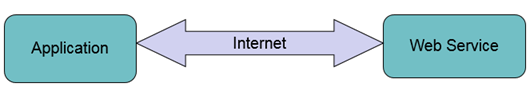
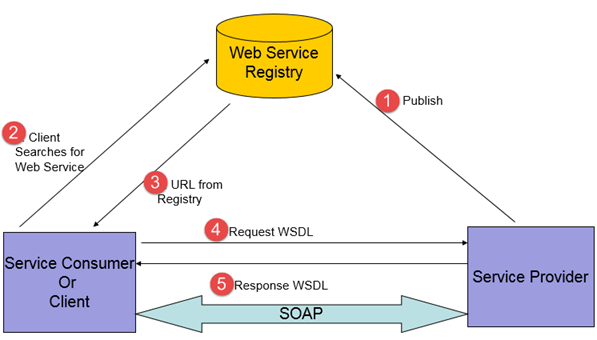
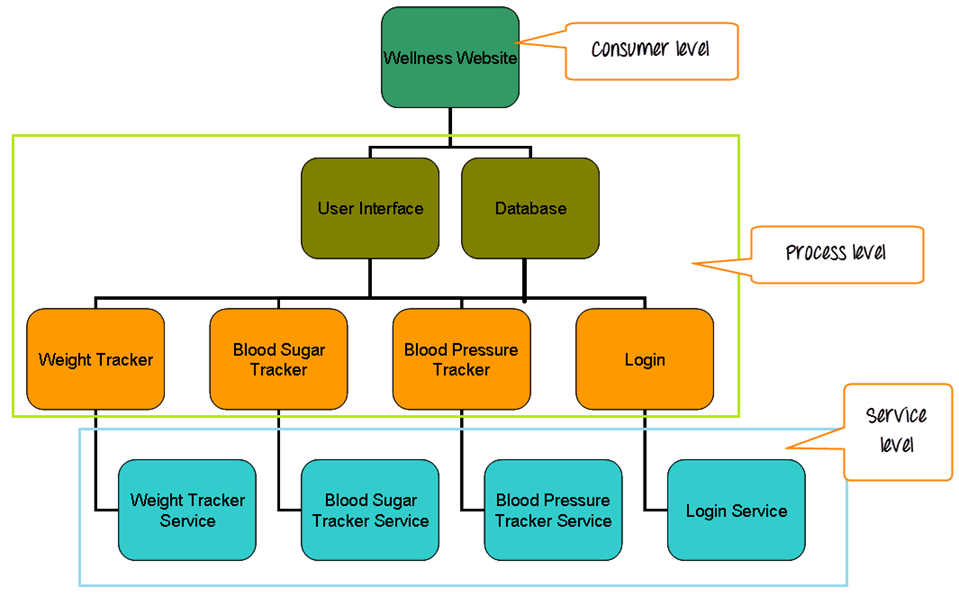
[](http://cdn.guru99.com/images/jsp/030116_0725_LearnSOATes2.png)

[](http://cdn.guru99.com/images/jsp/030116_0725_LearnSOATes3.png)

SOA Testing should focus on 3 system layers

**Services layer**

**Process layer**

**Consumer layer**

1. **Services layer** –

This layer consists of the services, services exposed by system derived from business functions.

For example –

Consider a Wellness Website which consists of

* 1. Weight Tracker
  2. Blood Sugar Tracker
  3. Blood Pressure Tracker

Trackers display the respective data and date they are entered. Services layer consists of the services which gets the respective data from the Database–

* Weight Tracker service
* Blood Sugar Tracker service
* Blood Pressure Tracker service
* Login Service

1. **Process layer** –

Process Layer consist the processes, collection of services which are part of a single functionality.

The processes might be a part of user interface (for ex – A search engine), a part of ETL tool (for getting data from the database).

The main focus in this layer will be in user interfaces and process.

The user interface of the weight tracker and its integration with the Database is the primary focus.

