Torry Harris

SOA

SOA Test Methodology

Abstract

Service-Oriented Architecture (SOA) promises significant benefits to today's organizations. Successfully delivering SOA benefits, especially Business Agility and Component Reuse, will be dependant 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

Table of Contents

1. 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, 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 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.

1.1 Service-Oriented Architecture Overview

1.1.1 What is SOA

A concise functional definition is provided below:

Oasis (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."

1.1.2 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

1.1.3 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.

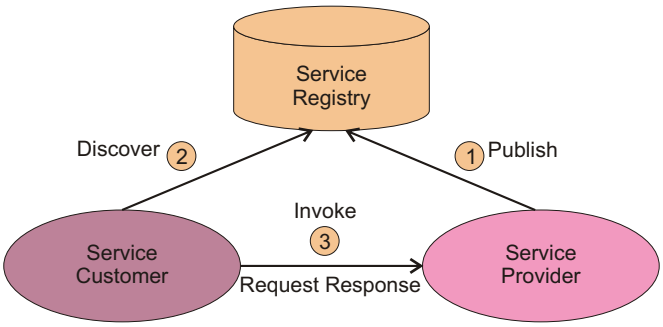

Figure 1.1 SOA Components

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.

1.1.4 Test Model

Testing activities such as design, analysis, planning and execution must happen throughout the entire SOA project life cycle. The V-Model is a suitable test methodology 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.

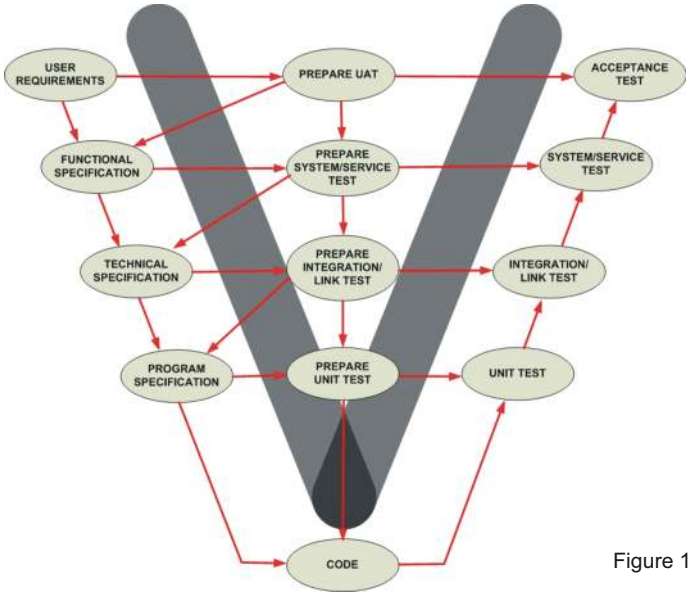

Figure 1.2 V-Model

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.

1.1.5 SOA Governance

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

2. SOA Test Methodology

2.1 Traditional Test Approach

Traditional 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.

2.1.1 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.

2.2 SOA Test Approach

2.2.1 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!

2.2.2 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 interdependencies and a SOA testing approach must follow the same pattern.

