International Journal of Computer Applications (0975 – 8887)
Volume 105 – No. 17, November 2014

Constraint Free Testing using Service Virtualization

Dharmalingam Subbiah
Principal Consultant
Center of Excellence Team
Syntel Ltd, Chennai, India

Balaji Arulmozhi
LISA Architect
Center of Excellence Team
Syntel Ltd, Chennai, India

Hariharasudhan Maruthamuthu
LISA Engineer
Center of Excellence Team
Syntel Ltd, Chennai, India

ABSTRACT

Service Virtualization is the latest trend in the software industry leveraged throughout the application development lifecycle. It has thrived recently due to its ability to address numerous constraints faced while developing and testing a software product.

It can break through delays, costs and risks imposed by dependent IT resources that are unavailable or inaccessible for development and testing of any enterprise application development or integration projects. Service Virtualization eliminates system dependency constraints by 'virtualizing' or capturing and modeling the target system's dynamic behavior, performance and data so that it reacts and responds in the same way as the live system.

Service Virtualization ensures test and development teams have concurrent, all time access to realistic test environments to shorten their release cycles. Costs for test labs, responders, and stubs are dramatically reduced, and quality is improved by testing more scenarios faster and earlier in the lifecycle

Keywords

Service virtualization, Mocking, Continuous testing, Parallel development, Unavailable and inaccessible software, Software dependency systems, Dependency constraints, Constrained software applications, Software application downtime, Test data challenges, Infrastructure cost reduction, Testing ,Third party system dependency, Stubbing

1. INTRODUCTION

Enterprise Application Integration projects deal with inter and intra components/applications dependencies during development and testing phases due to unavailable or constrained components and out of scope test data which cause delay in the delivery of the project. Some major challenges that the development and testing teams face during development lifecycle are:

Unavailable/Inaccessible systems become constrained due to production schedules, security restrictions, contention between teams, or because they are still under development. Poor Performing Downstream systems and mockups may not provide the functional behavior or performance response needed. Dealing with out of scope test data scenarios across distributed systems can be very time consuming and cost prohibitive. These are application and services that are too difficult to replicate through traditional hardware-based virtualization approaches. Developing and/or testing against cloud-based or other third party shared services can result in costly usage fees In quintessence, Service Virtualization is the product version of the practice of 'mocking/stubbing' in the development and test environments. It has enough practicality and framework to thrust forward development more swiftly, while shifting testing left in the lifecycle, so integration and release procedures can occur sooner, with higher quality and fewer risks. In this white paper, we shall look into the constraints faced during the development and testing phases of SDLC and how Service Virtualization can help both development and testing teams carry out their by leaving the system/environment constraints behind.

2. CONSTRAINTS FACED DURING THE DEVELOPMENT AND TESTING PHASES

A system may not be accessible by the development and testing teams, or the required data set may not be available in a system to proceed with the development or testing. For instance, a needed component A may be down, a component B could have corrupt data, or a third-party component may have high access cost. Efforts to replicate these environments by manually coding stubs and handling test data are expensive and unpredictable.

Service Virtualization resolves constraints by simulating the dependent systems with a high grade of erudition. It provides less expensive, 24/7 availability that respond just like the actual entity for functional and performance testing purposes at a lesser cost.

It can be very demanding to have a good test data set that addresses all the scenarios within an application. That intricacy increases as the application becomes more composite as with complex applications. To make data problems worse, testing team need to deal with the problems around negative testing, aging and maintenance of data. Service Virtualization is significant in resolving these data issues.

In order to move to a valuable agile development process, the team needs the connectivity between different systems to be intact. These could be actual systems, but would need innumerable hardware and application software to back up the massive number of agile development teams.

Service Virtualization addresses this constraint by allowing each development team, and possibly each developer, the capability to have their own interpretation of the complete application stack with all interfaces denoted.

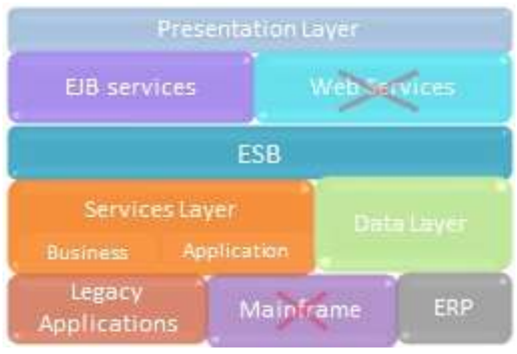

Fig 1. Environment with constraint

3. SOLUTION AND BENEFITS

3.1 Accelerated development and testing

When development and test teams work in parallel, the overall software lifecycle reaches a complete new level of efficiency and effectiveness.

