TESTING IN SOA ENVIRONMENT

SEWP ZG629T DISSERTATION

By

PUNEETH G. RAO
2010HW70293

Dissertation Work carried out at
Wipro Technologies, Bangalore

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
Pilani (Rajasthan) India
October, 2014

SEWP ZG629T DISSERTATION

TESTING IN SOA ENVIRONMENT

Submitted in partial fulfilment of the requirements of
M.S. Software Engineering Degree Program

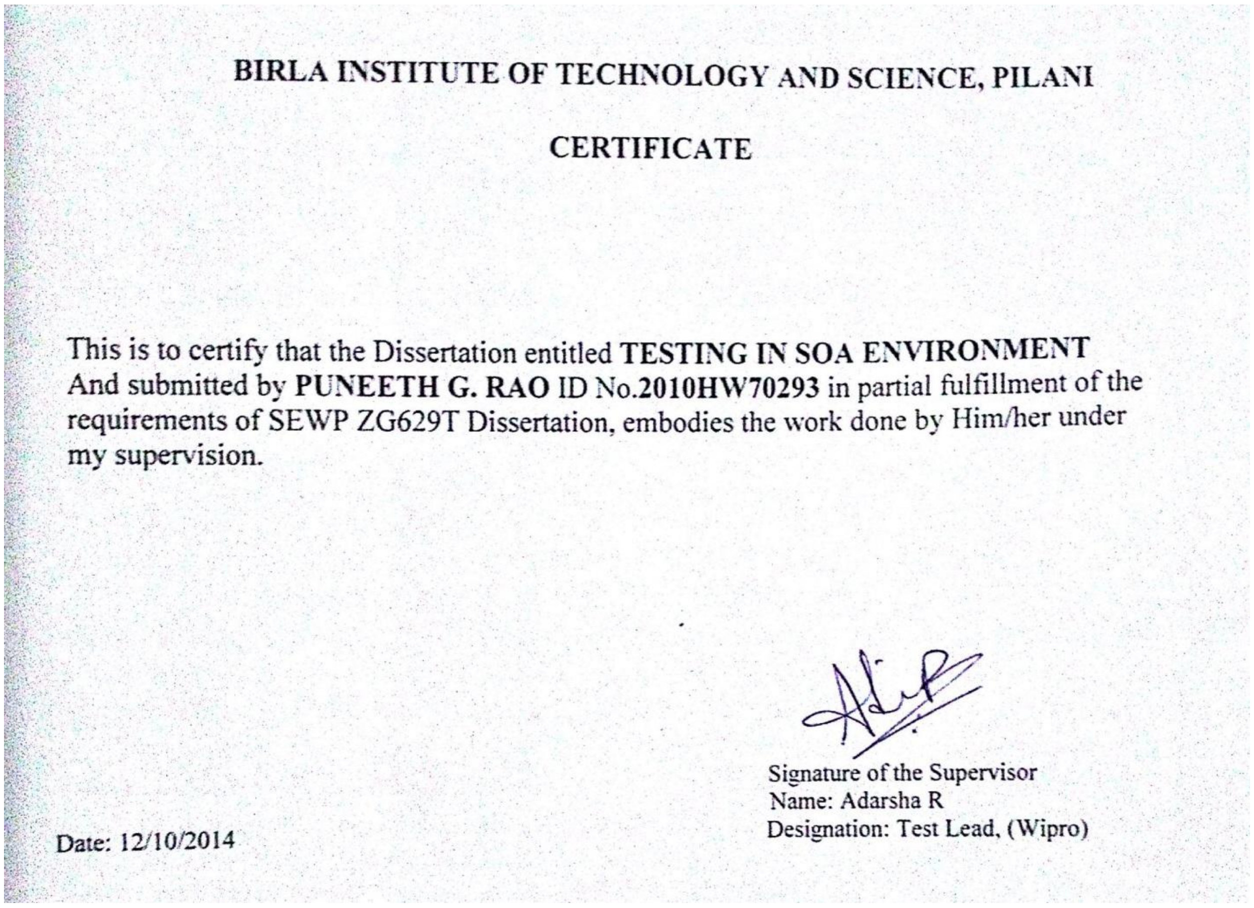
By

PUNEETH G. RAO
2010HW70293

Under the supervision of
Adarsha R
Test Lead

Dissertation work carried out at
Wipro Technologies, Bangalore

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)
(October, 2014)



BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the Dissertation entitled **TESTING IN SOA ENVIRONMENT** And submitted by **PUNEETH G. RAO** ID No.**2010HW70293** in partial fulfillment of the requirements of SEWP ZG629T Dissertation, embodies the work done by Him/her under my supervision.

[illegible]

Signature of the Supervisor
Name: Adarsha R
Designation: Test Lead, (Wipro)

Date: 12/10/2014

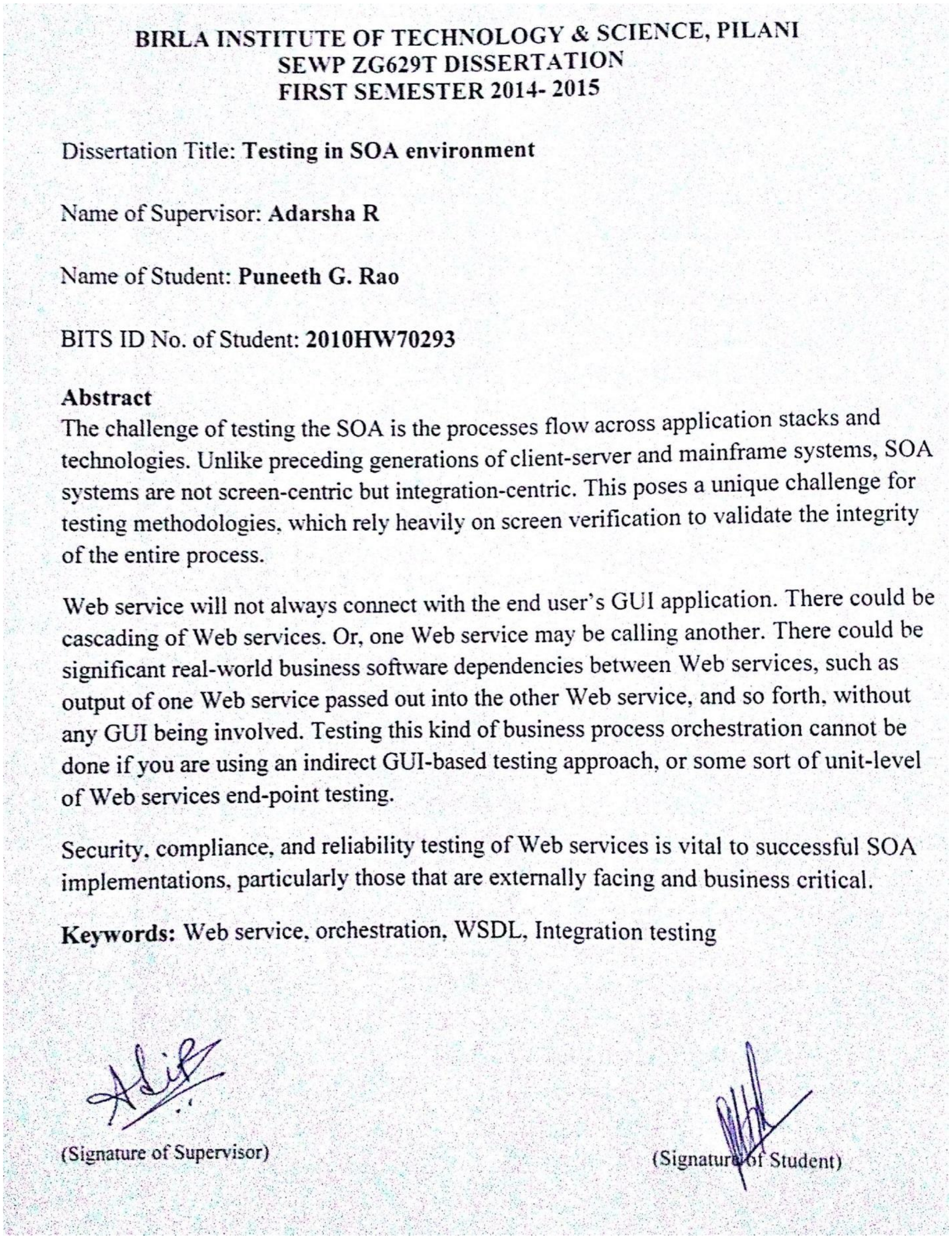
ACKNOWLEDGEMENTS

I would like to thank my project supervisor **Adarsha R** for his guidance, efforts and patience. He has been extremely supportive and co-operative in this whole effort. His valuable comments have been helpful in enhancing the quality of work.

I would also like to thank my senior colleagues, **Adarsh Lalage** and **Sri Vidhya Vijayanarayana** for the support and guidance extended to me during the dissertation. I also would like to thank **Narendra Shankarappa** for helping me in putting together this document and making me follow the standards of presenting the report.

I am also thankful to senior members of **SOA Testing Practice** and SPEZI, **Testing Centre of excellence** at Wipro for making available all the resources required for this dissertation.

My sincere regards and thanks to **BITS, Pilani WILP program**, **Prof. Ramesh** and **WASE department** for the information and support.



BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
SEWP ZG629T DISSERTATION
FIRST SEMESTER 2014- 2015

Dissertation Title: **Testing in SOA environment**

Name of Supervisor: **Adarsha R**

Name of Student: **Puneeth G. Rao**

BITS ID No. of Student: **2010HW70293**

Abstract
The challenge of testing the SOA is the processes flow across application stacks and technologies. Unlike preceding generations of client-server and mainframe systems, SOA systems are not screen-centric but integration-centric. This poses a unique challenge for testing methodologies, which rely heavily on screen verification to validate the integrity of the entire process.

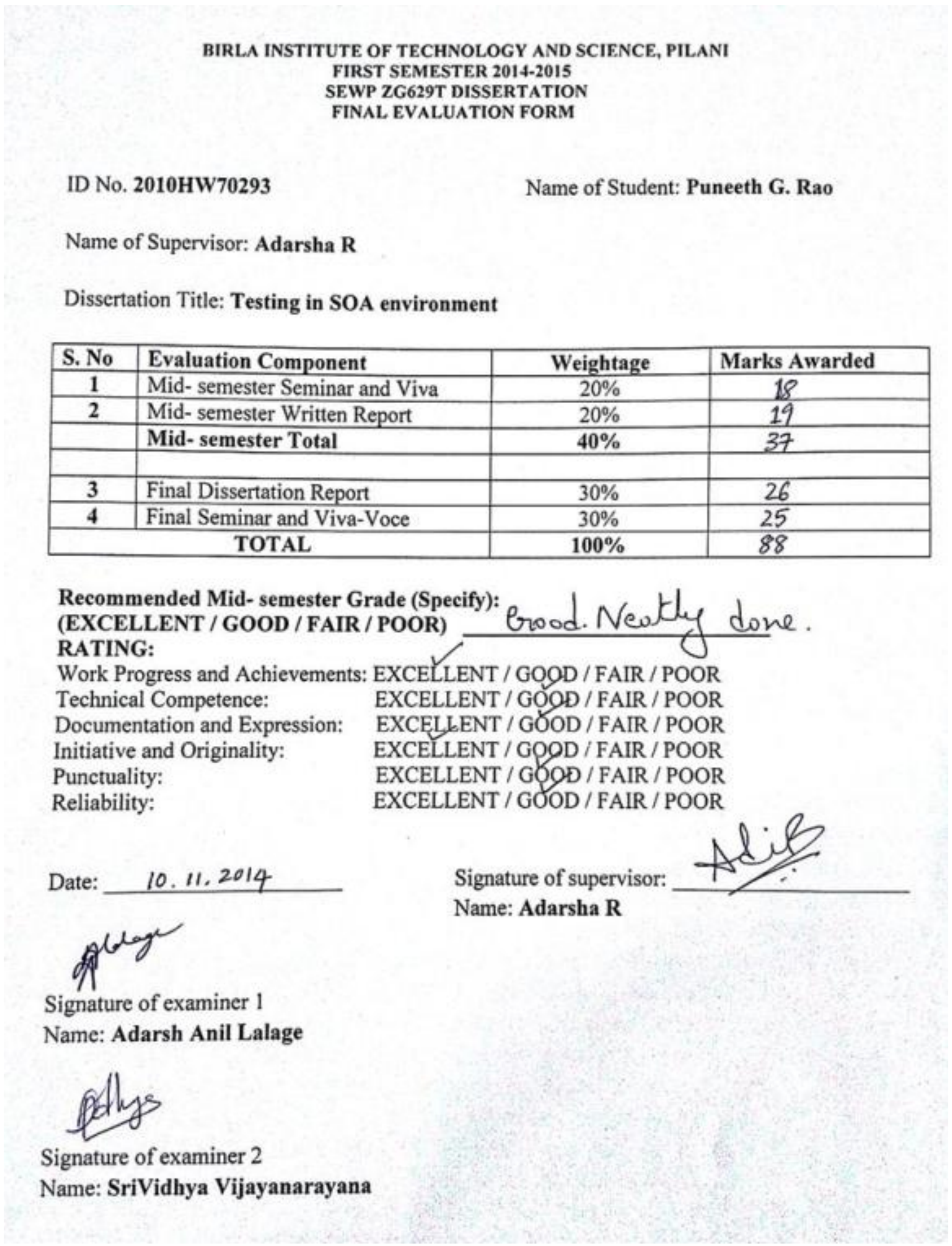
Web service will not always connect with the end user's GUI application. There could be cascading of Web services. Or, one Web service may be calling another. There could be significant real-world business software dependencies between Web services, such as output of one Web service passed out into the other Web service, and so forth, without any GUI being involved. Testing this kind of business process orchestration cannot be done if you are using an indirect GUI-based testing approach, or some sort of unit-level of Web services end-point testing.

