyuan1&utm_medium=display&utm_content=3yuecu)

**最新IT新闻**:  
· [埃里森：微软和亚马逊要壮大只能向甲骨文看齐](http://news.cnblogs.com/n/541186/)  
· [饿了么遭上海食药监局罚款12万元 再被立案调查](http://news.cnblogs.com/n/541171/)  
· [捅破窗户纸：揭开创业圈的三大谎言](http://news.cnblogs.com/n/541185/)  
· [首款Ubuntu平板4月发售：四核处理器拍照强大](http://news.cnblogs.com/n/541184/)  
· [为什么你的项目要花这么长时间？](http://news.cnblogs.com/n/541094/)  
» [更多新闻...](http://news.cnblogs.com/)

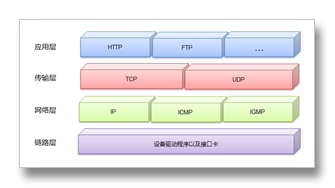
REST不是一个标准，而是一种软件应用架构风格。基于SOAP的Web服务采用RPC架构，如果说RPC是一种面向操作的架构风格，而REST则是一种面向资源的架构风格。REST是目前业界更为推崇的构建新一代Web服务（或者Web API）的架构风格。由于REST仅仅是一种价格风格，所以它是与具体的技术平台无关的，也就是说采用REST架构的应用未必一定建立在Web之上，所以在正式介绍REST之前，我们先来简单认识一下Web。

目录   
一、TCP/IP与HTTP   
二、Web资源   
      媒体类型   
      URI、URL和URN   
三、HTTP事务   
      HTTP方法   
      响应状态码   
四、HTTP报文

如果要问大家这样一个问题：“在过去半个世纪中，哪种信息技术对人类的影响最为深远？”，我想很多人的答案是Web（World Wide Web、WWW、W3或者万维网），因为它改变了我们的生活方式和思维方式。如果各位阅读过W3C介绍WWW的官方文档（“http://www.w3.org/WWW/”），应该对它的第一句话记忆犹新——“The World Wide Web (known as "WWW', "Web" or "W3") is the universe of network-accessible information, the embodiment of human knowledge”。如果将这句话翻译成简洁的中文，就是“Web是（网络）信息的来源，知识的化身”。

Web为我们提供了一种利用HTTP协议获取和操作网络资源的方式，这些将Web服务器作为宿主的资源不仅仅包含像文字和图片这些传统的信息载体，还包含音频和视频这些多媒体信息。Web的核心主要体现在三个方面，即HTTP、超文本（Hypertext）和超媒体（Hypermedia）[[1]](file:///E:\\????????????/ASP.NET%20Web%20API????????????/???1???%20ASP.NET%20Web%20API??????/" \l "_ftn1_5509)，超文本和超媒体规范了网络信息的表现形式，而HTTP则提供了网络访问的标准协议。接下来我们就以围绕着HTTP对Web作一下基本的介绍。

# 一、TCP/IP与HTTP

[](http://images.cnitblog.com/blog/19327/201401/05190925-9f6bcbb34c394a7e80fe30ca85ba0cce.png)

TCP/IP是以IP和TCP协议为核心的一整套网络协议的总称，所以有时候我们也称其为TCP/IP协议簇。毫不夸张地说，TCP/IP支撑着整个互联网，因为它就是互联网采用的网络协议。TCP/IP协议簇划分为如右图所示的4个层次[[2]](file:///E:\\????????????/ASP.NET%20Web%20API????????????/???1???%20ASP.NET%20Web%20API??????/" \l "_ftn2_5509)（应用层、传输层、网络层和链路层），构成整个协议簇的各个子协议处于相应层次中。

既然将整个协议簇命名为TCP/IP，那么IP和TCP自然就是其中最为核心的两个协议了。处于网络层的IP协议提供的IP数据报传输是不可靠的，因为它只承诺尽可能地将数据报发送出去，但不能保证发送的数据报能够成功地抵达目的地。IP协议的不可靠性还体现在它不能检测数据在传输过程中是否发生了改变，也就是说数据的完整性得不到保证。IP协议是一个无连接（Connectionless）的网络协议，每次数据报的处理对它来说均是独立的，因此IP协议也不能提供针对有序传输（数据接收的顺序与发送的顺序一致）的保证。

