

# The Art of SOA Testing

Ejaz Jamil  
Chief Architect,  
Soalib Incorporated

## Table of Contents

<u>The Art of SOA Testing.....</u>	<u>3</u>
<u>  1.1 Abstract.....</u>	<u>3</u>
<u>  1.2 Introduction.....</u>	<u>3</u>
<u>  1.1.1 Exclusive Server Testing.....</u>	<u>3</u>
<u>  1.1.2 Exclusive Client Testing.....</u>	<u>5</u>
<u>  1.2.1 Exclusive Integration Testing.....</u>	<u>6</u>
<u>  1.2.2 Complete SOA Testing.....</u>	<u>8</u>
<u>  1.2 Test Instances.....</u>	<u>10</u>
<u>  1.3 Conclusion.....</u>	<u>11</u>
<u>  1.4 About.....</u>	<u>11</u>
<u>  1.4.1 About Soalib, Inc.....</u>	<u>11</u>
<u>  1.4.2 About the Author.....</u>	<u>11</u>
<u>  1.5 Glossary.....</u>	<u>11</u>

# The Art of SOA Testing

*By Ejaz Jamil, MSEE, MBA, Chief Architect, Soalib Incorporated*

## 1.1 Abstract

*Testing a web service based Service Oriented Architecture software is a complex process when it comes to true SOA where multiple clients and server operating environments play an important role on how this testing is designed and how it is performed.*

## 1.2 Introduction

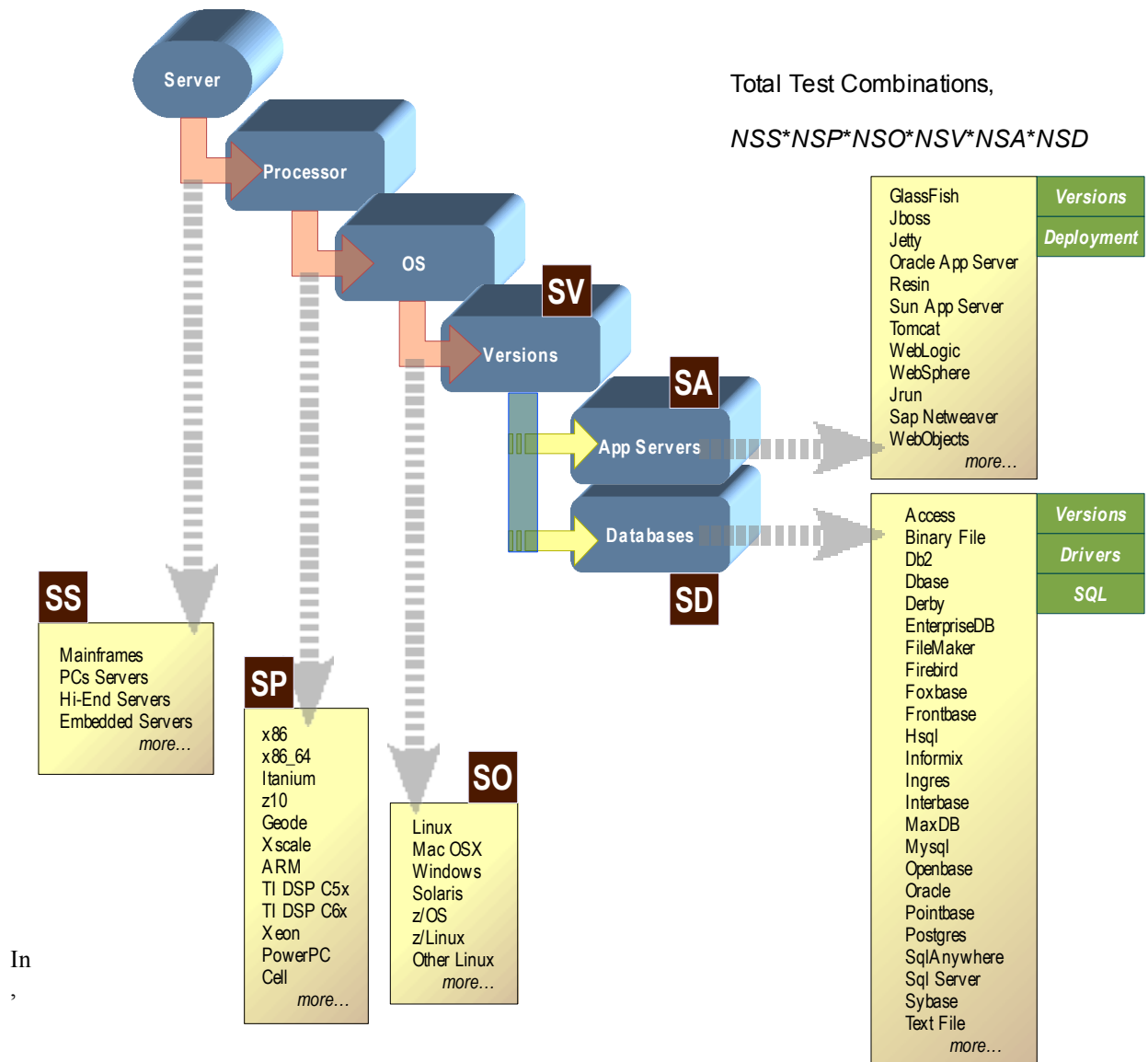
In Service Oriented Architecture the client consumes externally running standards based web services. These services are usually remote and run on a machine separated from where the client is located. By “true” SOA, it is meant that the client machines can be practically any machines that can bind to the service and use that service. In addition, the server may also work on multiple operating environments where the operating environment is defined as a combination of the following: operating systems, application servers, client execution environment, programming languages, mobile devices, embedded systems, mainframes, security environment, databases and standards compliance. The testing of this large distributed system is a significant task and a solid SOA test plan is needed to perform the tests, smoothly.