Security, compliance, and reliability testing of Web services is vital to successful SOA implementations, particularly those that are externally facing and business critical.

Keywords: Web service, orchestration, WSDL, Integration testing

[illegible]
(Signature of Supervisor)

[illegible]
(Signature of Student)



BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
FIRST SEMESTER 2014-2015
SEWP ZG629T DISSERTATION
FINAL EVALUATION FORM

ID No. **2010HW70293**   Name of Student: **Puneeth G. Rao**

Name of Supervisor: **Adarsha R**

Dissertation Title: **Testing in SOA environment**

S. No	Evaluation Component	Weightage	Marks Awarded
1	Mid- semester Seminar and Viva	20%	18
2	Mid- semester Written Report	20%	19
	Mid- semester Total	**40%**	37
3	Final Dissertation Report	30%	26
4	Final Seminar and Viva-Voce	30%	25
TOTAL		**100%**	88

Recommended Mid- semester Grade (Specify):
(EXCELLENT / GOOD / FAIR / POOR) Good. Neatly done.
RATING:

Work Progress and Achievements:	[✓ EXCELLENT] / GOOD / FAIR / POOR
Technical Competence:	EXCELLENT / [✓ GOOD] / FAIR / POOR
Documentation and Expression:	EXCELLENT / [✓ GOOD] / FAIR / POOR
Initiative and Originality:	[✓ EXCELLENT] / GOOD / FAIR / POOR
Punctuality:	EXCELLENT / [✓ GOOD] / FAIR / POOR
Reliability:	EXCELLENT / [✓ GOOD] / FAIR / POOR

Date: 10. 11. 2014   Signature of supervisor: [illegible]
Name: **Adarsha R**

[illegible]
Signature of examiner 1
Name: **Adarsh Anil Lalage**

[illegible]
Signature of examiner 2
Name: **SriVidhya Vijayanarayana**

Contents

1.0 Introduction and nuances

To discuss the various challenges faced and methods followed by the QA teams testing the business based in SOA environment. The best tools, solutions and practices involved in the SOA testing specialization.

SOA – What and why?

- A Service Oriented Architecture (SOA) is architectural style for building business applications as a set of loosely coupled black box components orchestrated to deliver a well-defined level of service by linking together business processes.
- A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.
- If a service-oriented architecture is to be effective, we need a clear understanding of the term service. A service is a function that is well -defined, self-contained, and does not depend on the context or state of other services.
- Services are what you connect together using Web Services. A service is the endpoint of a connection. Also, a service has some type of underlying computer system that supports the connection offered. The combination of services - internal and external to an organization - makes a service-oriented architecture.
- The below figure illustrates a basic service -oriented architecture. It shows a service consumer at the right sending a service request message to a service provider at the left. The service provider returns a response message to the service consumer. The request and subsequent response connections are defined in some way that is understandable to both the service consumer and service provider. A service provider can also be a service consumer.

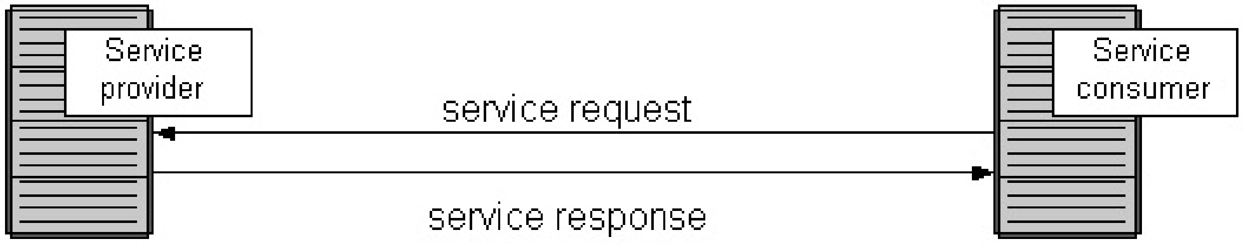


Fig 1.0

Some pointers before going further

SOAP (**Simple Object Access Protocol)** is XML based protocol for accessing a Web Service. It allows the applications to exchange information over **HTTP**, **SMTP** or **TCP**.

SOAP is Platform independent and language independent i.e. it provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

Extensible Markup Language, a new markup language published by the W3C to address the limitations of HTML

XML Schema describes the structure of an XML document. XML Schema language is also referred to as XML Schema Definition (**XSD**)

XML namespaces provide a way to distinguish deterministically between XML elements that have the same local name but are, in fact, from different vocabularies.

Web Services refers to the technologies that allow for making connections. Services are what you connect together using Web Services. Web Services can make your applications Web applications

WSDL (Web Services Description Language) is an XML -based language for describing Web services and how to access them. It specifies the location of the service and the operations (or methods) the service exposes.

Representational state transfer (**REST**) is a style of software architecture for distributed hypermedia systems such as the World Wide Web.

Rest promotes and recommends generic operations on resource and uses HTTP methods: PUT GET POST DELETE. It utilizes the caching mechanism

Web Services does not promote generic operations. The First generation only utilizes HTTP POST and has No caching capabilities

REST is over HTTP, but SOAP can be over any transport protocols such HTTP, FTP, STMP, JMS etc.

SOAP is using soap envelope, but REST is just XML.

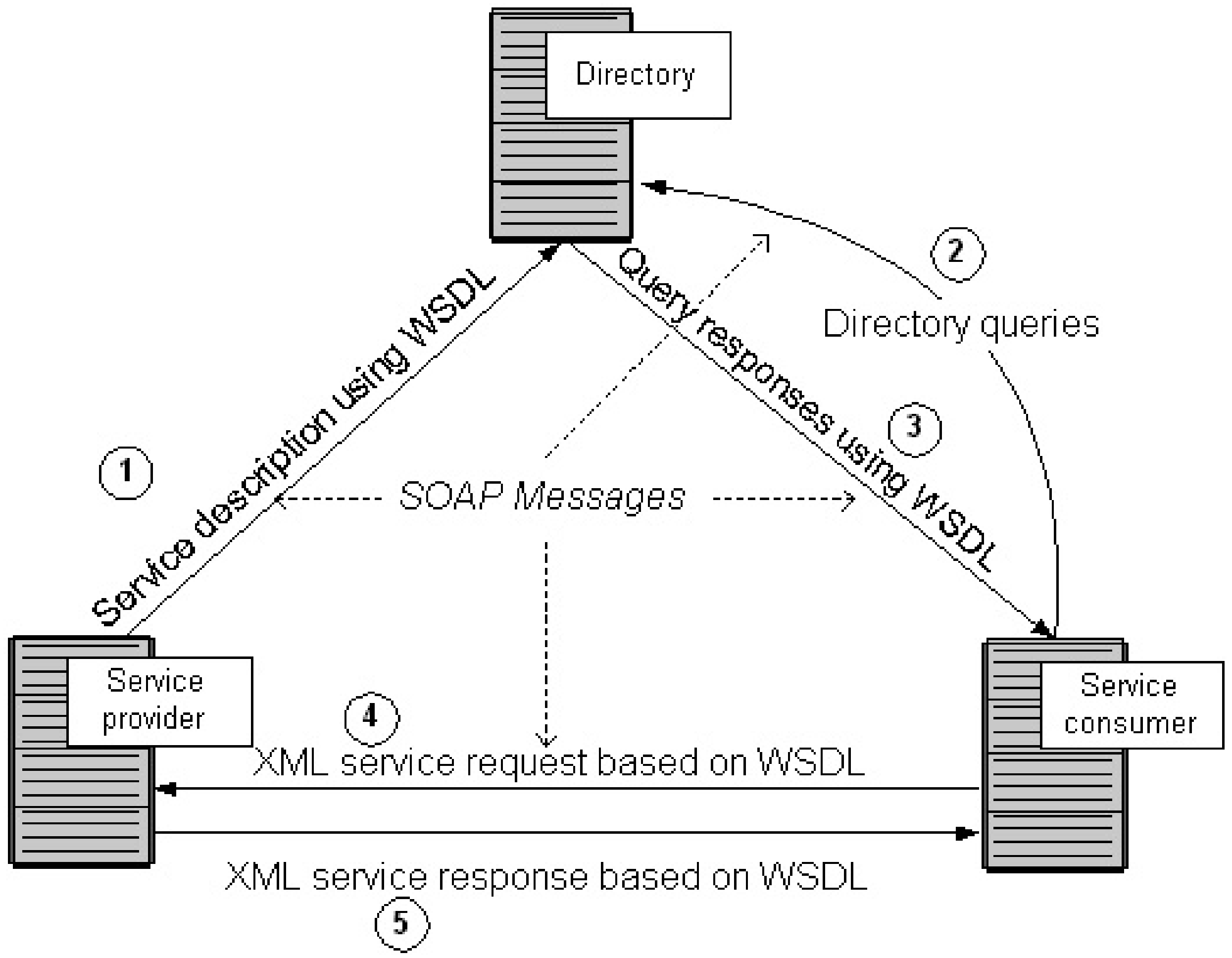

Directory
2
Directory queries
Query responses using WSDL
3
Service description using WSDL
1
SOAP Messages
Service provider
Service consumer
4
XML service request based on WSDL
XML service response based on WSDL
5

Fig 1.1

2.0 SOA – Conceptual Architecture

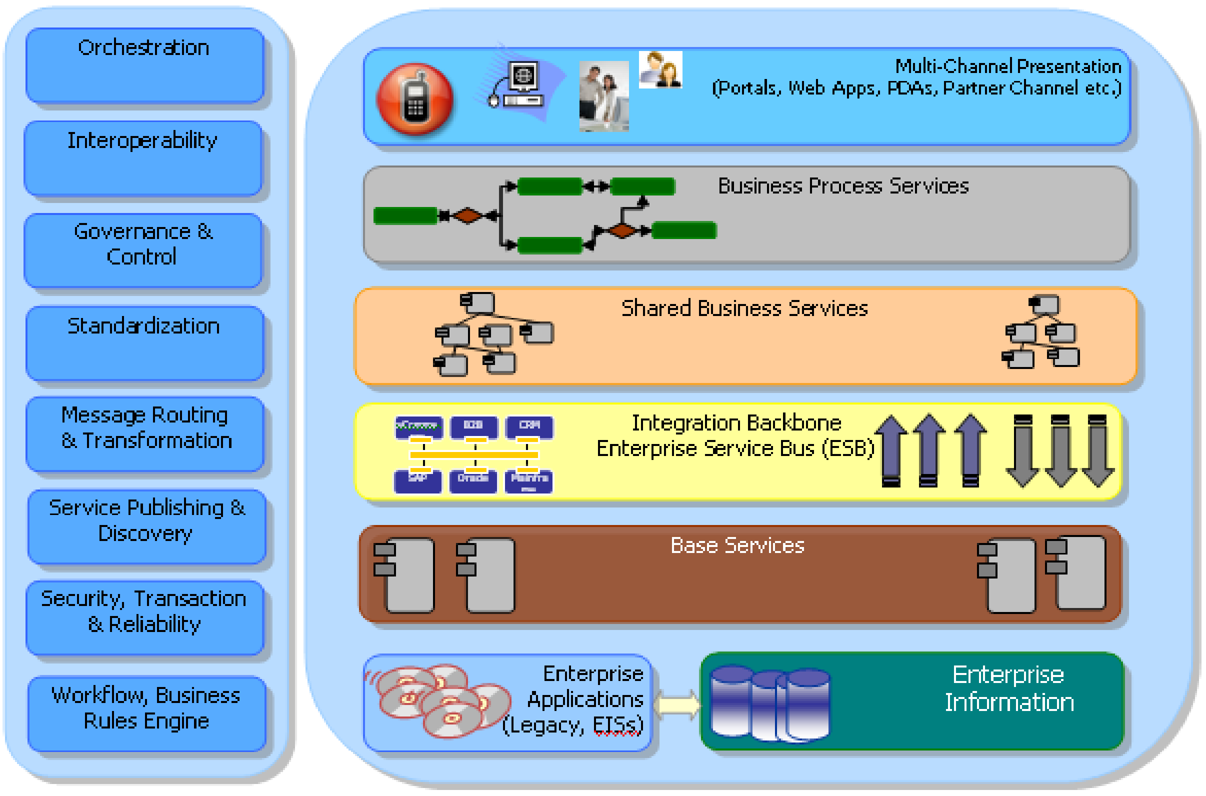

Orchestration
Interoperability
Governance & Control
Standardization
Message Routing & Transformation
Service Publishing & Discovery
Security, Transaction & Reliability
Workflow, Business Rules Engine
Multi-Channel Presentation (Portals, Web Apps, PDAs, Partner Channel etc.)
Business Process Services
Shared Business Services
Integration Backbone Enterprise Service Bus (ESB)
Base Services
Enterprise Applications (Legacy, EISs)
Enterprise Information

Fig 2.0

Like services, it is difficult to find an agreed-upon definition of architecture. However, unlike services, architecture is sometimes forgotten when people talk about SOA, and it clearly should not be! In fact, enterprise architecture and Service-Oriented Architecture share common goals around supporting the business with an integrated IT strategy. Enterprise architects, for example, are key to the success of SOA, as they look at the strategic evolution of an enterprise's IT systems based on evolving business needs and demands.