Below functions will be of consideration

* 1. Adding new data
  2. Editing existing data
  3. Creating new tracker
  4. Deleting data

1. **Consumer layer** –

This layer mainly comprises of user interfaces.

Based on the layer, the testing of an SOA application is distributed into three levels.

1. Service level
2. Interface level
3. End to End level

* Top Down approach is used for Test Designing.
* Bottom Up approach is used for Test Execution.

**Strategy for SOA Testing**

**Test Planning Approach,**

* The complete architecture of the application should be understood by the testers.
* The application needs to be broken down into independent services (Service, which has their own request and response structure and does not depend on any other service to form response).
* The application structure needs to be reorganized into three components – Data, Services, and front end applications.
* All the components need to be carefully analyzed, and business scenarios should be chalked out.
* The business scenarios should be classified as common scenarios and application specific scenarios.
* A traceability matrix should be prepared, and all test cases should be traced to business scenarios.

**Test Execution Approach**

* Each service component should be tested.
* Integration testing of the service components should be done to validate the data flow through the services and data integrity.
* System testing of the complete model should be done to validate the data flow between front end application and database.
* Performance testing should be done for fine tuning and optimum performance.

**SOA testing methods**

**Business scenario driven data based testing,**

* Various business aspects related to the system should be analyzed.
* Scenarios should be developed based on the integration of
  + Various web services of the application
  + Web services and application.
* Data set up should be done based on the above scenarios.
* Data set up should be done so as to cover end to end scenarios as well.

**Stubs**

* Dummy interfaces will be created to test services.
* Various inputs can be provided through these interfaces, and the outputs can be validated.
* When an application uses an interface to an external service, which is not under test (third party service), a stub can be created during integration testing.

**Regression testing**

* Regression testing on the application should be done when there are multiple releases so as to ensure the stability and availability of the systems.
* A comprehensive regression test suite will be created covering the services which form an important part of the application.
* This test suite can be reused in multiple releases of the project.

**Service Level Testing**

Service Level Testing includes testing the component for functionality, security, performance and interoperability.

Each and Every Service needs to be first tested independently.

**Functional Testing**

Functional Testing should be done on each service to

* Ensure that service delivers the right response to each request.
* Right errors are received for requests with Invalid data, bad data, etc.
* Check for each request and response for each and every operation the service has to perform in run time.
* Validate the fault messages when an error occurs at the server, client or network level.
* Validate that the responses received are in the right format.
* Validate that the data received on the response correspond to the data requested.

**Security Testing**

Security testing of the web service is an important aspect during service level testing of the SOA application; this ensures the safety of the application.

The following factors need to be covered during testing:

* Industry Standard defined by WS-Security testing should be abided by the Web Service.
* Security measures should work flawlessly.
* Encryption of data and Digital signatures on the documents
* Authentication and Authorization
* SQL Injection, Malware, XSS, CSRF, other vulnerabilities are to be tested on the XML.
* Denial of Service attacks

**Performance Testing**

Performance Testing of the service needs to be done since the services are reusable and multiple applications might be using the same service.

The following factors are considered during testing:

* Performance and functionality of the service need to be tested under heavy load.
* The performance of the service needs to be compared while working individually and within the application it is coupled with.
* Load testing of service should be performed
  + to verify response time
  + to check for bottle necks
  + to verify utilization of CPU and memory
  + to predict scalability

**Integration level testing**

* Service level testing ensures proper working of only the services individually, it does not guarantee the working of the coupled components.
* Integration testing is done focusing mainly on the interfaces.
* This phase covers all the possible business scenarios.
* The Non-Functional testing of the application should be done one more time in this phase. Security, compliance, and performance testing ensure the availability and stability of the system in all aspects.
* The communication and network protocols should be tested to validate the consistency of the data communication between the services.

**End to End testing**

This phase ensures that the application confirm to the business requirements both functionally and non-functionally.