For an example, how complex the SOA test task can become, consider the following facts. Lets consider SOALIB services – a true heterogeneous web service implementation which runs on all platforms, from embedded systems to mainframes. For this heterogeneous SOA test, the following test categories are required:

1. Exclusive Server Testing
2. Exclusive Client Testing
3. Exclusive Integration Testing

### 1.1.1 Exclusive Server Testing

In this category, testing is designed, written and performed on the server. No clients are involved in this testing. As shown in Figure 1, for each server, there is more details on what constitutes the server testing complexity. As shown in the figure, the server has a processor, on which runs an Operating System. Operating Systems and the server software are all version based. If Java is used, the right Java version is required as well. So, versions mean the version of every component that may affect the testing. A Java Application Server hosted web service, which is almost always used, must also be tested on all of the most popular Application Servers. Also, if the web service works with databases, there are also a wide variety of databases available for the particular server. Application Servers also have other variable components like its version and deployment testing. The same applies to databases for each new version of a database adds or removes some features, to ensure compatibility. For databases, additional complexities include the SQL query engine, which usually has some proprietary commands, functions, procedures, triggers etc - are added. Each database must have a database driver. Every database vendor designed their driver differently and in 90% of the cases, most of these JDBC or ODBC drivers are not fully implemented or fully standards compliant. The fact is that the JDBC specification tends to be getting more and more complex up to the point that maintaining standards would require significant additional resources. However, in the majority of cases, these drivers do contain the necessary classes and methods needed to operate with that database. If an ODBC driver is used, it is also always platform specific, and therefore adds even more complexity.



*Figure 1: Exclusive Server Testing.*

the small dark brown boxes indicate multiple test patterns for that category. For example, SS is the Server test pattern for each server. Which indicates the number of various server candidates for testing. NSS in the equation indicates the number of such candidates (or patterns) for the entire server test. As an example, test server candidates may be mainframe, PCs, embedded systems servers, etc.

The following table describes various test patterns in the picture above.

Test Attributes	Attribute Description
SS	Server test patterns. The number of server platform in which the test has to be performed. This is a broad category, for example it could be mainframe, embedded systems, or PC, etc.
SP	Server processor patterns. The number of various processors in each of the server pattern (SS). For each SS, there may be more than one SP. For example, if its a PC (i.e SS = PC), the PC processors may be Itanium, x86 family, Power PC family, z9/10 family, or even embedded processors.