The Open Group Architecture Forum (TOGAF) provides two definitions for architecture, based on context: "A formal description of a system, or a detailed plan of the system at component level to guide its implementation. The structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time."

These two definitions are relevant to understanding the "A" of SOA. Breaking it down even further, we find that architecture is necessary to do the following:

- Design and model at different levels of abstractions
- Separate specification from implementation
- Build flexible systems
- Make sure business requirements are addressed
- Analyze the impact of a change in requirements
- Ensure principles are followed

3.0 Challenges in SOA Testing

- 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.

4.0 Various Areas of SOA Testing

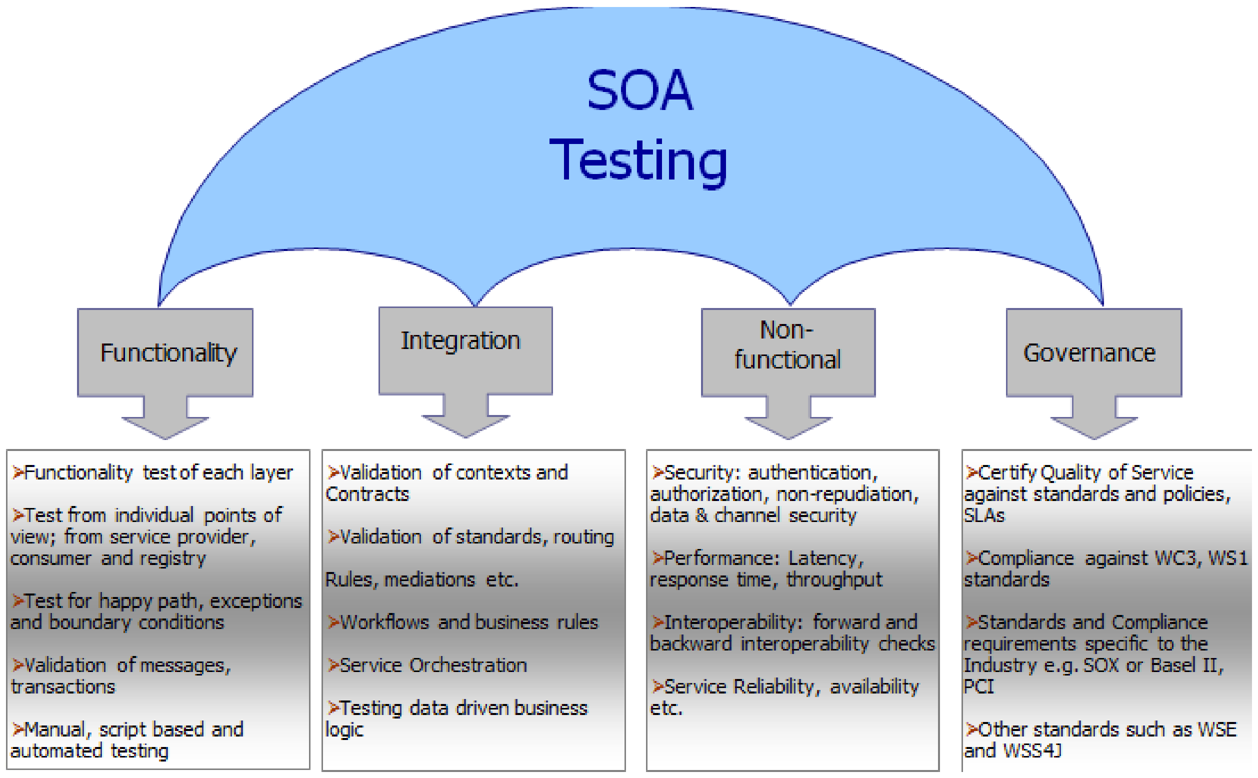
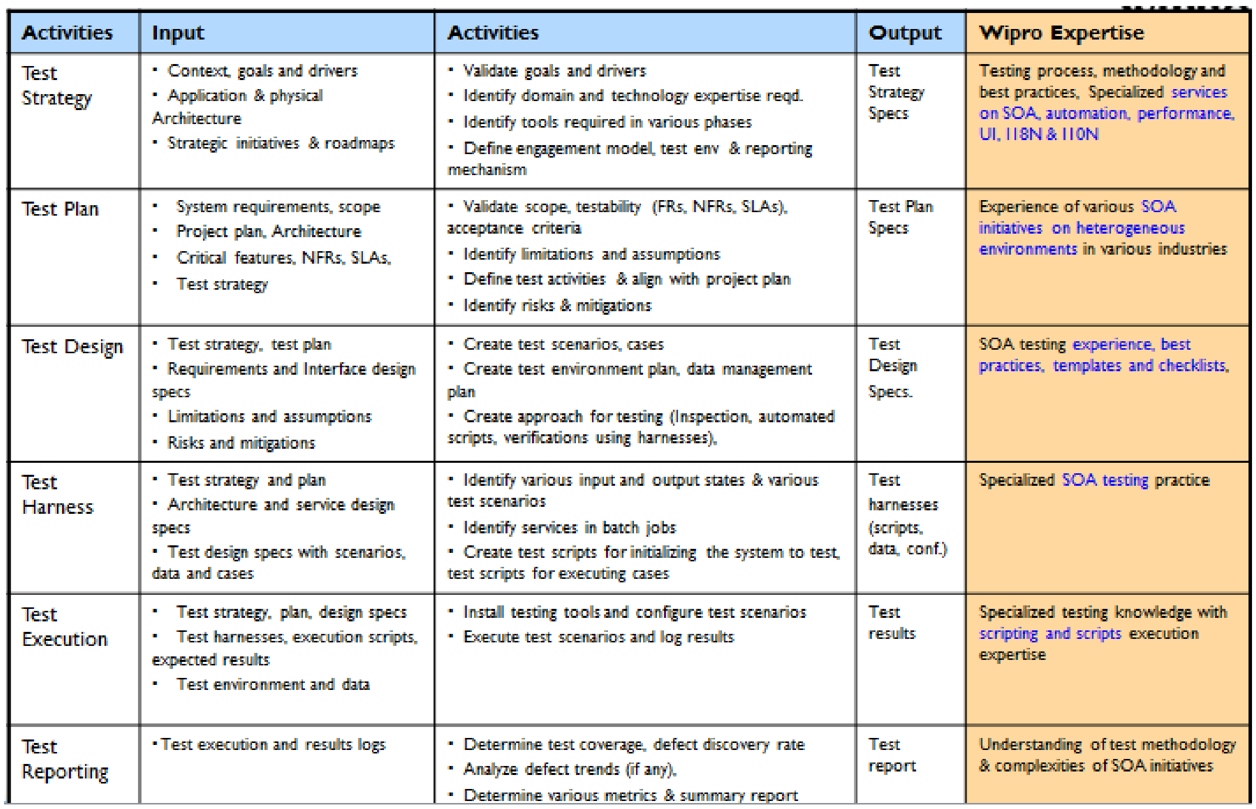

SOA Testing
Functionality
Integration
Non-functional
Governance
➢Functionality test of each layer
➢Test from individual points of view; from service provider, consumer and registry
➢Test for happy path, exceptions and boundary conditions
➢Validation of messages, transactions
➢Manual, script based and automated testing
➢Validation of contexts and Contracts
➢Validation of standards, routing Rules, mediations etc.
➢Workflows and business rules
➢Service Orchestration
➢Testing data driven business logic
➢Security: authentication, authorization, non-repudiation, data & channel security
➢Performance: Latency, response time, throughput
➢Interoperability: forward and backward interoperability checks
➢Service Reliability, availability etc.
➢Certify Quality of Service against standards and policies, SLAs
➢Compliance against WC3, WS1 standards
➢Standards and Compliance requirements specific to the Industry e.g. SOX or Basel II, PCI
➢Other standards such as WSE and WSS4J

Fig 3.1

SOA Testing and quality assurance, ideally, should be carried out by experts with considerable experience. Broad experience is needed because SOA testing involves not just functionality checks, but also the reuse and agility checks that require support from domain specialists, architects and designers in terms of design reviews. SOA testing however has seen lot of demand in the past couple of years, and the trend promises to continue in the years ahead. Because of the increased demand on testing resources, there has been considerable impact on the testing resource standards in some instances. Thus, design reviews have become even more critical to SOA quality assurance. While the functional testing can be done fairly easily by a moderately experienced SOA tester, using tools available in the market, the reuse and agility testing requires thorough assessment by the business analyst, SOA architect and the information architect. The business analyst would bring domain knowledge to the process, the SOA architect all the design and architectural decisions made, and the information architect would contribute knowledge on utilization of all data entities involved across the business processes.

Reuse and agility testing obviously require special attention as these are the main areas in which the enterprise leadership expects the maximum return on investment. Effective design reviews with special attention to reuse are a key factor in ensuring achievement of this goal. The significant investment of time and financial and personnel resources required to realize the benefits of SOA transformation is often an impediment to management’s wholehearted endorsement of a SOA initiative. Hence several IT initiatives are being sprinkled with SOA with either no long term strategy or without a major one-time investment. In either of these circumstances, in addition to institutionalizing effective governance practices to define and roll out necessary IT processes, strengthening quality assurance processes within micro SOA efforts would make overall SOA transformation a success. This would ensure that a piece of code or logic marked as a service is actually reusable in different contexts and contributes to required agility.