In parallel development and testing, virtual services act as the intermediate assets between the system under development and the system under test, in a cooperative fashion. For instance, assume that team A is developing an account balance service while team B is developing and testing the banking smartphone application. A virtual service is captured from the existing account balance service as an initial back-end for the banking smartphone application's testing activity. Then, as the testing continues, the banking smartphone application team can provide feedback about any unforeseen or new response requirements as feedback. These feedbacks are taken as input for virtual service requests that basically turn out to be the next set of requirement for development. Each simultaneous development and test cycle endures to hasten the each iteration of the Virtual service model and, feedback ensures that the updates happen with every new build. In case the team could not access the services or the service does not support the components, they can switch to virtual services. When there is a fresh build breaks or to use new data scenarios, they can switch back to live service mean while feedback will be received for virtual service update.

3.1.1 *Anticipated Benefits:*

Test and development cycles can be completed at a fair speed. Avoid deadlock situation and facilities continuous integration and testing around business requirements. Improved degree of issue acceptance and resolution prior to production.

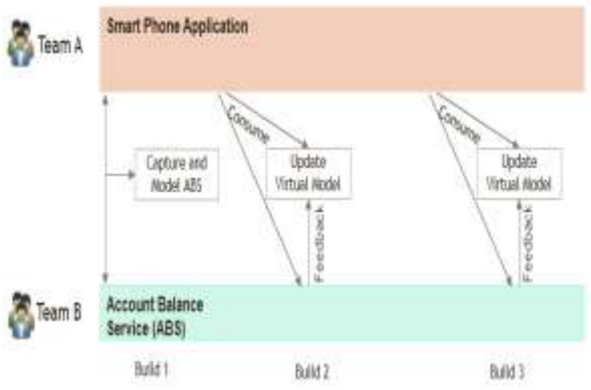

Fig 2. Accelerated development and testing

3.2 Creation of environments that replicate production systems

In the orthodox approach, teams try to advance with their own component development by "stubbing" the next downstream system only. For instance, when developing a mobile UI, the team would build a stub for a few expected responses from the immediate downstream component (E.g.: Security Gateway). Then the Security Gateway developers might stub out the next immediate downstream, component (E.g.: Oracle Security Stack) or try to simulate some user requests from the mobile UI Unfortunately, this is a manual process that is never adequate to capture many types of connections and data available within enterprise software architectures, and may be totally unattainable if a UI is not yet coded. Virtual services capture real time data scenarios and behaviors effectively which eliminates dependencies on the actual components. It also reduces the setup and infrastructure cost of setting a testing environment. Virtual services can be deployed without affecting the setup of the existing test or development environment.

Hence, there is no risk in deploying and managing virtual services in testing environment.

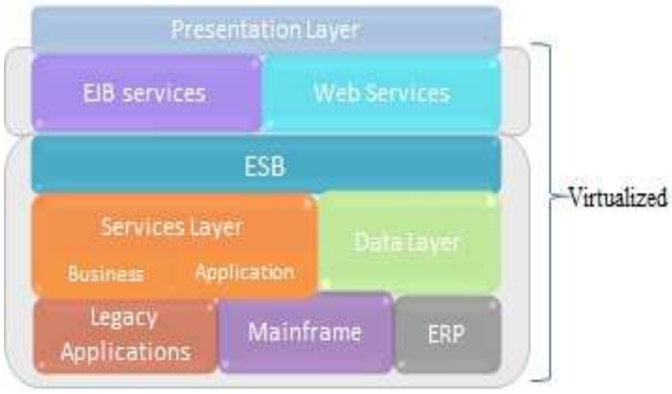

Fig 3. Environment replicates production

3.2.1 *Anticipated Benefits:*

Even if a specific component is down in an environment, development and testing can continue. The timelines of test execution can be reduced. Test coverage ratio can be increased as system dependencies are minimal. Coding quality can be improved due to the increased test coverage

3.3 Handling out-of-scope test data

Often, in testing scenarios, business processes that require access to third party services or interfaces that are out of scope or out of reach of the testing effort. Consider that a Point of Sale (POS) application is tested, that depends on a credit card validation or processing service. While the focus is on testing the developed application, there is no practical or cost effective way to test the application with the credit card validation service integrated. That service is not owned or controlled by the testing team, however, there may be several business processes in the application that might depend on it.