SO	Server operating system patterns. For each selected SS and SP, if the server supports multiple operating systems, then it has to be tested for all of them. As an example, SS=PC, SP=x86, then obviously, SO will need to be both Windows and Linux etc.
SV	Server version pattern. In a broad sense this part is all combined versioning. So, it combines versions for SO and all other attributes following it. Examples may be SV=PC, Itanium, Linux 2.6 Kernel. With, versioning for the following SA and SD also considered.
SA	Server Application Server pattern. This covers a broad range of application servers as shown in the figure. Example, SA=WebSphere, Web Logic, Tomcat etc.
SD	Server Database pattern. Covers a broad range of databases connected using the required connectivity drivers.

### 1.1.2 Exclusive Client Testing

In this category, only the client is tested without the server. In such tests, the client code is tested against static data, which is fixed binary, text, messages, files, databases, etc in static form. The static data may be automatically generated by scripts to make sure that this data is compatible with the server. Client systems have added complexity following the fact that they have to bind to the web service using proxies. Proxies are, on the other hand, dependent on the WSDL documents. As new versions of the server is released, WSDL's change, therefore, requiring new proxies to be built. On the client side, security is also a separate required pattern that must also be taken into account. As an example, public private key encryption architecture is an area of security where key sharing and transfer is usually not standardized. In Java, it's done in a different way than in .NET or C. There are also many other security implementations and some of them are proprietary. Overall, it is essential to make the client side communicate securely with the server, hence security testing must be done separately. The equation for the test combination is similar to the server side testing with the exception that the Application Servers are replaced by Proxies and the Database is replaced by Security.

Test Attributes	Attribute Description
CC	Client test patterns. The number of client platforms in which the test has to be performed. This is a broad category, for example, mainframe, embedded systems, PC, or mobile systems.
CP	Client processor patterns. The number of various processors in each of the client pattern (CC). For each CC, there may be more than one CP. For example, if its a PC (i.e. CC = PC), the PC processors may be Itanium, x86 family, Power PC family, z9/10 family, or even an embedded processor.
CO	Client operating system patterns. For each selected CC and CP, if the client supports multiple operating systems, then it has to be tested on

	them. As an example, CC=PC, CP=x86, then obviously, CO can be both Windows and Linux. CO also considers various RTOS when an embedded system is the client.
CV	Client version pattern. In a broad sense this is part of combined versioning. So, it combines versions for CO and all other attributes following it. Examples may be CV=PC, Itanium, Linux 2.6 Kernel. But, conversioning for the two below - CX and CS should also be considered.
CX	Client Proxy patterns. A web service must support all prior and new proxies. If only the latest proxies are used, then all other existing clients have the latest client proxies. But, this is not usually the case for embedded systems, where updating software may be difficult.
CS	Client Security patterns. A given client must support one or more security patterns. For example, all clients must support HTTP. However, HTTPS is a common and secure protocol that should also be used. WS-Security testing should also be added, if it is supported.

### 1.2.1 Exclusive Integration Testing

Exclusive integration testing only deals with tests of client with the server running in the background. Testing using the static test patterns on the client are now fed from the client to the server. In an ideal situation, all the integration tests will pass if the individual exclusive client and server testing works successfully. However, if it fails, then problems may be either in the messaging layer in web services, stubs or proxies layer, conversioning errors, or other area. It is also possible that either the server code or even client code could have bugs, but since they are fully tested at this point before the integration is started, the priority should be given to other failures first before suspecting client or server bugs. Figure 3 Shows how integration tests are done. Integration tests are performed by clients on the server. Prior to integration testing, all clients must pass the client test and all servers must pass the server test. If there are failures, then integration tests must not be executed until the other two categories of tests are passed. In integration testing, the following tests are performed in the sequence presented:

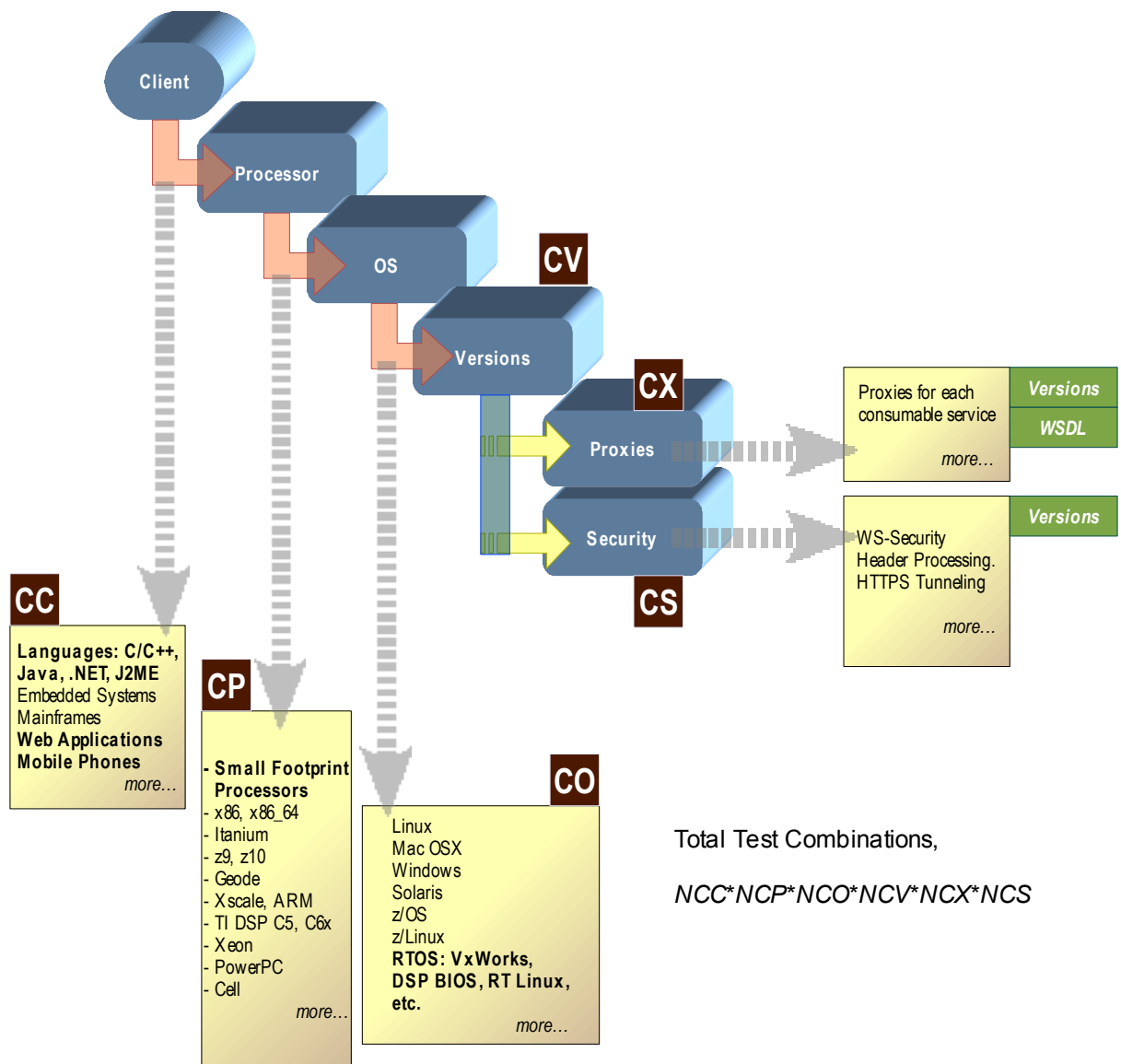


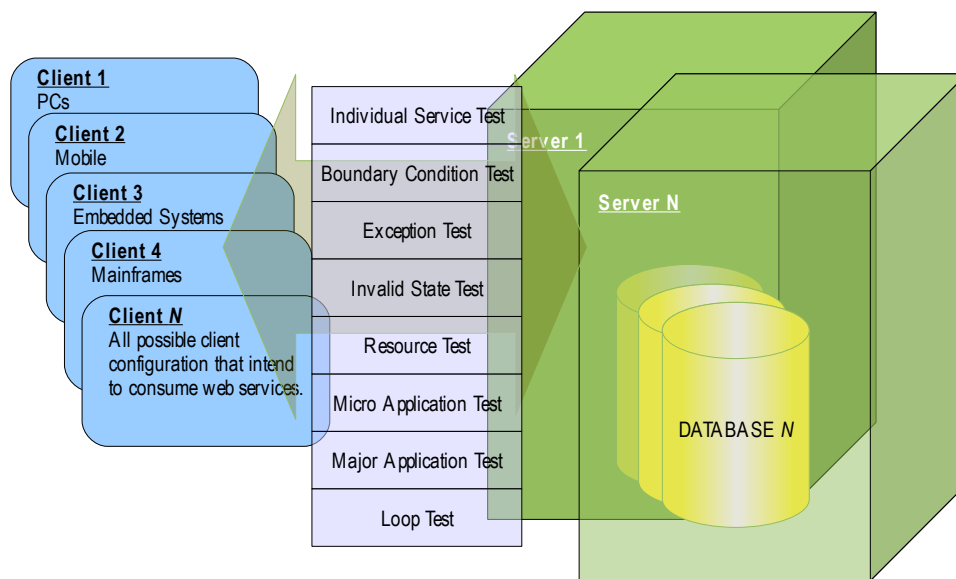
Figure 2: Exclusive Client Testing

1. Individual Service Test: Each service is called in some predefined order to make sure that the proxies are working correctly. It partially tests the bind and connectivity operation as well as ensuring that all methods are callable from the client. Each method is called with known values allowed by the operations and no invalid values are passed.
2. Boundary Condition Test: Boundary conditions test make sure that the service operations accept all the values within the boundary (or allowed values). This service passes all the boundary values, but no invalid values. There must also be a process to test that these boundary values were accepted as valid values and were not ignored.
3. Exception Test: Web Services throws various kind of exceptions under different conditions. If the conditions are known, these exceptions will be thrown. These tests create the conditions so that the exceptions are generated.
4. Invalid State Test: These tests attempts to create an invalid state in the web service. These tests are dependent on the web service and may not be applicable to all web services. As an example, if an attempt is made to update a table that used to exist, but some other process has deleted the table, an invalid state will occur suddenly in the web service. In this situation, a proper exception should be thrown and the service should exit without further processing. If the service continues to operate and attempts to operate on the

dropped table would be illogical. This kind of test ensures that if abrupt change occurs in state of the resources, the web service can detect it correctly and perform the proper operation.

5. **Resource Test:** Web Services are resource intensive when the instances of the service exists during the time the client consumes the service. Proper testing must be devised to make sure that these allocated resources are freed once the client closes the service. In data caching, data is cached during the period of service consumption. In some cases, the cache remains even after the client has closed the service. But if it is designed to clear this cache, this has to be tested to make sure that these resources were cleared.