5.0 Test Activities

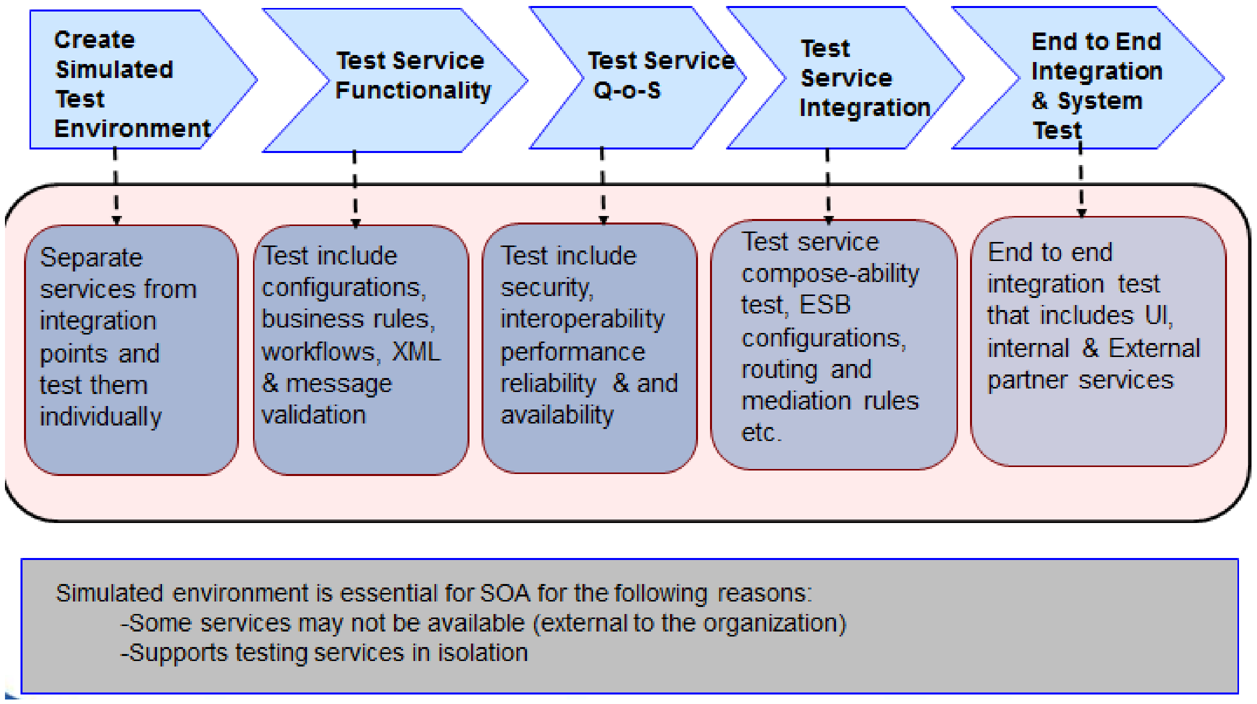


Activities	Input	Activities	Output	Wipro Expertise
Test Strategy	• Context, goals and drivers • Application & physical Architecture • Strategic initiatives & roadmaps	• Validate goals and drivers • Identify domain and technology expertise reqd. • Identify tools required in various phases • Define engagement model, test env & reporting mechanism	Test Strategy Specs	Testing process, methodology and best practices, Specialized services on SOA, automation, performance, UI, I18N & I10N
Test Plan	• System requirements, scope • Project plan, Architecture • Critical features, NFRs, SLAs, • Test strategy	• Validate scope, testability (FRs, NFRs, SLAs), acceptance criteria • Identify limitations and assumptions • Define test activities & align with project plan • Identify risks & mitigations	Test Plan Specs	Experience of various SOA initiatives on heterogeneous environments in various industries
Test Design	• Test strategy, test plan • Requirements and Interface design specs • Limitations and assumptions • Risks and mitigations	• Create test scenarios, cases • Create test environment plan, data management plan • Create approach for testing (Inspection, automated scripts, verifications using harnesses),	Test Design Specs.	SOA testing experience, best practices, templates and checklists,
Test Harness	• Test strategy and plan • Architecture and service design specs • Test design specs with scenarios, data and cases	• Identify various input and output states & various test scenarios • Identify services in batch jobs • Create test scripts for initializing the system to test, test scripts for executing cases	Test harnesses (scripts, data, conf.)	Specialized SOA testing practice
Test Execution	• Test strategy, plan, design specs • Test harnesses, execution scripts, expected results • Test environment and data	• Install testing tools and configure test scenarios • Execute test scenarios and log results	Test results	Specialized testing knowledge with scripting and scripts execution expertise
Test Reporting	• Test execution and results logs	• Determine test coverage, defect discovery rate • Analyze defect trends (if any), • Determine various metrics & summary report	Test report	Understanding of test methodology & complexities of SOA initiatives

Fig 3.3

- In any software development project, testing requires significant time, effort, and discipline. The following common steps should be followed to assure proper and efficient testing:
- Define a test plan that outlines the testing process and exit criteria
- Derive test cases from use cases or business requirements
- Generate test data and/or scripts for each test case
- Outline the expected results for each test case
- Execute the test cases
- Verify whether the results of the test cases match the expected results
- Generate reports to measure the software's quality against the test cases
- Fix/resolve remaining defects in the software
- Continue executing/verifying/fixing until all test cases succeed and defects are resolved

6.0 Testing Approach



Create Simulated Test Environment
Test Service Functionality
Test Service Q-o-S
Test Service Integration
End to End Integration & System Test
Separate services from integration points and test them individually
Test include configurations, business rules, workflows, XML & message validation
Test include security, interoperability performance reliability & and availability
Test service compose-ability test, ESB configurations, routing and mediation rules etc.
End to end integration test that includes UI, internal & External partner services
Simulated environment is essential for SOA for the following reasons:
-Some services may not be available (external to the organization)
-Supports testing services in isolation

Fig 4.1

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.

7.0 Testing Methodology

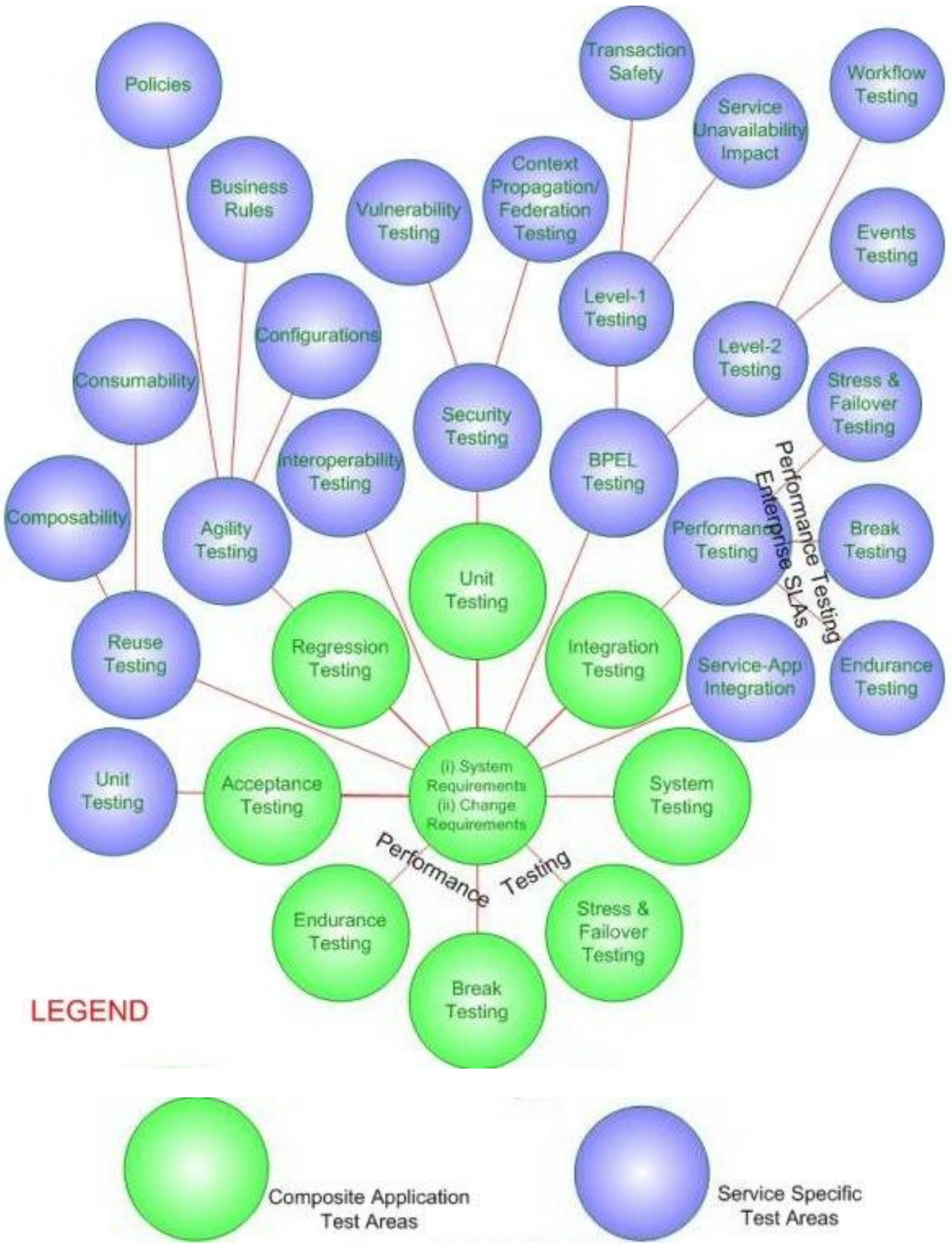


Fig 4.2

An enterprise SOA program usually has multiple projects involving applications and services developed in parallel. Ideally, service development should lead composite application Development by at least one activity or cycle for smooth integration and testing of applications and services. Careful planning is required to ensure that all dependencies are accounted for between services and composite application development. This means that before unit testing the (Composite) application, the services required for it should have already been unit tested.
Services should have also undergone additional tests for reusability, agility, inter-operability and Security before they are consumed by composite applications.

Note that SOA transformation involves iterative development of services and applications and hence involves testing in all iterations with automated regression testing as shown in figure4.2... GREEN colored circles represent composite application testing with interspersed service test areas represented in BLUE colored circles.
Traditional systems would undergo functional testing comprising unit, integration, system, acceptance, regression testing and performance testing comprising stress, fail-over, break and endurance testing. SOA functional testing focuses on testing the functionality for the composite application that is built using sub-systems or modules that are independent of the rest of the IT eco-system. It also involves testing services that are woven into the application logic with dialog Control handled by either BPEL (Business Process Execution language) engines or non-BPEL implementations. The concept of using and composing reusable services to construct new applications thus brings the following additional test areas into overall SOA testing as depicted in the diagram shown above and below.

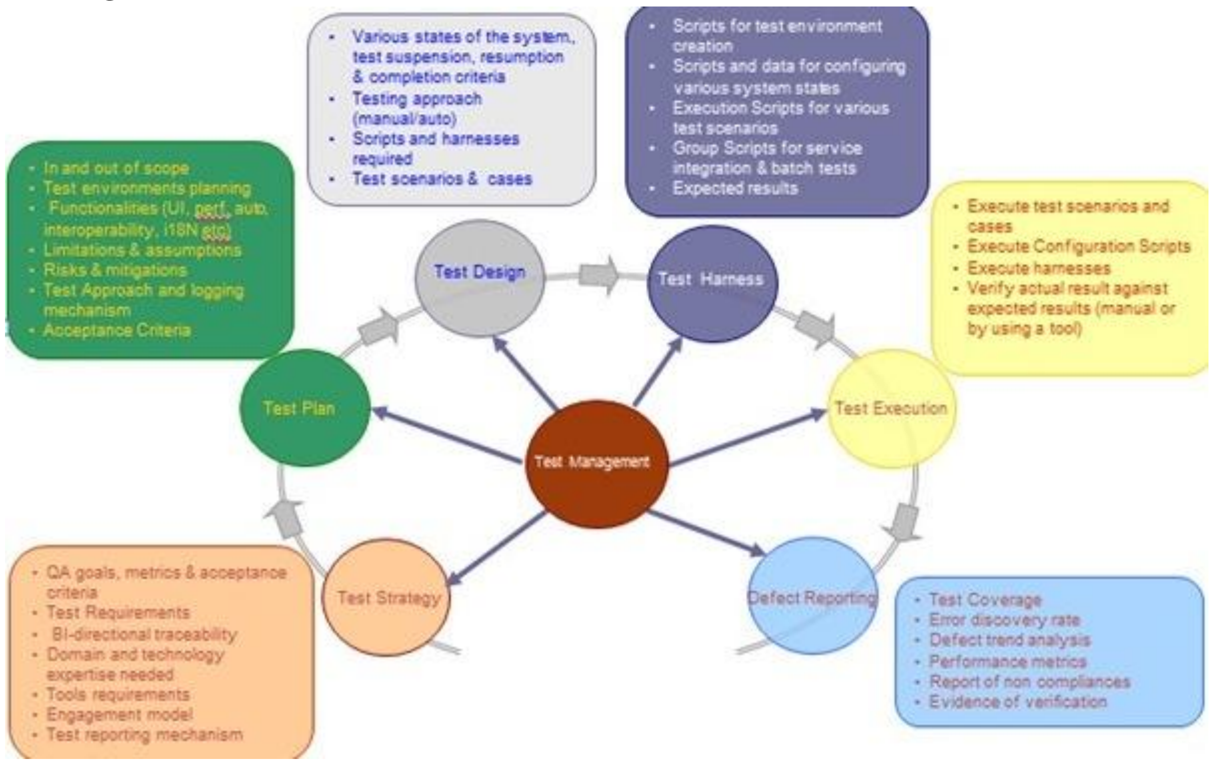

Various states of the system, test suspension, resumption & completion criteria
Testing approach (manual/auto)
Scripts and harnesses required
Test scenarios & cases
Scripts for test environment creation
Scripts and data for configuring various system states
Execution Scripts for various test scenarios
Group Scripts for service integration & batch tests
Expected results
In and out of scope
Test environments planning
Functionalities (UI, perf, auto, interoperability, i18N etc)
Limitations & assumptions
Risks & mitigations
Test Approach and logging mechanism
Acceptance Criteria
Execute test scenarios and cases
Execute Configuration Scripts
Execute harnesses
Verify actual result against expected results (manual or by using a tool)
Test Design
Test Harness
Test Plan
Test Execution
Test Management
Test Strategy
Defect Reporting
QA goals, metrics & acceptance criteria
Test Requirements
Bi-directional traceability
Domain and technology expertise needed
Tools requirements
Engagement model
Test reporting mechanism
Test Coverage
Error discovery rate
Defect trend analysis
Performance metrics
Report of non compliances
Evidence of verification