虽然IP协议只能提供不可靠的数据传输，同时有序传输也得不到保证，但是建立在它之上的传输层协议TCP有效地解决了这两个问题。TCP是一个基于连接的协议，数据交换双方在进行报文传输之前需要建立连接，报文传输结束之后需要关闭连接。这是一个双工（Duplex）连接，数据交换的双工均可以利用它向对方发送数据。

TCP利用“接收确认”和“超时重传”机制确保了数据能够成功抵达目的地。具体来说，接收方在成功接收到数据之后会回复一个确认消息。发送方在本地具有一个存放尚未得到确认的已发消息的缓冲区，如果发送方在一个设定的时限内没有接收到针对某个已发报文的确认消息，它会从该缓存区中选择对应的报文进行重新发送。在接收到确认之后，相应的报文会从缓存区中移除。

为了解决有序传输的问题，发送方会为每个报文进行编号，报文的序号体现了它们被发送的顺序。接收端在接收到某个报文之后，它会利用此序号判断是否具有尚未成功接收的已发报文，如果有的话，该报文会被存放到本地的缓冲区中。等到之前发送的报文全部被接收之后，接收方按照序号对接收的报文依次向上（应用层）递交，成功递交的报文会被从缓存区中移除。除了接收到“失序”的报文之外，接收方还有可能接收到重复的报文，因为没有报文均具有一个唯一的序号，如果该序号小于已经成功递交或者添加到缓存区中的报文序号，它会被认为是重复接收的报文而被丢弃。

由于每个TCP报文段都具有一个16位的检验和（Checksum），所以接收方可以根据它确认数据在传输过程中是否被篡改。除此之外，TCP还提供了“流量控制”功能避免了双方因缓存区大小不一致而导致报文丢失。具体来说，如果发送方的缓冲区大于接收方的缓存区，会导致接收方在缓冲区已满的情况下无法处理后续接收的报文，所以接收方会将自己缓存区剩余的大小及时通知给发送端，后者据此控制报文发送“流量”。

HTTP（Hypertext Transfer Protocol），全称为“超文本传输协议”，是TCP/IP协议簇的一部分。从图1-1可以看出，这是一个位于应用层的网络协议，在它之下的就是TCP协议。由于TCP协议是一个“可靠”的协议，HTTP自然也能提供可靠数据传输功能。

IP协议利用IP地址来定位数据报发送的目的地，而利用域名系统（DNS）可以实现域名与IP地址之间的转换。TCP协议利用端口号标识应用程序，所以某个应用程序在使用TCP协议进行通信的时候必须指定目标应用的IP地址（或者域名）和端口号。HTTP默认采用的端口号为80，而HTTPS（利用TLS/SSL为HTTP提供传输安全保障）的默认端口号则为443，当然在网络可达的前提下，我们可以指定任意的端口。

# 二、Web资源

这里所说的资源是一个宽泛的概念，任何寄宿于Web服务器可以利用HTTP协议获取或者操作的“事物”均可以称为资源。这也是一个抽象的概念，不仅仅是寄宿于Web服务器的某个静态物理文件可以视为Web资源，通过Web应用根据请求动态生成的数据也是Web资源。

## 媒体类型

资源实际上是一种承载着某种信息的数据，相同的信息可以采用不同形态的数据来展现，数据的“形态”主要体现为展示数据所采用的格式，比如一个数据对象可以通过XML格式来表示，也可以通过JSON格式来表示。数据的处理必须依赖于一种已知的格式，所以将Web资源的形态以一种标准化的方式固定下来显得尤为重要，这就是我们接下来着重介绍的媒体媒体（Media Type）。

不论是通过HTTP请求从Web服务器上获取资源，还是利用请求向服务器提交资源，响应或者请求的主体（Body）除了包含承载资源本身的数据之外，其报头（Header）部分还应该包含表示数据形态的媒体类型。

媒体类型又被称为MIME（Multipurpose Internet Mail Extension）类型，MIME是一个互联网标准，它扩展了电子邮件标准，使其能够支持非ASCII字符、二进制格式附件等多种格式的邮件消息。由于MIME在电子邮件系统应用得非常好，所以被HTTP用于描述并标记多媒体内容。下面的列表给出了一种常用的媒体类型。

* text/html：HTML格式的文档。
* text/xml（application/xml）：XML格式的文本。
* text/json（application/json）： JSON格式的文本。
* image/gif（image/jpeg、image/png）：GIF（JPEG、PNG）格式的图片。
* audio/mp4（audio/mpeg、audio/vnd.wave）：MP4（MPEG、WAVE）格式的音频文件。
* video/mp4（video/mpeg、video/quicktime）：MP4（MPEG、QUICKTIME）格式的视频文件。