6. **Micro Application Test:** Micro application test is a small utility application created using all the services in the web service. Running this application ensures that the web services can actually work together.
7. **Major Application Test:** Major application testings are written based on the web services. These may be applications in production that make sure that existing versions of applications continue to work with new versions. In many cases, these test could be manual testing if GUI is involved.
8. **Loop Test:** Loop test is a powerful and important test to make sure that system does not run out of resource when a service is called repeatedly in a loop. It is possible that if an infinite loop is created on a method on a service, the service will take a huge amount of processor time as well as jam the network. Sometimes, this operation is allowed and in other cases, the operation has to be brought to attention, by way of an alarm. This kind of operation will slow down the web service. This test should make sure that the web service detects a loop and throws exception or inform the operation to the administrator.

Many other additional tests may be performed based in addition to the testing listed above to make sure that all services are tested to the fullest and maximum extent. SOA testing can be very time consuming. In integration testing, clients and servers are chosen at random as the total number of test become unusually large. By using standardized random sampling, it is made overall statistically acceptable that testings have reached the required level.



*Figure 3: Exclusive Integration Testing*

### 1.2.2 Complete SOA Testing

Complete SOA testing takes a large number of actual tests. All three categories of testing must pass. We will check the extensiveness of each kind of testing. In a typical server test, we have the following typical numbers. In addition, there is the fourth category named Load Testing.