Fig 4.3

8.0 SOA Testing Techniques

Different levels of SOA testing

1. Service Agility Testing
    a. Configurations
    b. Business Rules
    c. Policies

2. BPEL Level-1 Testing
    a. Compensating transactions
    b. Service unavailability impact

3. BPEL Level-2 Testing
    a. Workflow Testing
    b. Events Testing

4. Security Testing
    a. Denial-of-Service (DOS)
    b. Vulnerability
    c. Context propagation/Federation

5. Service Design Testing
    a. Interoperability Testing
    b. Service-App Integration Testing
    c. Reuse Testing (Consumability and Composability)
    d. Service Data Testing

6. SOA Performance Testing

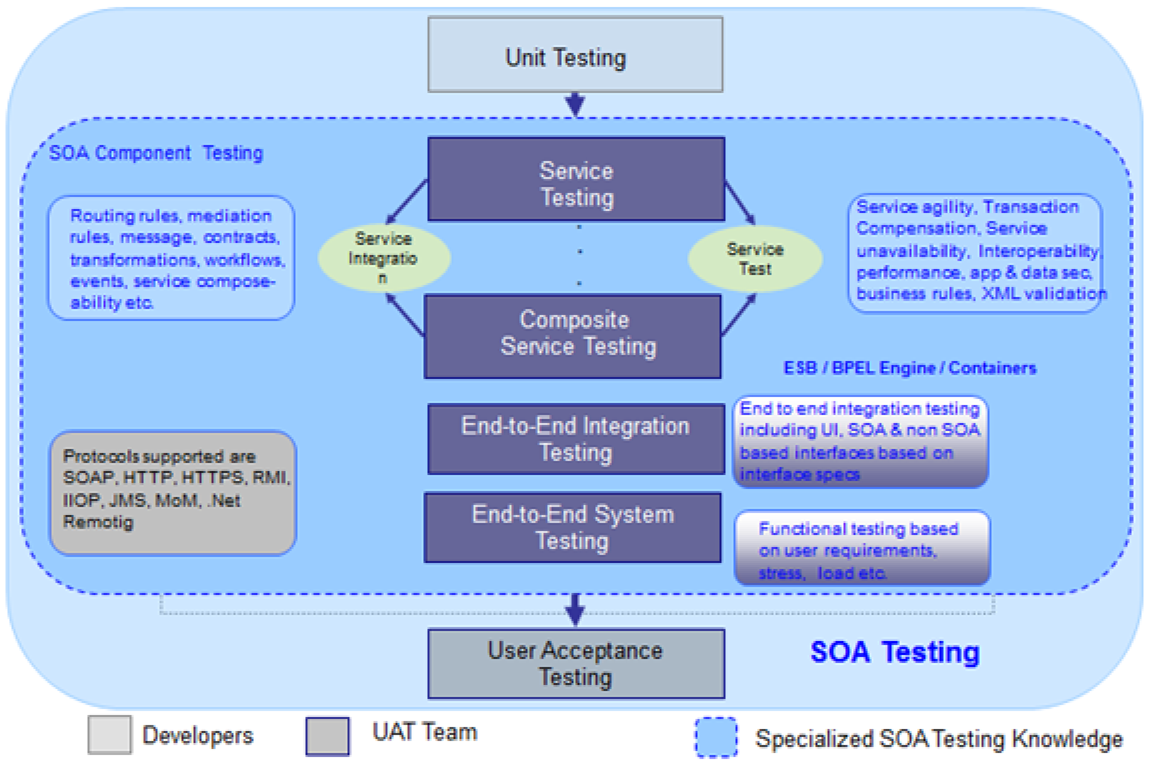

Unit Testing
SOA Component Testing
Routing rules, mediation rules, message, contracts, transformations, workflows, events, service compose-ability etc.
Service Testing
Service Integration
Service Test
Composite Service Testing
Service agility, Transaction Compensation, Service unavailability, Interoperability, performance, app & data sec, business rules, XML validation
ESB / BPEL Engine / Containers
Protocols supported are SOAP, HTTP, HTTPS, RMI, IIOP, JMS, MoM, .Net Remotig
End-to-End Integration Testing
End to end integration testing including UI, SOA & non SOA based interfaces based on interface specs
End-to-End System Testing
Functional testing based on user requirements, stress, load etc.
User Acceptance Testing
SOA Testing
Developers
UAT Team
Specialized SOA Testing Knowledge

Fig 5.0

Service Agility Testing: This procedure requires complete understanding of the requirements for which the service is designed. The review should focus on identifying the volatile parts of the requirements, since agility is all about externalizing them into parameters via configurations, rules and policies. All of these parameters would change the behavior of a service or component in different contexts. Configurable items influence system functions, rules would influence business functions, and policies would influence security, monitoring, performance etc. of services. The reviewer should make sure that all the volatility is captured in either configurations or rules or policies

This essentially reinforces the fact that most of the iterative development models like TDD, SCRUM, and XP etc. are pushing the design and domain specialists to be more involved in testing and quality assessment activities.

Another important factor in agility testing is to measure how quickly service can be provisioned to accept a new consumer. Please refer to interface, implementation, data and packaging considerations of service for design guidelines.

One of the measures is to define certain test cases for data integration by changing some data entity semantics and changing the appropriate map to see how quickly the systems in question come back to normal operations.

BPEL Testing

In the case of composite services, two levels of testing are required

i. Orchestration (BPEL) Testing
ii. Process level testing.

Level-1 Testing is aimed at making sure that the sequence of service calls made to various systems will not leave any of the systems in an inconsistent state. This test also helps in Understanding the impact of one or more services in the chain not being available. Note that the more a service gets reused, the greater the impact associated with its downtime will be.

Services are composed to form the business processes either partially or completely by orchestrating them in a defined sequence. These services may be hosted on different platforms with different underlying transactional engines. In the event of failure of any service invocation in the sequence, BPEL provides compensating transactions (service methods) to maintain the data in a clean state. All the compensating transactions have to be tested thoroughly with exhaustive test cases and data. It is essential to test the level of impact in case of service unavailability to the business process as a whole, and to check the remedial mechanisms provided and their effectiveness.

Level-2 Testing is aimed at testing the flexibility in the workflow design provided and also the effectiveness of process visibility details captured via events.

Since SOA involves a lot of asynchronous communication across systems, it is possible that messages will not reach their destinations in an orderly fashion because of delays in processing at end-points (target systems). Some systems will respond quickly, and some slowly, depending on the load at any given point in time. Response messages must to be correlated centrally before they are dispatched to the target systems. Changes in workflow may result in changes in the delivery order of the messages from one system to another. Test cases should be aimed at testing this scenario to measure how quickly a workflow can be changed. The results are in fact a reflection on the extent of agility built in to the enterprise's SOA.

Testing the above scenarios requires thorough involvement of the development team during the testing process.

Security Testing

Typically, major SOA initiatives involve exposing services to and consuming services from the outside world. This fact opens up a host of vulnerabilities, such as DOS (denial-of-service) attacks, penetration, high volumes of spam data, etc. Typical security policies have to be enforced at the network level, middleware level and at the end-point level to create bullet proof SOA. Specific test cases aimed at targeting these policies need to be designed to fully test SOA security.

Service Design Testing

Reuse Testing: Knowledge of design decisions made and the utilization of data entities within various business processes would be validated by the domain knowledge of the business analyst to make certain that the interface or contract would work in all possible business scenarios, thus ensuring consumability and ability to compose.

Interoperability Testing: Since enterprise SOA development involves different technology platforms and development tools at the enterprise level, it is quite possible that developers use many different tools to generate the contracts (WSDL) and modify them in the process. Also, different runtime SOAP engines would interpret contracts differently, based on their implementations. Hence it is essential to make sure that all (web) services deployed are compliant with WS-I basic profile. Tools such as SOAPScope, SOAPTest and tools from WS-I org etc. Provide the necessary support in this testing process.

Service – Application Integration Testing: In a big SOA program, it is quite common that different services and applications are developed independently in parallel by multiple vendors in different geographical locations. Defining the contracts or interfaces at the beginning would help the development effort to go smoothly. But at the time of integration, it is useful to have an integrated test environment where services can be deployed, tested and integrated with Applications for overall testing.

Data Level Testing: For services, data must be tested for completeness, correlation across different consumers, separation of data for consumers, element level security as needed and ownership assessment by testing support services such as reporting and data archival mechanisms, etc.

SOA Performance Testing

Performance testing is needed for both services and composite applications. Performance testing usually involves a lot of time and repetitive test cycles, and needs careful planning for dependencies. Performance testing depends on the following enterprise level and application/service level activities:

Enterprise Level Activities:

1. Standardizing and procuring the software required

2. Standardizing and procuring the hardware required

3. Setting up the required test environment/lab

4. Setting up the shared dedicated service hosting zone

Application/Service (development) Level Activities:

The following few line items add some additional effort, time and cost, and need to be included in the overall planning of the SOA projects.

1. Scripting necessary for load generation

2. Preparing test data

3. (for continuous monitoring by tools like Sitescope etc.) Providing a sample UI to test the service Optimizations in both service and application code are to be expected from the issues that emanate from performance testing.

Part 1: Services Testing

Services are typically intended to have high usage and high reuse opportunities. Considering this fact, their performance requirements are more stringent compared to composite applications. For example, a service has been identified as part of a micro SOA effort and has been given a performance SLA of serving 50 concurrent users with 8 sec of response time. The service factory team would therefore need to target a performance requirement of at least 5(50) = 250 or 10(50) =500 concurrent user load with 8 sec response time.
Typically, services are server side components that do not have a front end. It is therefore useful to have a sample GUI interface that can be used to load test the service from a web client. But in the real world, services would be accessed by different types of clients and with different date formats. For testing the real world service, it is a good idea to select a test tool that can emulate different clients with different types of data (good and bad) and volumes to check if the service is responding as expected. This measure would improve the productivity of the development team significantly.

Part 2: Composite Application Testing

When a composite application is being tested for load, stress, fail-over and endurance, it is important to observe the throughput of the mediation component or service bus to see if a bottleneck occurs. For SOA architects in the IT services business, it might be a challenge to convince the customer-side business and IT management to accept reasonable end-to-end performance SLAs. In the pre-SOA era, where tiered architectures mapped on to a web server, application server and a database in different configurations, the 8-sec rule was used as reference point. While the same rule can be applied to SOA applications, test cases need to ensure that the response time is maintained under various levels of stress.

Note that in a distributed architecture like SOA where services developed internally and externally are used to build composite applications,, keeping the bar at 8 sec might be a challenge in certain business scenarios.

SOA architects must consider virtualization and GRID computing as potential performance solutions. And also, it is extremely important for SOA architects to involve the operations engineering team for required capacity planning to get ground realities before committing to SLAs.

9.0 SOA Testing Best Practices

SOA Quality Assurance Processes

The following are the important testing activities that ensure SOA quality.

- Automate test case execution to improve productivity and to detect errors at the time they are introduced into the code by developers. Look for tools that facilitate static code analysis to detect standard code errors and enforce coding standards, and run-time analysis to detect resource leakages, potential bottlenecks, security breaches, etc.
- Ensure that real-world test data is used right from the outset. This facilitates the detection of errors that creep in as newer architectural layers are added.
- Automate regression testing to account for short cycles of iterative development. This helps to manage the whole testing process.
- Maintain continuous regression testing through nightly batch process.
- Develop a system for dash board reporting of errors to the entire SOA team
- Maintain regular quality assurance audits to ensure that major scenarios catering to business processes are run without errors. This needs involvement of SOA architect, business analyst and executives.

SOA governance must institutionalize the processes to ensure that design reviews are targeted towards reuse and agility testing against the requirements.

Requires paradigm shift from traditional testing.

- Test Services in Isolation and then as a whole
- Test earlier in the life cycle

Requires testing from multiple perspectives.