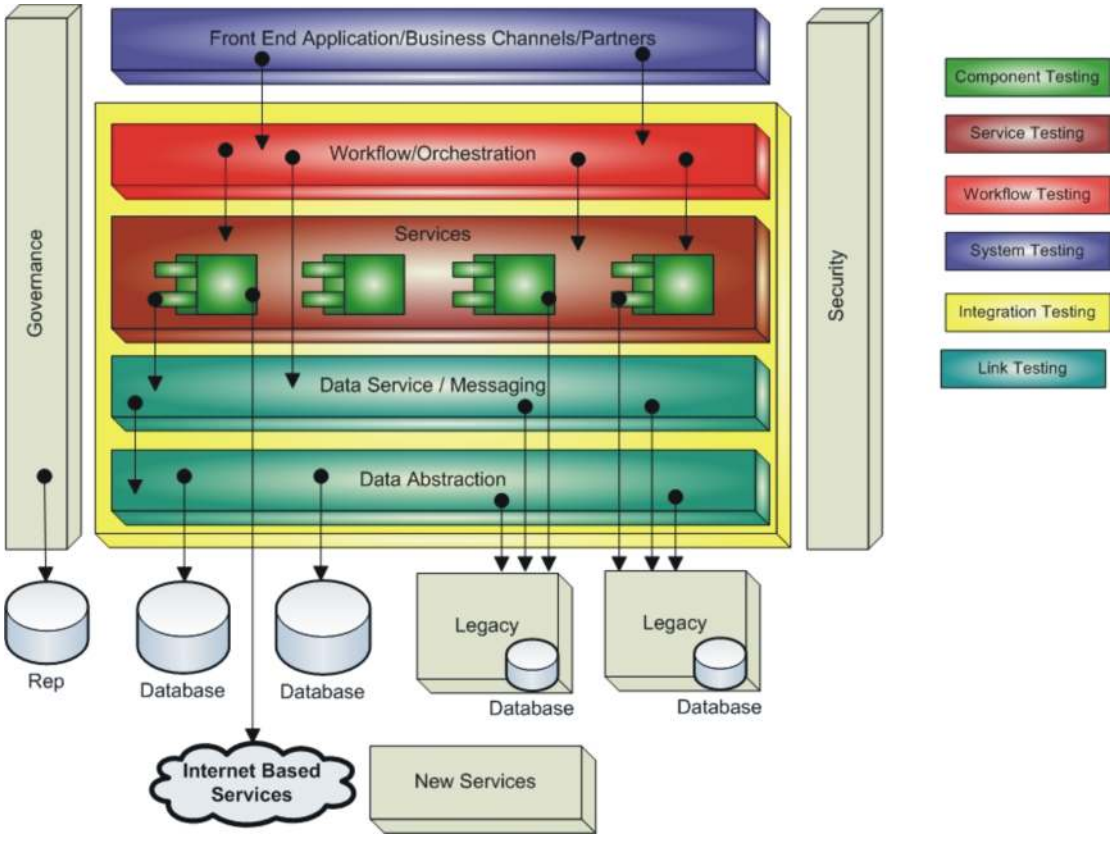

Figure 2.1 SOA components

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

2.2.3 Governance Testing

SOA Governance is a key factor in the success of any 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.

2.2.4 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

2.2.5 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.

2.2.6 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.

2.2.7 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).

2.2.8 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.

2.2.9 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.

2.3 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

2.3.1 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 Requirements, Technical Design & Governance Standards and Policies	✓	✓	-	-	✓	✓

TEST PHASES	Guiding Documents	Functional	Performance	Interoperability	Backward Compatibility	Compliance	Security
Integration & Orchestration Testing Phase	Business Requirements, Technical Design & Governance Standards and Policies	✓	-	✓	✓	-	✓
System/ Process (User Acceptance) Testing Phase	Business Requirements, Technical Design & Governance Standards and Policies	✓	✓	✓	✓	✓	✓

2.3.2 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.

2.3.3 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.

2.3.4 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.

2.3.5 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.

2.3.6 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.

2.3.7 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.

3 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.

4 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.

5 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.

6 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.

7 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.

