Abstract

Today's dynamic business needs are shaping the modern IT industry. Lower costs and faster time to market have propelled forward market-changing phenomenon like SOA, internet business models and cloud computing. Further, changes in business needs have led to changes in system architectures, from monolithic systems and slow batch processes to SOA and real-time processes. These have resulted in businesses asking a new set of questions to test managers, consultants and architects, like how does one test these modern SOA systems? This paper attempts to answer the questions surrounding modern SOA testing.
SOA makes promises that include increased agility, larger application lifecycles, better integration and lower costs. However, in reality these promises are difficult to deliver. Building complex integrated systems is not a simple task but takes a combination of sophisticated tools, customized methodologies and a lot of creative approaches to correctly implement, test and deliver the SOA systems.

SOA testing is a combination of service testing, process verification, Test Data Management (TDM) and accelerated UI automation. It also includes enabling practices such as continuous integration testing and service virtualization. Testing teams need to test the systems at the service provider and the client end to ensure an error free execution of systems. Tests also need to be grouped correctly into a regression suite. Enabling regression suites with a workflow based data provisioning system is a key factor that we shall examine later in this paper.

Starting Our Journey With A Reality Check

Many of us have been introduced to SOA via webinars, articles and books published by SOA vendors. The promises made by the SOA evangelists seem easy to achieve and are every CTO's dream. But it is during the implementation phase that reality surfaces. Each SOA promise is in reality an enormous challenge. Let us have a quick look at some of the common challenges we have come across, while implementing SOA, over the recent past:

Figure 1: “Typical SOA Implementation Challenges”

Some of the challenges SOA brings for the testing teams are that they have to learn new technologies, processes and new tools that are being used. Many testing teams often check if they can continue to use classical tools and processes in the new SOA environment. Once they begin SOA testing, they quickly realize that this can't be done. Let's look at some of the reasons and questions that lead to this realization:
The SOA Testing Road Map

Every SOA project can be classified into four distinct phases - Requirements Management, Analysis and Design, Development (and Testing) and Deployment & Production Management. Most SOA vendors provide repositories and BPM/BAM systems that support deployment and production management phase. These phases are managed by dedicated SOA teams and are not focus areas for testers. CIT and SIT, which occur at the design, development and testing phase of the SOA lifecycle, hold a lot of interest for SOA Testers.

The diagram below is an illustration of the different phases of the Core SOA Solution:

*Figure 2: “Different Phases of the Core SOA Solution”*
Let's take a closer look at the Design and Develop phase to understand what it takes to implement SOA Testing correctly.

**Initial Steps of the SOA Testing Process**

SOA Testing focuses on three system layers:

1. **Services Layer**
   - Includes services exposed by systems which are derived from business functions.

2. **Process Layers**
   - Contain implementation aspects of business processes. They mostly exist as composite services, choreographies and process models (orchestrations). These processes might also be embedded in user interfaces, ETL tools or domain models. We would focus on processes that are implemented in user interfaces and process layers.

3. **Service Consumers**
   - These can be other services or user interface components.

The first step in the SOA Testing process is validation of services for functional and nonfunctional requirements.

Next, we validate the processes implemented within the process layers (this is synonymous with integration testing for providers). This validation of services and process components results in a certified provider layer.

The subsequent step is to test service consumers. UI automation is a possibility and is encouraged for improved regression testing. Service consumer testing ensures that the entire system stack works as expected. SOA brings with it huge regression needs, because of which the tests are rolled into regression suites, to shield them from breaking changes.

This process changes slightly for systems with no separate process tiers. In such cases, processes are embedded inside the user interfaces and domain models. The key in such cases is to test services and then automate consumer testing for validating processes and the system stack. This approach is shown in Figure 3 below:

*Figure 3: “SOA Testing – The Four Step Approach”*
Challenges in SOA Testing Processes Implementation

SOA testing processes are hard to implement because of the need for complex supporting work streams. For example, regression tests can only run correctly if systems exposing services have the right data. The capabilities of test data traceability and workflow based data provisioning are key prerequisites for running regression suites. Another suitable example is the process for accelerating UI automation. Many processes here have repetitive steps (e.g., login or logout) that require a lot of effort. When you repeat these steps across test cases then changing them in an agile SOA environment can be quite challenging. Another challenge in the process is the need to test services that have per-use costs or inaccessible systems.