- Test from the Service Providers, consumers and registry perspective
- Requires Tools!!
- Use a tool for better coverage and quality assurance
- Requires testing along multiple dimensions.
- Functionality
- Interoperability and compliance to standards
- Security
- Load and Performance
- Transactions
- Reliability and availability

10.0 Benefits of Testing SOA

- Ensure the reliability, quality, security and interoperability of the Web service.
- Penetration testing integrated with functional testing for complete coverage.
- Uniform test suites can be rolled over from unit testing to functional testing to load testing to security testing.
- Prevent errors, pinpoint weaknesses, and stress test long before deployment.
- Verify data integrity and server/client functionality.
- Identify server capabilities under stress and load.
- Accelerate time to market.
- Leverage existing assets.
- Easier to integrate and manage complexity.
- More responsive and faster time-to-market.
- Reduce cost and increase reuse.
- Be ready for what lies ahead.

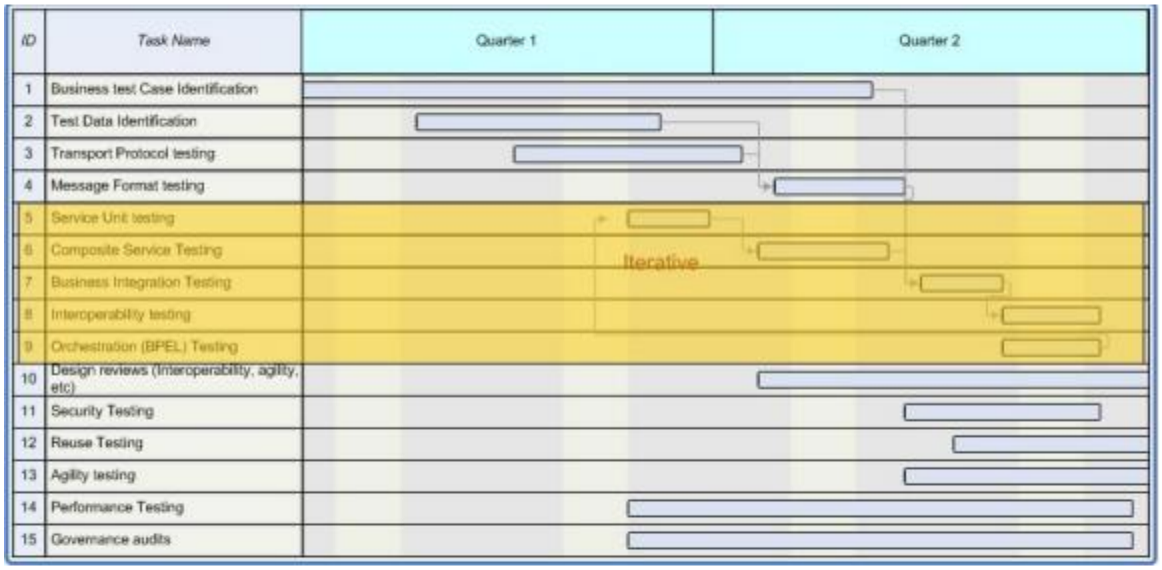
The last and best option is a fully integrated testing solution that enables testers to test at each point and end-to-end. This would include these key capabilities:

i. Creating and sending a message to a provider application;
ii. Receiving and verifying a message to a consumer application;
iii. Either sending or receiving messages to or from files when the applications aren't available or to isolate the infrastructure, and
iv. Testing end-to-end through the user interface of the provider and consumer applications and the messages in between.

11.0 SOA Test Roadmap

In the previous sections, we have differentiated the various aspects of SOA testing from conventional software testing. Putting it all together is another significant task in which timing is key to ensure minimum redundancy and maximum test efficiency.

The following indicative SOA Test Roadmap gives a bird's eye view of the overall sequence of activities during SOA Testing. From figure 6.0 below, it can be seen that Service level testing is iterative in nature and occurs throughout the duration of the project. Other test activities, such as Governance Audits, performance, reuse, agility and security testing are continuous activities that take place throughout the duration of the project.



ID	Task Name
1	Business test Case Identification
2	Test Data Identification
3	Transport Protocol testing
4	Message Format testing
5	Service Unit testing
6	Composite Service Testing
7	Business Integration Testing
8	Interoperability testing
9	Orchestration (BPEL) Testing
10	Design reviews (Interoperability, agility, etc)
11	Security Testing
12	Reuse Testing
13	Agility testing
14	Performance Testing
15	Governance audits

Fig 6.0: Roadmap for SOA testing

12.0 Service Virtualization

Applications are essential to business today, yet time-to-market, development costs, and application quality remain significant challenges. The complexity and constraints inherent to composite application development continue to disrupt application development efforts despite ever-increasing investment and focus.

Most of the legacy IT environments and business processes are built on loosely coupled and highly distributed environment. Inherent constraints and dependencies in such heterogeneous platform adversely impacts Development and QA timelines. These challenges lead to delays, lower quality and higher project costs which are in conflict with customer's interest of achieving faster time to market with enhanced quality at reduced cost.

Service virtualization overcomes these constraints and dependencies by capturing and simulating the behavior, data and performance characteristics of dependent systems. This create and deploy virtual services that represent the dependent systems without any constraints allowing software to the developed and tested hence delivered faster with lower costs and higher reliability.

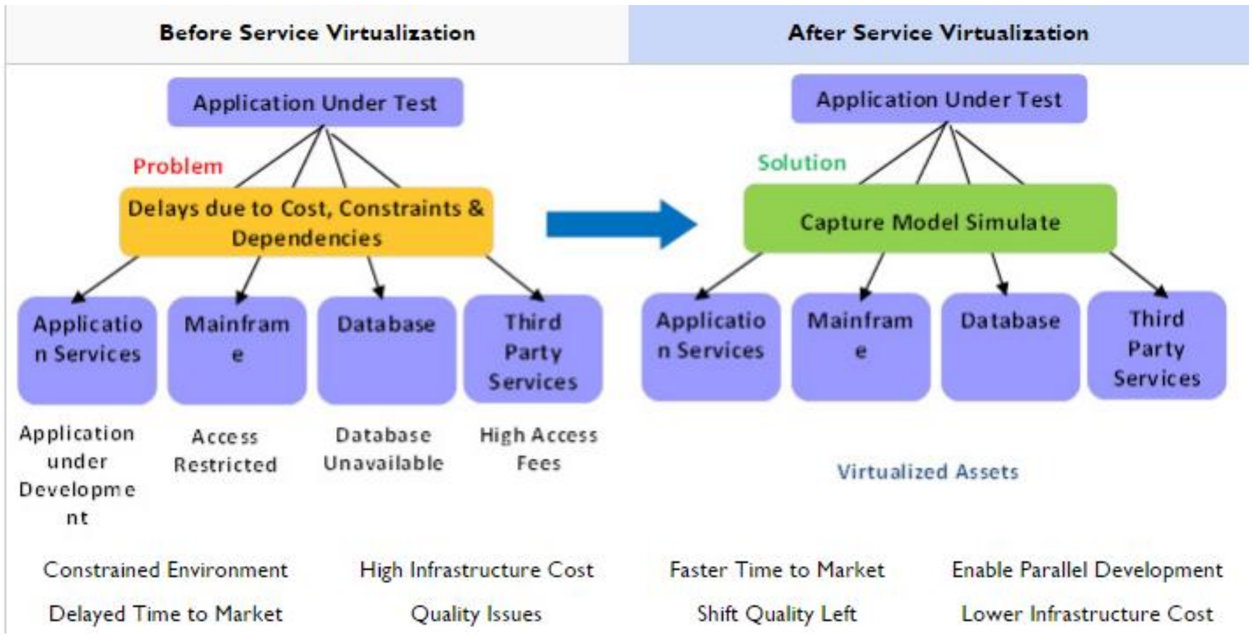
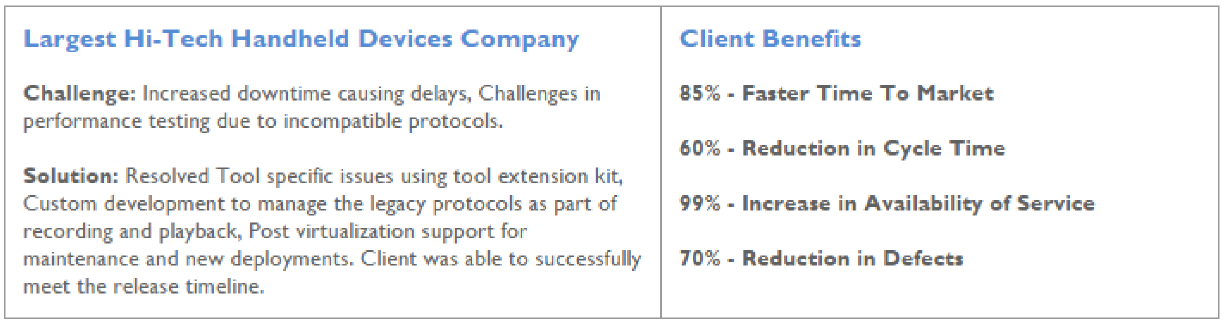


Fig 7.0



Largest Hi-Tech Handheld Devices Company	Client Benefits
Challenge: Increased downtime causing delays, Challenges in performance testing due to incompatible protocols.	**85% - Faster Time To Market**
	60% - Reduction in Cycle Time
Solution: Resolved Tool specific issues using tool extension kit, Custom development to manage the legacy protocols as part of recording and playback, Post virtualization support for maintenance and new deployments. Client was able to successfully meet the release timeline.	**99% - Increase in Availability of Service**
	70% - Reduction in Defects

Fig 7.1

How Service Virtualization Relates to Stubbing and Mocking

An alternative approach to working around the test environment access constraints outlined in this article's introduction is for team members to develop method stubs or mock objects that substitute for dependent resources. The shortcoming of this approach became apparent in the early 2000s with the rise of Service-oriented architecture. The proliferation of Composite applications that rely on numerous dependent services, plus the rise of Agile software development following the 2001 publication of the Agile Manifesto, made it increasingly difficult for developers or testers to manually develop the number, scope, and complexity of stubs or mocks required to complete development and testing tasks for modern enterprise application development

The first step in the evolution from stubbing to service virtualization was the technology packaged in SOA testing tools since 2002. The earliest implementations of service virtualization were designed to automate the process of developing simple stub-like emulations so that composite applications could be tested more efficiently. As enterprise systems continued to grow increasingly complex and distributed, software tool vendors shifted focus from stubbing to the more environment-focused service virtualization. While stubbing can still be completed through manual development and management of stubs, what has become known as "service virtualization" is completed by using one of the available commercial off the shelf (COTS) service virtualization technologies as a platform for the development and deployment of their "service virtualization assets

13.0 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

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.

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.

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.

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).

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.

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.

Open Source Products

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.

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 behavior 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.

The Figure below show the representation of best players/vendors in the market based on their popularity and usage in the market. The above mentioned tools can be figured out in the below quadrants based on their popularity with clients.