SS	SP	SO	SV	SA	SD	Total
PC Mainframe Embedded Systems	Itanium x86 Series Power PC Sparc	Linux Windows Solaris	We will only deal with latest version	All application servers listed in Figure 1	Add Database listed in Figure 1	
3	4	3	1	12	24	10368

This tells us that a full server test per unit test will require 10368 tests to be performed for the given server configurations. The word “per unit test” means that to test performance of only a single service. A service is composed of operations, which are simply remote function calls to the server object. A server test must be designed such that each service is tested as an unit and covers all the tests needed to test the entire service, which includes testing all the operations within that service.

CC	CP	CO	CV	CX	CS	Total
PC Mainframe Embedded Systems Mobile Phones	Itanium x86 Series Power PC Sparc Xscale Geode DSP 5x DSP 6x	Linux Windows Solaris VxWorks RT Linux DSP BIOS	We will only deal with latest version	Each proxy represents a service.	With and without WS- Security	
3	8	6	1	21	2	6048

The above table shows that we will need to perform 6048 tests per client. A typical client test is composed of more diverse operating environment than the server. Because, most clients are designed to consume web services, not host them. But, client code is smaller and simpler, which make it easier to test the clients, from a volume perspective. If the client test simply requires calling the web service method, then it is not required to perform a client test for that particular service at all, as the integration testing will determine that functionality.

The third functionality is to check the integration testing. Integration testing is usually exhaustive in nature as it tries to crash the client and server by passing invalid values in addition to pass valid values. However, by this time, both client and server tests are fully functional, and these rigorous tests will not result in bugs in the client and the server. Instead, if any bugs are found they will be related to messaging, proxies, versions, deployments and other external matters which are related to other system related issues. For example, if a rigorous test results in failure of connection timeout, then it is possible that the server has timed out before any message was returned. In our server and client test, we may see the total number of tests required to test the SOA 100%:

Server Tests	Client Tests	Total
Total Number of Server Tests	Total Number of Client Tests	Total Number of Integration Tests
10368	6048	62,705,664

The total number of tests need to test the web service fully against all clients and servers is actually huge. Fortunately, doing absolutely all these tests will not be necessary. Only a selected number of client and server could be used to do integration tests which ensures the likelihood of 90% or more probability that all of the remaining clients and servers will work without a problem.

## **1.2 Test Instances**

As explained in Section Complete SOA Testing, the client and server tests may be tested independently using many different approaches. One of the best and effective ways is to use Virtual Machines for each Operating Systems, as they are available nowadays in many instances. Some possible VM configurations are:

1. Quad or Octal Core processors to a minimum.
2. Minimum of 2GB per VM
3. Each VM must be fully loaded
4. All VM tested at the same time

All Virtual Machines must be fully loaded. VMs may contain both clients, servers, databases, and other resources. However, they don't cover all tests. For example, all embedded systems would require true embedded hardware. Similarly, mobile testing would require real cellphones. Mainframe tests must be done on real mainframes. Uncommon servers like AS400, VMS, etc would require respective servers in place for testing. This leads to the cost of testing to become time consuming and expensive. But there is no alternative. SOA is a technology which integrated everything and there is no short cut to make sure that it all works everywhere and anywhere.

The cost of true SOA testing should not be overlooked. The testing cost determines in large part the price of the SOA solution. Therefore, a test platform should not be adopted if there is no requirement for it. For example, if there is no market for the SOA to be working on Open VMS, then the test cost of running in Open VMS can be eliminated by not including Open VMS in the test architecture. By eliminating each server, processor, version etc. as shown in Figure 1, Figure 2 and Figure 3 above, the number of test instances to be executed can be rapidly reduced.

Virtualization reduces the cost of testing very rapidly. By using a single large core large memory server instead of using multiple smaller server is a better approach and saves test cost by saving power, maintenance and virtualization.

Lets put this in an example. For a 8 core processor with 16GB memory (assuming there is plenty of hard drive space), with 2GB per VM, we can have each processor dedicated to each VM. The testing will be fast and can be run simultaneously. If three such servers are used, one for client, one for server and one for integration testing, all these three category tests could be running at the same time making not only the test time reduced, but also reducing the total cost of testing. However, when it comes to processor type, it is definitely required to have a server with the given processor. For example, a 8 core Intel x86\_64 processor can server the Intel x86 processor, but cannot server the purpose of Intel Itanium processor. So a separate Intel Itanium processor would be required. Some Operating Systems are proprietary, like Mac OS X, which requires a server from the same vendor. Similarly, there is no alternative to mainframe testing.