The common practice in these situations is to stub the required functionality and often simply skip it, reducing the testing scope, accuracy, value etc… While the goal is not to test the interface or the third party service, the ideal situation is to have it participate in the functional/performance test in support of the business process that would be of interest in testing. Simple stubs are often too static, require development time and do not properly represent the service they are designed to emulate.

Service virtualization has the ability not only to replicate the functional behavior of the interface but also the performance behavior. Out-of-scope downstream components related scenarios are recorded as virtual services. Hence, Service Virtualization eliminates the problem of missing or unavailable data behind the in-scope components.

With virtual services the teams will always have access to datasets that cover all the valid scenarios to ensure that intensive regression testing and functional testing is carried out.

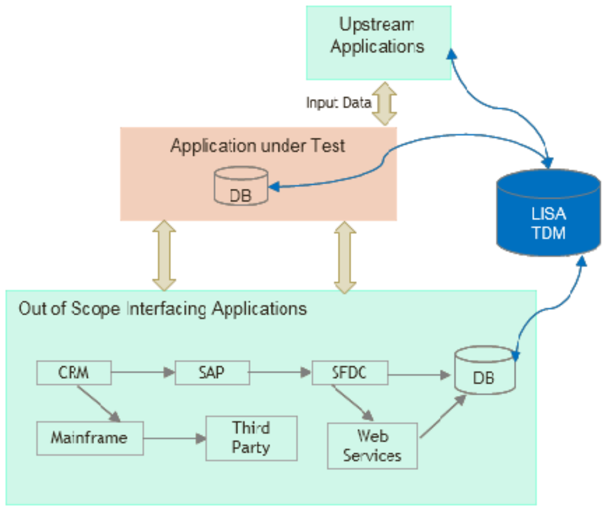

Fig 4. Handling Out-of-Scope Test Data

3.3.1 Anticipated Benefits:

No delay due to dependency on third party systems or data which is not available for testing. There will be no test data conflicts with the other testing teams as the data inputs would be provided by the independent test team. Impact on live systems is minimal. Test execution timeline is reduced as less time is required to setup the data.

3.4 Technology support for various platforms

Every project team spends a lot of money on building the infrastructure in development, testing, pre-production and production environments. Most applications in the bank are hosted on multiple environments.

The biggest challenge all organizations face is that the pre-production environments are never complete in most situations.

Many systems are used in enterprise IT environments. Hence, service virtualization should be implemented to virtualize all the components that would impact on the system under test. This includes web traffic (HTTP requests), databases, mainframes, web services (SOAP/XML, REST/JSON) and other third party components.

3.4.1 Anticipated Benefits:

Cost in setting up the pre-production environments can be massively reduced significantly. System testing and unit testing can be given preference to ensure that the quality is good even before the SIT and UAT test begins.

4. VIRTUALIZATION APPROACH

Service virtualization involves the imaging of software service behavior, and modeling of a virtual service to stand in for the actual service during development and testing cycles.

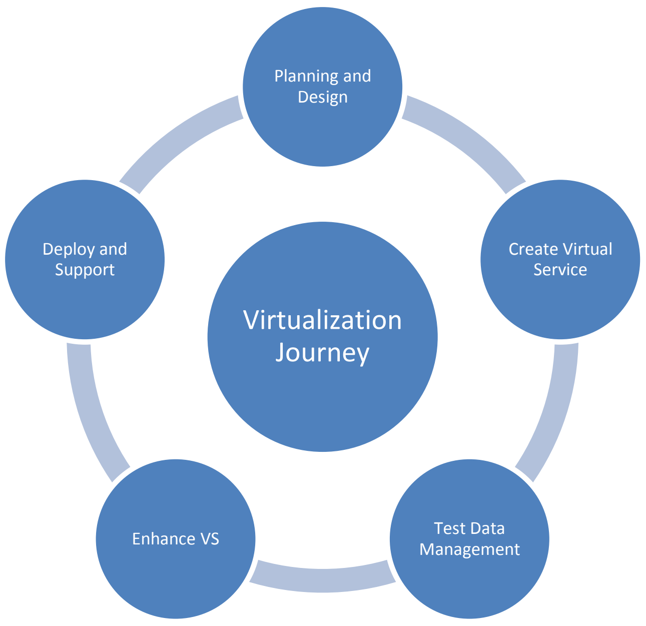

Fig 5. Virtualization Approach

Before selecting a service for virtualization, benefits will be studied. Checklist provided in 'Best Practice' section will guide users to select appropriate target services.