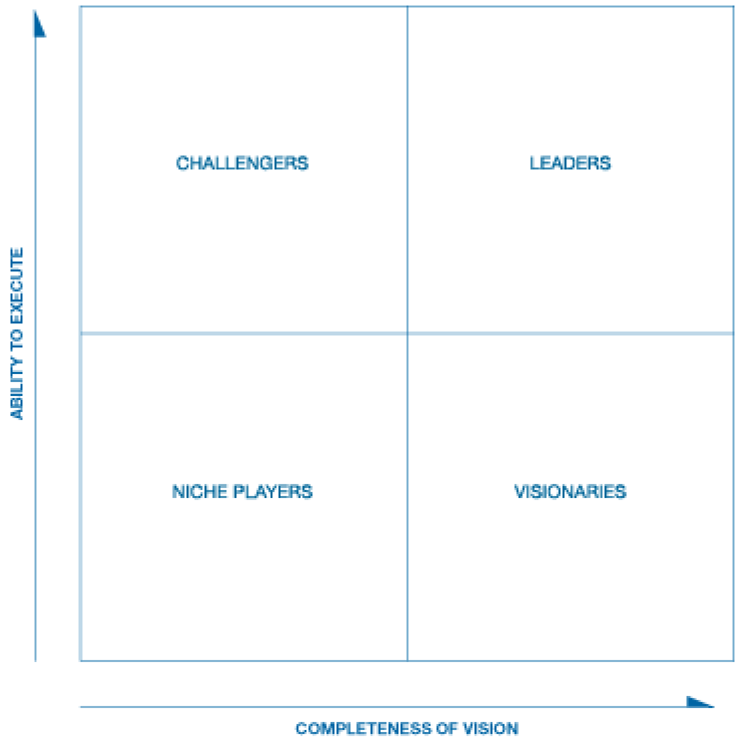
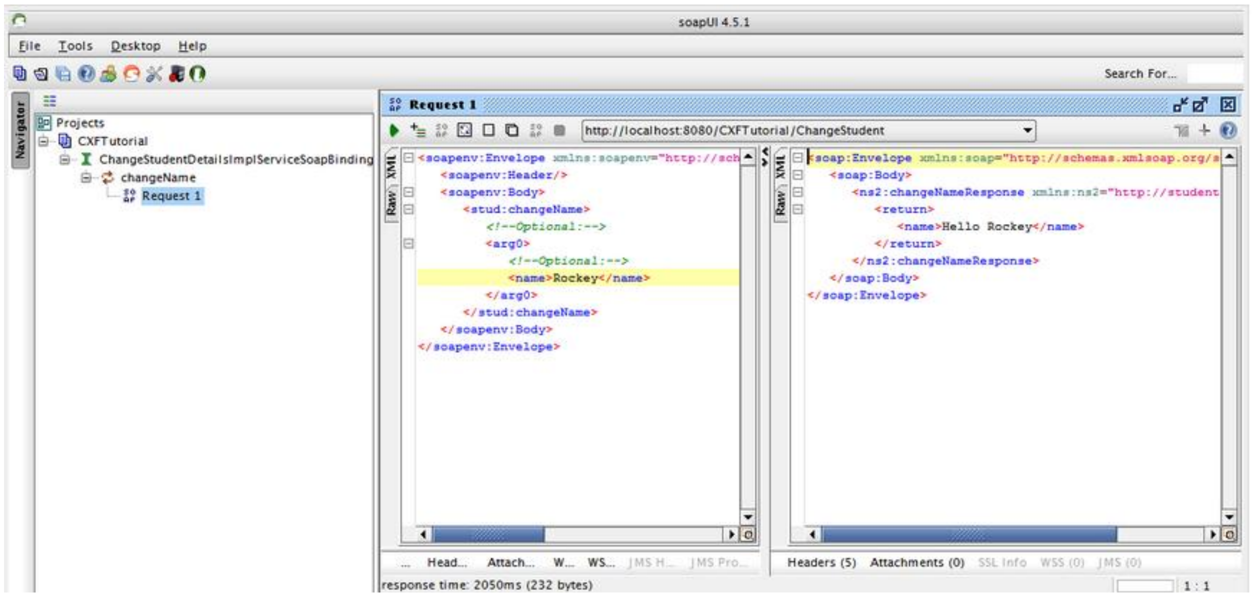


Fig 8.0

14.0 Sample Web service project



```
<soapenv:Envelope xmlns:soapenv="http://sch
   <soapenv:Header/>
   <soapenv:Body>
      <stud:changeName>
         <!--Optional:-->
         <arg0>
            <!--Optional:-->
            <name>Rockey</name>
         </arg0>
      </stud:changeName>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/s
   <soap:Body>
      <ns2:changeNameResponse xmlns:ns2="http://student
         <return>
            <name>Hello Rockey</name>
         </return>
      </ns2:changeNameResponse>
   </soap:Body>
</soap:Envelope>
```

response time: 2050ms (232 bytes)

Fig 9.0

Sample – Adding of two numbers

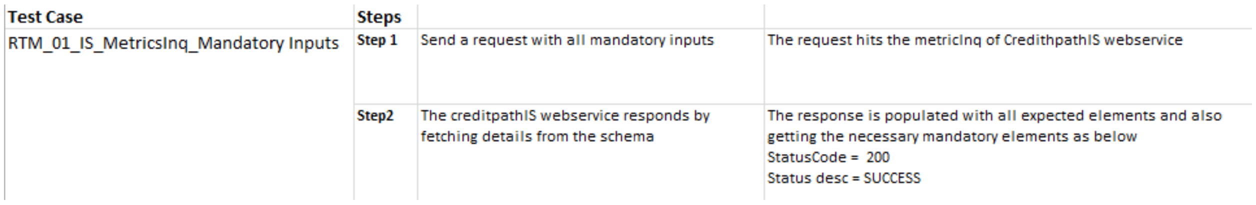
REQUEST FILE

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tem="http://tempuri.org/">
   <soapenv:Header/>
   <soapenv:Body>
      <tem:AddThis>
         <tem:x>5</tem:x>
         <tem:y>10</tem:y>
      </tem:AddThis>
   </soapenv:Body>
</soapenv:Envelope>
```

RESPONSE FILE

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <AddThisResponse xmlns="http://tempuri.org/">
         <AddThisResult>15</AddThisResult>
      </AddThisResponse>
   </soap:Body>
</soap:Envelope>
```

Example web service test case



Test Case	Steps		
RTM_01_IS_MetricsInq_Mandatory Inputs	**Step 1**	Send a request with all mandatory inputs	The request hits the metricInq of CredithpathIS webservice
	Step2	The creditpathIS webservice responds by fetching details from the schema	The response is populated with all expected elements and also getting the necessary mandatory elements as below StatusCode = 200 Status desc = SUCCESS

Sample Groovy scripting used SoapUI

Groovy is an object-oriented programming language for the Java platform. It is a dynamic language with features similar to those of Python, Ruby, Perl, and Smalltalk. It can be used as a scripting language for the Java Platform, is dynamically compiled to Java Virtual Machine (JVM) byte code, and interoperates with other Java code and libraries. Groovy uses a Java like curly bracket syntax. Most Java code is also syntactically valid Groovy.

XML and JSON Processing

On XML and JSON processing Groovy employs the Builder pattern, making the production of the data structure less verbose.

```
import jxl.*
import jxl.write.*

def groovyUtils = new com.eviware.soapui.support.GroovyUtils(context)
WritableWorkbook book = Workbook.createWorkbook(new File("D:\\ \\SOUP_UI\\Banking\\RQT N RSP\\validateOutput4.xls"))
WritableSheet sht = book.createSheet("validate",0)

Label label1 = new Label(0,0, "User")
sht.addCell(label1)

Label label2 = new Label(1,0, "RequestXML")
sht.addCell(label2)

Label label3 = new Label(2,0, "ResponseXML")
sht.addCell(label3)

FileReader Input = new FileReader(new File("D:\\puneeth\\SOA\\SOUP_UI\\Banking\\RQT N RSP\\validateInput4.csv"))
BufferedReader read = new BufferedReader(Input)

String[] row = null
String line
int i = 1
```

```
def ValidateRequest, ValidateResponse
while((line = read.readLine()) != null)
{
            row = line.split(",")

            ValidateRequest = groovyUtils.getXmlHolder("TS_validate#Request")
            context.setProperty("username",row[0])
            context.setProperty("password",row[1])
            testRunner.runTestStepByName("TS_validate")

            ValidateResponse = groovyUtils.getXmlHolder("TS_validate#Response")

            label1 = new Label(0, i ,row[0])
            sht.addCell(label1)


            label2 = new Label(1, i, ValidateRequest.getXml())
            sht.addCell(label2)

            label3 = new Label(2, i, ValidateResponse.getXml())
            sht.addCell(label3)


            i++
            log.info"validating "+i

}
log.info"Check output"
read.close()
book.write()
book.close()

return null
```

In most SOA testing projects the requirement is obtained and feasibility study is done regarding efforts and capacity of individual to develop/ test the developed service.

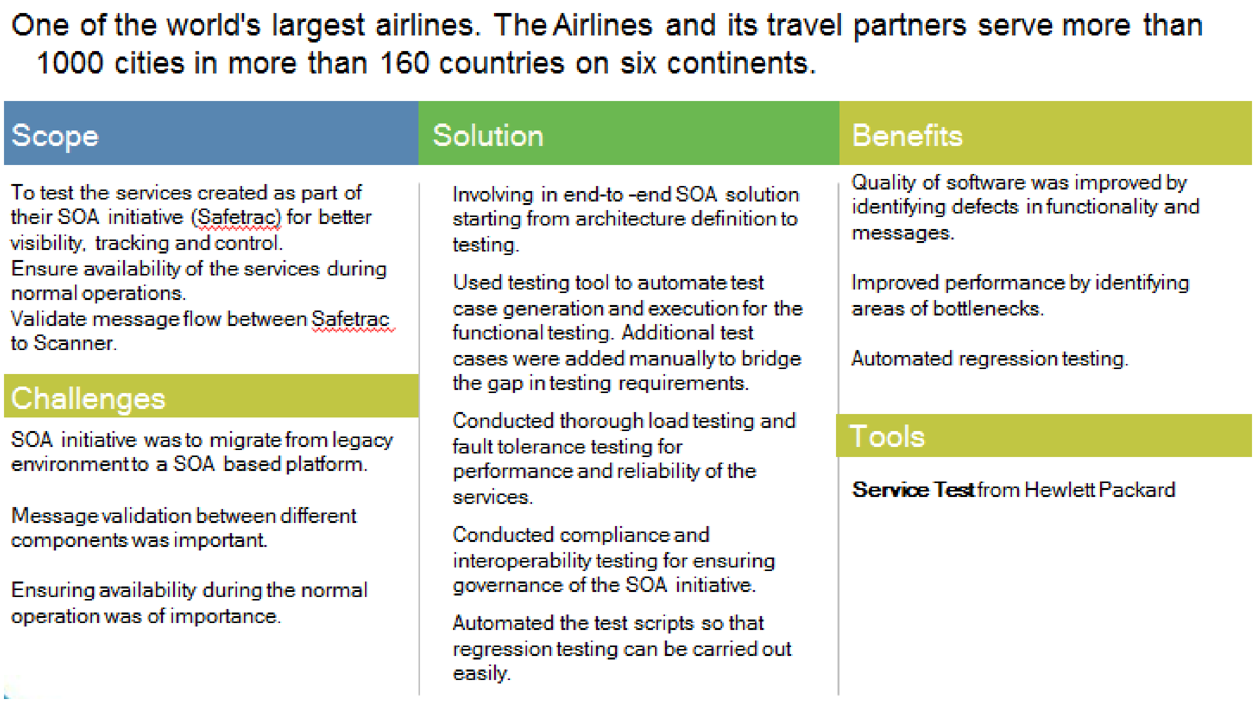
In Some projects SOA testing is carried out manually with already built in framework. This is may vary with the projects, Especially Banking projects never disclose much about the framework or allow tester to mess with it hence most of the testing is manual of course with bulk execution of test case using tools like CA LISA and HP UFT. Extensive inter-operability testing is done using SOAPUI or SOAPUI Pro version.

If automation is the need then scripting comes into picture. Each tool allows testing to script using open source scripting like Groovy, Java script etc. Some tools even allow to record and play simulation of web service in order to mock the web service.

15.0 Case Study

Case Study: Service Testing for Airlines

Client profile (Travel Industry): Service Testing for Airlines



One of the world's largest airlines. The Airlines and its travel partners serve more than 1000 cities in more than 160 countries on six continents.

Scope

To test the services created as part of their SOA initiative (Safetrac) for better visibility, tracking and control.
Ensure availability of the services during normal operations.
Validate message flow between Safetrac to Scanner.

Challenges

SOA initiative was to migrate from legacy environment to a SOA based platform.

Message validation between different components was important.

Ensuring availability during the normal operation was of importance.

Solution

Involving in end-to -end SOA solution starting from architecture definition to testing.

Used testing tool to automate test case generation and execution for the functional testing. Additional test cases were added manually to bridge the gap in testing requirements.

Conducted thorough load testing and fault tolerance testing for performance and reliability of the services.

Conducted compliance and interoperability testing for ensuring governance of the SOA initiative.

Automated the test scripts so that regression testing can be carried out easily.

Benefits

Quality of software was improved by identifying defects in functionality and messages.