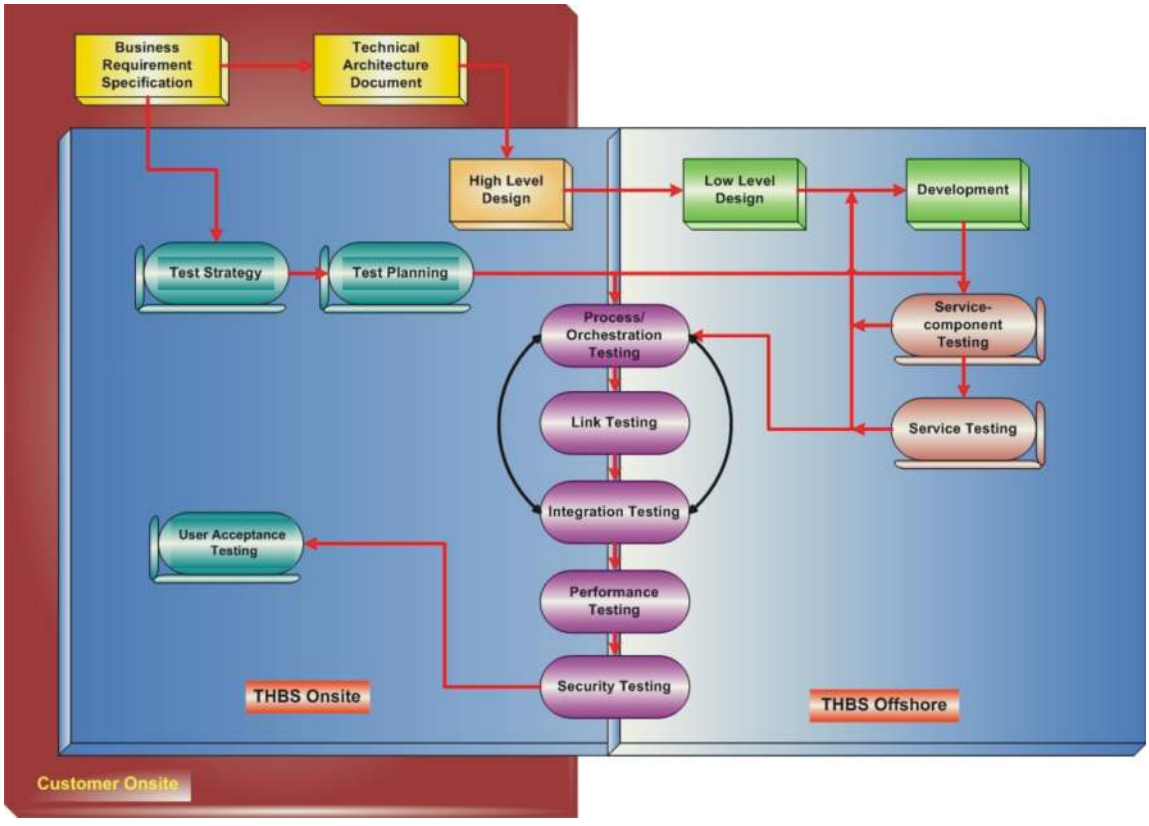

Figure 8.1 Onsite-Offshore Test Model

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.

8 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.

8.1 Commercial Products

8.1.1 Green Hat GH Tester

This 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.

8.1.2 Mercury

Mercury 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.

8.1.3 Parasoft SOAtest

This 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.

8.1.4 AdventNet QEngine

This is a complete Web-based test automation tool 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).

8.1.5 Borland SilkPerfomer SOA edition

This is an automated Web Services test tool that supports functional and performance testing of services and interoperability functional and performance testing between services.

8.1.6 LISA WS – Testing

LISA WS-Testing 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.

8.2 Open Source Products

8.2.1 SOAP UI

SoapUI 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.

8.2.2 PushToTest TESTMAKER

TestMaker is an open-source utility and framework to build and use intelligent test agents to check Web-enabled 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.

9 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.

Torry Harris Business Solutions Inc, a US based services provider with a large base of technologists located in the UK, India and China has provided cost effective solutions at a design, development and support level to a variety of enterprise clients across the world since 1998. The company specializes in integration, distributed computing, and its focus on SOA is a result of nearly a decade of expertise gathered in the middleware space. The company has partnerships with almost all the leading SOA and integration product vendors. SOA, involving the creation of autonomous parts of a solution, lends itself admirably to the cost effective model of offshore service collaboration. A separate white paper entitled "SOA Implementation with an offshore partner" available for download, explores this model in a more detailed manner. Further information about the company and a variety of white papers on SOA are available at www.thbs.com/soa.

For more information, write to us at soa@thbs.com.