## URI、URL和URN

可操作的Web资源应该具有一个 唯一的标识。虽然具有很多唯一性标志符的种类可供选择（比如GUID），但是采用URI来标识Web资源已经成为了一种共识，实际上URI的全称为“统一资源标志符（Uniform Resource Identifier）”。

我想有很多人弄不清楚URI和URL之间的区别，有人甚至觉得这是同一概念的不同表述而已。一个URL肯定是一个URI，但是一个URI并不一定是一个URL，URL仅仅是URI的一种表现形式而已。两者的差异其实可以直接从其命名来区分，URI是Web资源的标志符，所以只要求它具有“标识性”即可；URL全称为“统一资源定位符（Uniform Resource Locator）”，所以除了标识性之外，它还具有定位的功能，用于描述Web资源所在的位置。

URL不仅仅用于定位目标资源所在的位置，还指名了获取资源所采用的协议，一个完整的URL包含协议名称、主机名称（IP地址或者域名）、端口号、路径和查询字符串5个部分。比如对于“ http://www.artech.com:8080/images/photo.png?size=small”这样一个URL，上述的5个部分分别是“http”、“www.artech.com”、“8080”、“/images/photo.png”和“?size=small”。

除了URL，URN也是URI的一种表现形式，URN全称“统一资源定位符（Uniform Resource Name）”。URN与资源所在的位置无关，倘若采用URN来唯一标识某个资源，在位置发生改变的时候标志符依然可以保持不变。URN一般也不会涉及到获取被标识资源采用的网络协议，所以不需要为利用不同协议访问的相同资源定义不同的标志符。

# 三、HTTP事务

虽然TCP是一种基于连接的传输层协议，并且保存双方针对同一个连接的多轮消息交换的会话状态，但是建立其上的HTTP则是一种无状态的网络协议。HTTP采用简单的“请求/响应”消息交换模式，一次HTTP事务（Transaction）始于请求的发送，止于响应的接收。针对客户端和Web服务器的多次消息交换来说，每个HTTP事务均是相互独立的。

## HTTP方法

HTTP采用简单的请求/响应模式的消息交换旨在实现针对某个Web资源的某种操作。至于针对资源的操作类型，不外乎CRUD（Create、Retrieve、Update和Delete）而已。一个HTTP请求除了利用URI标志目标资源之外，还需要通过HTTP方法（HTTP Method或者HTTP Verb）指名针对资源的操作类型。我们常用的HTTP方法 包括GET、POST、PUT、DELETE、HEAD、OPTIONS、TRACE、CONNECTION和PATCH等，我们将在《[设计篇](http://www.cnblogs.com/artech/p/restful-web-api-02.html)》以REST的视角来对它们进行详细介绍。

## 响应状态码

针对客户端向Web服务器发送的任意一个HTTP请求，不论在何种情况下得到一个响应，每个响应均具有一个由3位数字表示的状态码和相应的描述文字。不同数值的状态码体现了不同类型的响应状态，W3C对响应状态码的范围作了如下的规范。

* 100~199：信息状态码，代表请求已被接受，需要继续处理。
* 200~299：成功状态码，代表请求已成功被服务器接收、理解、并接受。
* 300~399：重定向状态码，代表需要客户端采取进一步的操作才能完成请求。
* 400~499：客户端错误状态码，代表了客户端看起来可能发生了错误，妨碍了服务器的处理。
* 500~599：服务器错误状态码，代表了服务器在处理请求的过程中有错误或者异常状态发生，也有可能是服务器意识到以当前的软硬件资源无法完成对请求的处理。

# 四、HTTP报文

客户端和Web服务器在一次HTTP事务中交换的消息被称为HTTP报头，客户端发送给服务器的请求消息被称为请求报文，服务器返回给客户端的响应消息被称为响应报头。请求报文和响应报头采用纯文本编码，由一行行简单的字符串组成。一个完整的HTTP报文由如下三个部分构成。

* 起始行：代表HTTP报文的第一行文字，请求报文利用起始行表示采用的HTTP方法、请求URI和采用的HTTP版本，而响应报文的起始行在承载着HTTP版本和响应状态码等信息。
* 报头集合：HTTP报文的起始行后面可以包含零个或者多个报头字段。每个报头表现为一个键/值对，键和值分别表示报头名称和报头的值，两者通过冒号（“：”）进行分割。HTTP报文采用一个空行作为报头集合结束的标志。
* 主体内容：代表报头集合结束标志的空行之后就是HTTP报文的主体部分了。客户端提交给服务器的数据一般置于请求报头的主体，而响应报头的主体也承载着服务器返回给客户端的数据。不论是请求报文还是响应报文，其主体部分均是可以缺省的。