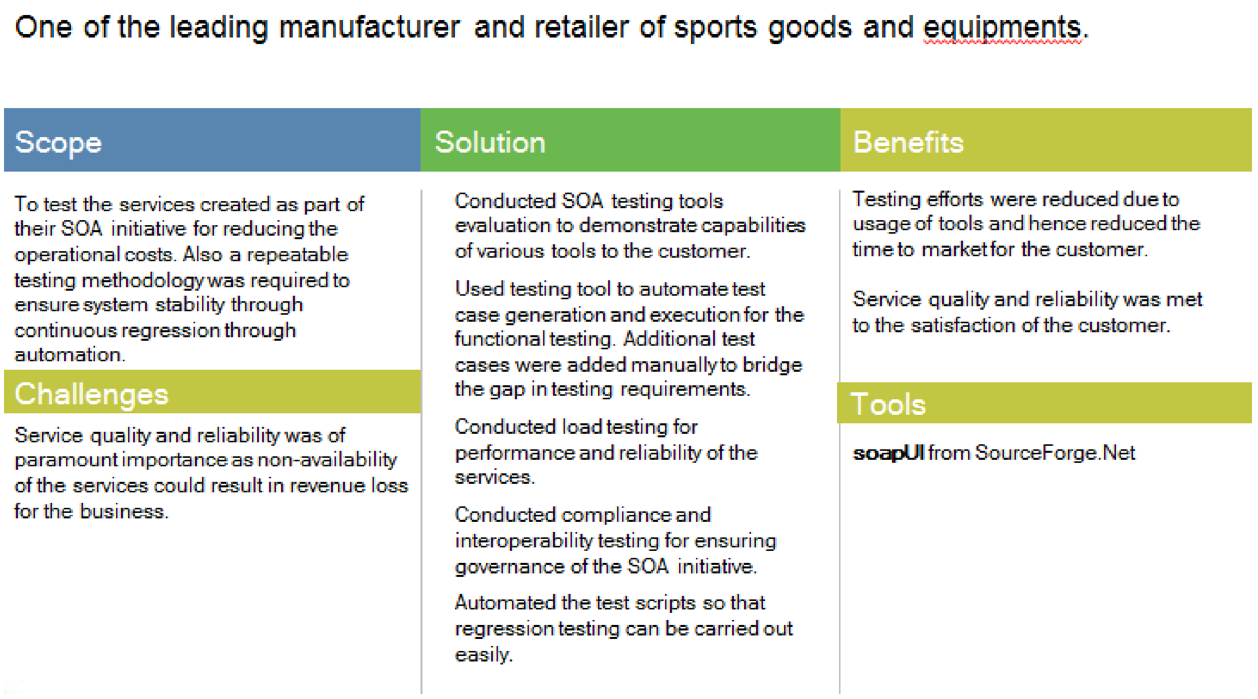
Improved performance by identifying areas of bottlenecks.

Automated regression testing.

Tools

Service Test from Hewlett Packard

Case Study: Service Testing in Retail Industry
Client Profile: (Retail industry) Service Testing in Retail Industry



One of the leading manufacturer and retailer of sports goods and equipments.

Scope

To test the services created as part of their SOA initiative for reducing the operational costs. Also a repeatable testing methodology was required to ensure system stability through continuous regression through automation.

Challenges

Service quality and reliability was of paramount importance as non-availability of the services could result in revenue loss for the business.

Solution

Conducted SOA testing tools evaluation to demonstrate capabilities of various tools to the customer.

Used testing tool to automate test case generation and execution for the functional testing. Additional test cases were added manually to bridge the gap in testing requirements.

Conducted load testing for performance and reliability of the services.

Conducted compliance and interoperability testing for ensuring governance of the SOA initiative.

Automated the test scripts so that regression testing can be carried out easily.

Benefits

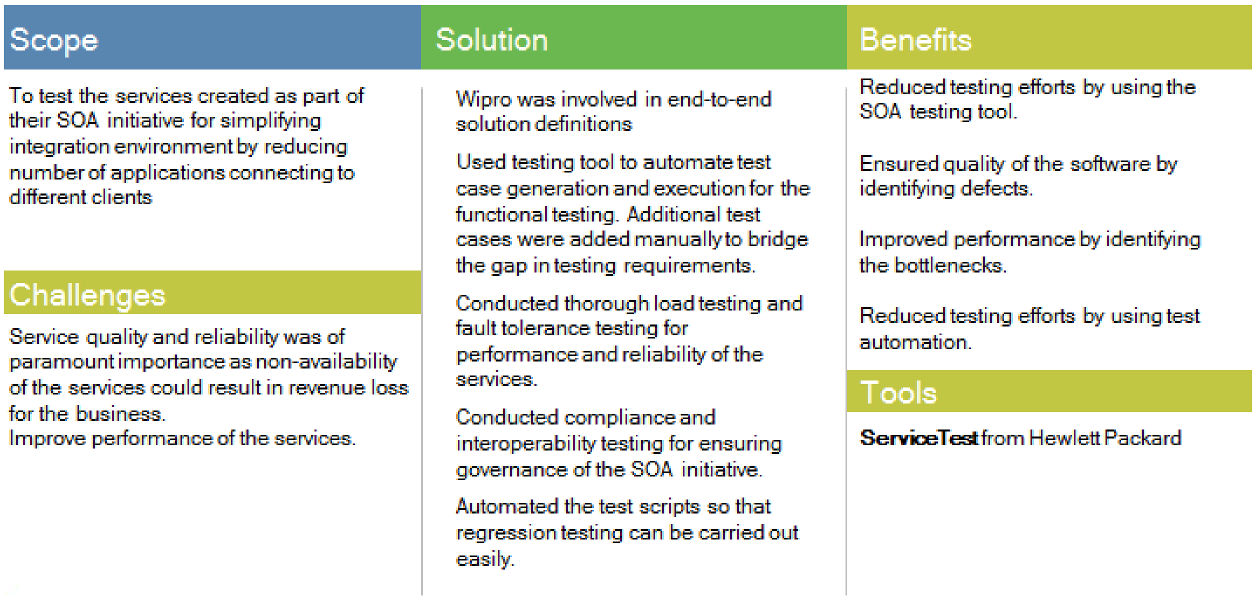
Testing efforts were reduced due to usage of tools and hence reduced the time to market for the customer.

Service quality and reliability was met to the satisfaction of the customer.

Tools

soapUI from SourceForge.Net

Case Study: Web Service Testing in Insurance Industry
Client profile: (Insurance Domain)



Scope

To test the services created as part of their SOA initiative for simplifying integration environment by reducing number of applications connecting to different clients

Challenges

Service quality and reliability was of paramount importance as non-availability of the services could result in revenue loss for the business.
Improve performance of the services.

Solution

Wipro was involved in end-to-end solution definitions

Used testing tool to automate test case generation and execution for the functional testing. Additional test cases were added manually to bridge the gap in testing requirements.

Conducted thorough load testing and fault tolerance testing for performance and reliability of the services.

Conducted compliance and interoperability testing for ensuring governance of the SOA initiative.

Automated the test scripts so that regression testing can be carried out easily.

Benefits

Reduced testing efforts by using the SOA testing tool.

Ensured quality of the software by identifying defects.

Improved performance by identifying the bottlenecks.

Reduced testing efforts by using test automation.

Tools

ServiceTest from Hewlett Packard

16.0 Conclusion

In this report, we examined the additional test areas for SOA and the need for a specialized testing methodology. The key to SOA quality is to involve business analysts, executives, and Architects as an integral part of regular quality audits, in addition to test engineers as specified in this document to ensure reuse and required agility at the enterprise level. Going by the current technology trend in SOA Testing of CICD (continuous integration and continuous deployment) and the future of DevOps, next likely technology or Buzz word in SOA testing. I am concluding my desertion report with few words on DevOps.

Future in SOA domain would be DevOps

The specific goals of a DevOps approach include improved deployment frequency, which can lead to faster time to market, lower failure rate of new releases, shortened lead time between fixes, and faster mean time to recovery in the event of a new release crashing or otherwise disabling the current system. Simple processes become increasingly programmable and dynamic, using a DevOps approach, which aims to maximize the predictability, efficiency, security, and maintainability of operational processes. Very often, automation supports this objective.

DevOps integration targets product delivery, quality testing, feature development, and maintenance releases in order to improve reliability and security and provide faster development and deployment cycles. Many of the ideas (and people) involved in DevOps came from the Enterprise Systems Management and Agile software development movements.

DevOps aids in software application release management for an organization by standardizing development environments. Events can be more easily tracked as well as resolving documented process control and granular reporting issues. Companies with release/deployment automation problems usually have existing automation but want to more flexibly manage and drive this automation — without needing to enter everything manually at the command-line. Ideally, this automation can be invoked by non-operations employees in specific non-production environments. The DevOps approach grants developers more control of the environment, giving infrastructure more application-centric understanding

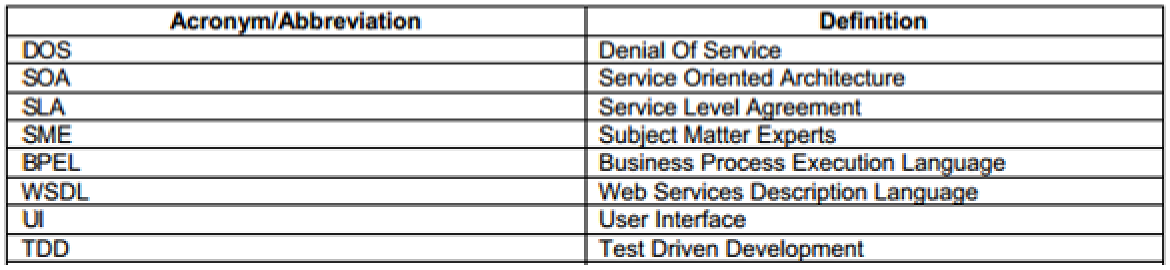
17.0 Plan of work (16 Weeks)

Task #	Tasks or subtasks to be done (be precise and specific)	Planned duration in weeks	Specific Deliverable in terms of the project
1.	Requirement gathering	3	Requirements Specification Document.
2.	Design Phase	2	Block diagrams and graphics
3.	Detailed study on SOA and testing practices	3	Contents on different sub topics
4.	Service virtualization	2	Contents and information on Service virtualization
5.	SOA testing tools	2	Report on all the industry tools
6.	Sample File execution and groovy scripting using SoapUI	1	Execution screen shots of sample services
7.	Case studies from different domains	2	Case study reports
8.	Working on comments from mid semester and Submission of final dissertation report to the supervisor	1	Project Report.

18.0 References

1. White paper from http://www.bptrends.com:
   SOA Maturity Model - Srikanth Inaganti and Sriram Aravamudan, 2007
2. Center of excellence reference guide:
   SOA Testing – Neelima Talluri,2010
3. Pre sales presentation:
   SOA testing– Jayashree Kar, 2012
4. A journal Paper :
   Transformation Testing in ESB and SOA environment – Shashi Kant Dalmia,2014
5. Knet website Paper from Wipro:
   An Introduction to SOA Testing Methodology - Dheeraj K, 2009
6. Tool/vendor Website:
   CA Service Virtualization – www.ca.com
7. Wikipedia site:
   DevOps – http://en.wikipedia.org/wiki/DevOps
8. Capital One account : Client account in Wipro:
   Working experience in banking project on SOA platform
9. SPEZI Centre of excellence:
   Training in SOA practice

19.0 GLOSSARY



Acronym/Abbreviation	Definition
DOS	Denial Of Service
SOA	Service Oriented Architecture
SLA	Service Level Agreement
SME	Subject Matter Experts
BPEL	Business Process Execution Language
WSDL	Web Services Description Language
UI	User Interface
TDD	Test Driven Development

20.0 Candidate Project Details

Title: RTM (Real Time messaging)

Client: Capital One Bank, USA

Duration: April 30, 2014 to Present

Tools: SOAPUI, HP UFT, Altova XMLSpy, HP ALM

Team Size: 9

Project Description

Real time messaging is a middleware system of Capital one Banking which is being used across various Line of business such as Commercial, Retails, Enterprise and Cards. RTM is a middle tier for obtaining transaction and customer details. The System involves REST service, Web services, and BPEL/BLS layers.

Roles and Responsibilities

- Walk through on requirements with Business Analyst/Developer and raising queries, if any.
- Gathering required information for testing from development team.
- Developing & executing test cases using applicable tools to verify the application functionality.
- Involving in Smoke Testing, System Integration, Regression and Ad hoc testing of REST service, Web service and BPEL.
- Working in Late evening shifts and actively participating in agile ceremonies like Sprint planning, Daily stand-up, Sprint Retrospective and sprint review.
- Updating the test cases in Test management tool HPALM.
- Logging valid defects in Bug tracking tool (HP ALM), Tracking the defects triage and following up for closure of defects with Development.

21.0 Checklist

a)	Is the Cover page in proper format?	Y
b)	Is the Title page in proper format?	Y
c)	Is the Certificate from the Supervisor in proper format? Has it been signed?	Y
d)	Is Abstract included in the Report? Is it properly written?	Y
e)	Does the Table of Contents' page include chapter page numbers?	Y
f)	Is Introduction included in the report? Is it properly written?	Y
g)	Are the Pages numbered properly?	Y
h)	Are the Figures numbered properly?	Y
i)	Are the Tables numbered properly?	Y
j)	Are the Captions for the Figures and Tables proper?	Y
k)	Are the Appendices numbered?	Y
l)	Does the Report have Conclusions/ Recommendations of the work?	Y
m)	Are References/ Bibliography given in the Report?	Y
n)	Have the References been cited in the Report?	Y
o)	Is the citation of References/ Bibliography in proper format?	Y
p)	Candidates Current Project Details	Y