Let us look at some aspects of supporting processes below:

1. **Providing workflow based data provisioning and catering to data integrity, data volatility, federated data integrity and data masking**: The practitioner’s answers lie in TDM solutions or in Systems Virtualization.
2. **Accelerating Automation**: Select tools or develop frameworks that generate UI automation scripts and centralize repetitive activities. An example here would be the Test Automation Accelerator (ITAA) from Infosys.
3. **Access Constrained Systems**: Use service virtualization to alleviate services that have per use costs or are not available for testing.
4. **Continuous Integration Testing Mandates**: Use testing tools that are capable of monitoring changes in your repositories. They must run regression suites to avoid breaking changes. Execute regression suites on all major build and releases for effective test coverage.

Below we cover some other aspects of the SOA Testing processes:

1. **Security Mandates**: Use security testing tools with a matching process that focuses on grey box and black box testing. Remember to use penetration testing tools and look for common roadblocks such as injection (script) and denial of service. In some application architectures, testing services for security is vital in order to check the weak links in the security chain.
2. **Performance**: Services and process models can be obstacles to performance. Sometimes even patches to underlying infrastructure such as ESBs can affect performance. Other likely performance bottlenecks include services that transform data (especially outside transformation containers) and services that are synchronously invoked from consuming applications (especially where performance needs are critical enough to bypass ESBs).
3. **Ensuring Compliance to Governance Mandates**: Governance mandates include ensuring compliance to policies such as PCI-DSS (security) or WS-I Basic Profile (interoperability). Your best option is to work with a schema assertion tool for these needs and utilize your ESB/repository vendor’s product stack completely. Your SOA testing tool might support these needs as well.

The SOA Test Automation Process

The greatest ROI from SOA testing comes from the automation benefits. This is true because regression testing is crucial to the essentially agile and the ever-changing SOA world. Your choice of tools will be determined by your needs. These needs vary from simple open source tools to bleeding edge commercial stacks. Your approach however will almost always be the same and is shown in the illustration (Figure 4):
Figure 4: “A Simplified View of Test Automation for SOA”

Figure 4 is a simplified view of test automation for SOA (excluding TDM, end-to-end testing with UI automation, etc) and targets very generic SOA systems. The main steps here are:

1. Testers use an automation tool to invoke services or to publish and subscribe to a broker. Services can be REST/P0X/JSON/SOAP exposed over multiple transport protocols like HTTP, FTP, and JMS etc. Brokering is mostly provided by ESBs such as from IBM, TIBCO etc.
2. Testers use a variety of validation techniques. Content can be identified by regular expressions, XPath to customized script (Java). Validation rules can range from plain arithmetic (<, >, =) to customized complex Java.
3. Testers have a variety of options for parameterization.
4. Test Cases are rolled into regression suites.
5. Suites can be executed on schedule or triggered to execute on changes to repository.
6. NFRs can play a major role as most testing tools allow functional tests to be rolled into performance suites. Some tools also include security testing features.
Conclusion

SOA testing has led to improved testing approaches and a new generation of specialized tools. SOA testers need to now learn larger processes, tools and expand their skill sets to encompass a wider range of practices. These include UI automation, TDM, service virtualization, security and performance. Success in SOA implementations will come to QA organizations that leverage expertise in selecting the right tools and supporting processes. This move will ensure that the next generation of systems will deliver on the promises that meet the business user's expectations.

About the Author

Gaurish Hattangadi, Test Consultant, Infosys Limited.

Gaurish's focus areas include SOA testing and enterprise architecture. He can be reached at Gaurish_Hattangadi@infosys.com.