The below items are ensured to be tested during end to end testing

* All services working as expected after integration
* Exception handling
* User Interface of the application
* Proper data flow through all the components
* Business process

**Challenges in SOA testing**

* Lack of interfaces for Services
* Testing process spans across multiple systems thus creating complex data needs
* The application is a collection of various components which tends to change. The need for regression testing is more frequent.
* Due to Multilayer architecture, it is difficult to isolate defects.
* Since the service will be used in different interfaces, it is difficult predicting load, hence making performance test planning cumbersome.
* SOA is a collection of heterogeneous technologies. Testing of an SOA application requires people with different skill sets which in turn increase the planning and execution costs.
* Since the application is an integration of multiple services, security testing has its own share of woes. Validation of authentication and authorization is pretty much difficult.

**SOA Testing Tools**

There are many tools available in the market to help testers in testing SOA applications. Few of them are listed below.

**SOAP UI**

"SOAP UI" is an open source Functional testing tool for Services and API Testing.

* Desktop application
* Supports multiple protocols – SOAP, REST, HTTP, JMS, AMF, JDBC
* Web services can be developed, inspected and invoked.
* Can also use for load testing, automation testing and security testing
* Stubs can be created by MockServices
* Web Service requests and tests can be generated automatically through its web service client.
* Have inbuilt reporting tools
* Developed by SmartBear

**iTKO LISA**

"LISA" is a product suite which provides a functional testing solution for distributing systems like SOA.

* Can also use for regression, integration, load and performance testing.
* Developed by iTKO (CA Technologies)
* Can be used to design and execute tests.

**HP Service Test**

"Service Test" is a functional testing tool, which supports both UI and shared services testing

* Both functional and performance test of services can be done by a single script.
* Integrated with HP QC.
* The massive amount of service and data can be managed.
* Supports interoperability testing by simulating JEE, AXIS, and DotNet client environments.
* Developed by HP.

**ParaSoft SOA Test**

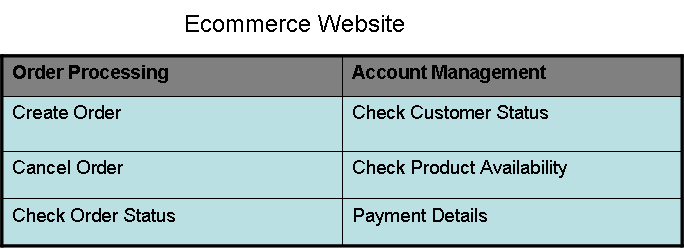
SOA Test is a testing and analysis tool suite developed for API and API applications testing.

* Supports Web Services, REST, JSON, MQ, JMS, TIBCO, HTTP, XML technologies.
* Functional, Unit, Integration, Regression, Security, Interoperability, Compliance and Performance Testing are possible.
* Stubs can be created using Parasoft Virtualize, which are intelligent than SOAP UI.
* Developed by ParaSoft

**SOA Testing Use Cases**

Consider an ecommerce Website, which contains the below functions and Sub-functions:

**Order Processing**

[](http://cdn.guru99.com/images/jsp/030116_0725_LearnSOATes7.png)

**PHASE 1** - In first phase of SOA testing i.e. Test Strategy Phase, the application is broken into Services and business functions.

Let us consider below are the Services in the application.

* Create Order
* Check Customer Status
* Change Order Status
* Check Order Status
* Check Inventory

Business functions being the same as the functions of the Website.

**Note:** The test strategy document would contain the list of the service and the functions which have to be tested.

**PHASE 2** – Test Planning phase. Test cases are written for each level.

1. End to End level. The test cases are written for each business use case and flow.

Below are the example of test cases

* 1. Create order with active user.
  2. Create order with inactive user.
  3. Create order with available product with order quantity < available quantity.
  4. Create order with available product with order quantity > available quantity.
  5. Create order with multiple items
  6. Cancel an order completely.
  7. Cancel order partially.