Overall, testing a true SOA is still manageable and affordable by proper test planning. SOA testing can be a good size investment and a good testing plan must exist first before any test project

is undertaken.

## **1.3 Conclusion**

SOA Testing is a challenge of all the SOA architectures. A good SOA web service is the work of hundreds of hours of successful testing. All SOA implementations must be robust, otherwise, the power of SOA is not fully realized. A service should never quit working at any time. A SOA system should never leak resource. A true SOA application must be running at all times without running out of resources. A good SOA implementation must also be predictable, that means, it must process a service within a finite amount of time. All these are part of the design, and therefore, a lot of thought process goes into designing a SOA. A good design leads to faster testing of services as these services become more modular.

There are other white papers available from Soalib, Inc on SOA design as well.

## **1.4 About**

### **1.4.1 About Soalib, Inc**

Soalib Inc. has developed a suite of SOA libraries that allow companies to develop SOA applications more efficiently, more effectively and over a much shorter time-frame than any other SOA technology. These pre-built libraries are designed to work across a wide range of operating systems, application servers, client environments, programming languages, mobile devices, security environments and databases, collectively known as Operating Environments. Soalib products have been developed with a totally platform agnostic approach for the true integration of multiple languages, diverse technologies, disparate databases and heterogeneous legacy platforms.

Soalib, Inc has many other white papers focused on this SOA topic in much more detail. Due to various complexities involved in service design, SOA development cannot be taken lightly. Soalib's SOALIB pre-built services products, completely eliminates the development and test period needed to duplicate these functionality from scratch. Please contact Soalib for more information on how to best design and develop your SOA services in the most cost effective fashion.

### **1.4.2 About the Author**

Soalib is the brainchild of Ejaz Jamil, the Chief Architect of Soalib Incorporated. Since 2004, Ejaz and his team have developed SOA services with a focus on building tested libraries which work from embedded systems to mainframes. Ejaz brings years of embedded systems experience with Texas Instruments, where he worked as an Embedded System Engineer and Mathworks, Inc, the Natick based simulation software company, where he developed complex products for the embedded systems industry. Back in 2003 Ejaz understood that the True power of SOA comes from the integration of real-time embedded systems devices with all other technology platforms. Soalib's core product, SOALIB® solves this heterogeneous integration problem for companies embarking on global projects using standards based SOA.

Ejaz Jamil can be reached at [soalib@soalib.com](mailto:soalib@soalib.com) or via telephone +1-508-460-1116.

## **1.5 Glossary**

*Virtual Machine  
Virtualization*

A mixed hardware and software process that consume the same hardware resources in a single server such that none of the hardware resources (processor, memory, hard drive) used by one process affect other processes. Each process remains completely independent to all others.

<i>Client Machines</i>	Referred to as machines where data is collected. Typical machines are PCs and applications, mobile phones or devices, embedded systems devices, enterprise applications and others that collect data.
<i>Data Hub</i>	A generic term to indicate where all data is collected from the clients. Usually a very large database management system.
<i>JDBC</i>	Java Database Connectivity, the database driver technology for Java programming language.
<i>JVM</i>	Java Virtual Machine
<i>ODBC</i>	Open Database Connectivity, the database driver technology used to access from traditional programming languages like C/C++/VB, etc.
<i>Operating Environment</i>	Operating environment as all combinations of the following: operating systems, application servers, client execution environment, programming languages, mobile devices, embedded systems, mainframes, security environment, databases and standards compliance.
<i>SOA</i>	Service Oriented Architecture, an architecture that uses standards based protocols to access and consume remote application services over the network.
<i>SOALIB®</i>	Service Oriented Architecture LIBraries, a core product of the company. A set of SOA compliant libraries that integrates embedded systems to mainframes.
<i>SOAP</i>	Simple Object Access Protocol, the most common XML based interoperable protocol used for communication to and from web services.
<i>WSDL</i>	Web Services Description Language, an XML document which describes the messaging format for each of the exposed services in a web service.