接下来我们看看一个具体HTTP报文具有怎样的结构。下面这个文本片段反映的是我们通过Chrome浏览器访问微软的官网（www.microsoft. com）对应的HTTP请求，起始行体现了HTTP请求的三个基本属性，即HTTP方法（GET）、目标资源（http://www.microsoft.com/en-us/default.aspx）和协议版本（HTTP/1.1）。

1: GET http://www.microsoft.com/en-us/default.aspx HTTP/1.1

2: Host: www.microsoft.com

3: Connection: keep-alive

4: Cache-Control: max-age=0

5: User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.7 (KHTML, like Gecko) Chrome/16.0.912.75 Safari/535.7

6: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

7: Accept-Encoding: gzip,deflate,sdch

8: Accept-Language: en-US,en;q=0.8

9: Accept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.3

10: Cookie: ...

上述这个请求报文不具有主体，所以起始行之外的所有内容均为报头集合，我们们可以根据这些报头获得主机名称、采用的缓存策略、浏览器相关信息、以及客户端支持的媒体类型（Media Type）、编码方式、语言和字符集等。

前面的HTTP请求通过浏览器发送给服务端之后会接收到具有如下结构的响应报文，我们可以此从它的起始行得到采用的HTTP版本（HTTP/1.1）和响应状态码（“200 OK”，表示请求被正常接收处理）。响应的内容被封装到响应报文的主体部分，其媒体类型的通过报头“Content-Type”表示。由于该响应报文的主体内容是一个HTML文档，所以“Content-Type”报头表示的媒体类型为“text/html”。

1: HTTP/1.1 200 OK

2: Cache-Control: no-cache

3: Pragma: no-cache

4: Content-Type: text/html; charset=utf-8

5: Content-Encoding: gzip

6: Expires: -1

7: Vary: Accept-Encoding

8: Server: Microsoft-IIS/7.5

9: X-AspNet-Version: 2.0.50727

10: VTag: 791897542300000000

11: P3P: CP="ALL IND DSP COR ADM CONo CUR CUSo IVAo IVDo PSA PSD TAI TELo OUR SAMo CNT COM INT NAV ONL PHY PRE PUR UNI"

12: X-Powered-By: ASP.NET

13: Date: Wed, 18 Jan 2012 07:06:25 GMT

14: Content-Length: 34237

15:

16: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

17: <html>…</html>

[[1]](file:///E:\\????????????/ASP.NET%20Web%20API????????????/???1???%20ASP.NET%20Web%20API??????/" \l "_ftnref1_5509)超文本/超媒体（HyperText/HyperMedia）：超文本是一份呈现文本内容的电子文档，其核心在于可以利用内嵌的“超链接（Hyperlink）”直接访问引用的另一份文档。超媒体对超文本作了简单的扩展以呈现多媒体内容（比如图片、音频和视频等）。HTML文档是我们常见的最为典型的超文本/超媒体文件。

[[2]](file:///E:\\????????????/ASP.NET%20Web%20API????????????/???1???%20ASP.NET%20Web%20API??????/" \l "_ftnref2_5509) 除了采用这种4个层次的划分方法之外，还具有另外两种典型的划分方式。其中一种在链路层下面添加一个基于物理网络硬件的物理层，这种划分方法与此没有本质的区别。另外一种则是将TCP/IP协议簇划分为包括应用层、表示层、会话层、传输层、网络层、链路层和物理层在内的7个层次。

参考资料：   
[1] 《HTTP： The Definitive Guide》, By By David Gourley, Brian Totty, Marjorie Sayer, Anshu Aggarwal, Sailu Reddy   
[2] 《RESTful Web Services》, RESTful Web Services   
[3] 《A Brief Introduction to REST》，<http://www.infoq.com/articles/rest-introduction>   
[4] 《TCP/IP Illustrated (Volumn 1: The Protocol)》, by W. Richard Stevens

[我所理解的RESTful Web API [Web标准篇]](http://www.cnblogs.com/artech/p/restful-web-api-01.html)   
[我所理解的RESTful Web API [设计篇]](http://www.cnblogs.com/artech/p/restful-web-api-02.html)

分类: [[04] 架构思想](http://www.cnblogs.com/artech/category/219608.html)