1. Integration level. Test cases are written for integration of database and user interface.

Below are example test cases.

* 1. Create a new order with a single item. Verify that the order is created on the database.
  2. Create a new order with single item. Verify that the price calculated for the order is correct.
  3. Create a new order with single item. Verify that the quantity of the available product is less by the order amount.
  4. Verify that the status of the order displayed on the UI is same as that on the database.
  5. Cancel the order and verify that the status of the order is modified on the database.
  6. For first time payment, verify that the payment details entered on the UI is saved on the database.
  7. For returning payments, verify that the payment details on the database are displayed on the UI.

1. Service level. Each service is tested for all the data conditions.

Below are few examples.

* 1. Create Order.

No. of Items = 1

Quantity on order < Quantity on database

* 1. Create Order.

No. of Items > 1

Quantity on Order < Quantity on database.

* 1. Create Order

No. of Items = 1

Quantity on Order > Quantity on database

* 1. Check Order status

Status on database = Active

* 1. Check Order status

Status on database = Shipped

* 1. Check Order status

Status on database = Cancelled

* 1. Check Order status

Order id = Invalid

* 1. Check product availability

Quantity of product >0

* 1. Check product availability

Quantity of product =0

* 1. Check product availability

Product id = invalid

**PHASE 3 – Test Execution**

Test Execution uses bottom up approach i.e. service level testing is done first, then Integration level and at last End to End testing.

1. **Service level** – Let us consider that SOAPUI tool is considered for testing the application.

The WSDL and URL are browsed into the test window of SOAP.

The request for each service will be displayed on the request window.

By modifying the data as per the service level test cases, requests are created for each test case.

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Request** | **Expected response** |
| Create Order. No. of Items = 1Quantity on order < Quantity on db | <productid>x2</productid><quantity>2</quantity> | <orderid>o3251</orderid><message>Successful</message> |
| Create Order.No. of Items > 1Quantity on Order < Quantity on db | <productid>y1</productid><quantity>1</quantity><productid>y2</productid><quantity>3</quantity> | <orderid>o3251</orderid><message>Successful</message> |
| Create OrderNo. of Items = 1Quantity on Order > Quantity on db | <productid>x23</productid><quantity>200</quantity> | <orderid>null</orderid><message>Unsuccessful</message> |
| Check Order statusStatus on database = Active | <orderid>o9876</orderid> | <status>Active</active><message>Successful</message> |
| Check Order statusStatus on database = Shipped | <orderid>o9656</orderid> | <status>Shipped</active><message>Successful</message> |
| Check Order statusOrder id = Invalid | <orderid>y5686</orderid> | <status>null</active><message>Unsuccessful</message> |
| Check product availabilityQuantity of product >0 | <productid>d34</productid> | <quantity>34</quantity><available>yes</available><message>Successful</message> |
| Check product availabilityQuantity of product =0 | <productid>y34</productid> | <quantity>0</quantity><available>no</available><message>Successful</message> |
| Check product availabilityProduct id = invalid | <productid>sder</productid> | <message>Unsuccessful</message> |

1. **Integration Level** – The integration level test cases are executed on the User interface and database.

* Create order with a single item –
* User opens the website.
* Goes to place order.
* Selects a valid product and quantity and saves the order.
* Message saying the Order is placed successfully should be displayed.
* User opens database and checks if the details of the order are same as that entered on the website.

1. **End to End level** – The business flows and use cases are executed on the User interface.

* Create order with multiple items –
* User opens website.
* Goes to place order.
* Enquires about a valid product and quantity, adds them to cart.
* Other valid products are added with valid quantities and the order is saved. Payment is done through a new payment method and order is placed.
* Message saying "Order placed successfully" should be displayed.
* Tester should validate that the whole flow is done without skewing of data.

**Conclusion:**

By sketching the right strategy for testing, resources, tools and compliance to provide good service, SOA testing can deliver completely and perfectly tested application.