Virtualization approach comprises of five phases starting with 'Design' phase followed by 'Create Virtual Service', 'Test Data Management', 'Enhance' and ends with 'support' phase

4.1 Planning and Design:

Design involves understanding the scope and prioritization of the services to be virtualized for each identified services.

4.1.1 Understanding requirements:

The Virtual Service developer will start virtualization by understanding the proposed functional and non-functional requirements. This will be followed by identifying desired development and test environments, examining all the processes that consume the planned services for virtualization, listing challenges/concerns of the services/components, group all the use cases and map with identified services.

4.1.2 Prioritize the services to be virtualized:

The services to be virtualized would be prioritized based on the development needs. This information will help the team align the virtual service development timeline with the development timeline/schedule to ensure improved agility.

4.1.3 Preparing Design document:

A design document comprises of the following information: Details of Transport and Data protocols to be used for the various services to be virtualized. Port details reserved for each service will avoid conflicts while deploying multiple services in virtual service environment. Request response details mapped. List of request and response fields identified for parameterization and test data management. Test data required for running virtual services for consumption.

4.2 Creation of Virtual Service:

This phase involves creation of virtual services and population of data. There are different approaches to construct virtual services depending upon the type of services and their existence.

Several tools are available in the market for Service Virtualization. Each tool has its own benefits and supports various ways to create virtual services.

For instance, CA LISA tool allows users to create virtual services by recording the transactions of the target service and uploading the request/response pairs.

4.3 Test Data Management

Test data management is pivotal in reducing the time spent on test data preparation and reducing risk of a compliance or security breach through inappropriate use of test or production data.

As best practices in test data management, call for the test data to retain the proper data structure for testing while parameterizing or masking the data for security purposes. MS Excel is used for preparing the data sheets.

4.4 Enhance

Developers, testers or partners may consume and utilize the Virtual Services which are available in an environment to conduct testing when needed. Activities involved in this phase include: Creation of new transactions depending on a new requirement. Build custom extensions, programs if required. Deploy the Virtual Service into the Virtual Service Environment. Configure application and Test Virtual Service for all identified scenarios.

4.5 Support

Provide training to the project front end team. Update or enhance the virtual services, setup test data, based on the change request. Support project team during regression testing. Closely work with the project team in case of any issues identified during regression and the identified issues should be fixed.

4.6 Tools for Service Virtualization

List of tools which supports Service Virtualization are listed below:

Table 1: Tools available in market

Sl No	Tool Name	Vendor	Supported Technologies	Supported Protocols
1	LISA SV	CA	Web Services, XML, IBM Web Sphere, SAP PI/XI, JBoss, TIBCO, , MQ Series, Progress Sonic, Sun JMS/JCAPS, any J2EE Container, etc.	HTTP(S),IBM MQ, JMS, TCP/IP,JAVA, SOAP, etc.
2	Green Hat/RT VS	IBM	TIBCO, Software AG web Methods, SAP, IBM, Oracle, JMS-based Middleware, etc.	HTTP, TCP/IP JMS,IBM MQ, Sonic MQ, Healthcare - HL7, HIPAA, Financial Services - FIX, etc.
3	HP SV	HP	Web Services, MQ, JMS, TIBCO EMS, IMS Connect, CICS, SAP (XI/PI, RFC, and IDoc), etc.	HTTP(S), JMS, JDBC, SAP, MQ, QRACLE, SOAP, etc.
4	soapUI	Open Source	Web Services	SOAP, JSON

5. CONCLUSION

Several project teams depend on other teams to complete development and/or testing activities. There are lots of dependencies on various teams when developing or testing a software project.

Ultimately, a team cannot complete its job until the other teams finish their jobs. Each team involved with a complex application must be able to build its own virtual component from the infrastructure. There will inevitably be intra-team dependencies. This is why, service virtualization abilities are very important. Each team can take the interface specification document of the relevant downstream component and build the expected responses of the downstream component, even before the first build of the system under test is ready.

A well-versed approach to service virtualization will take into consideration not just both the technical and operational fine points but also the operational ones that can assure widespread acceptance and accomplishment.

6. REFERENCES

[1] http://www.itko.com/solutions/constraints.jsp

[2] John Michelsen and Jason English “Service Virtualization - Reality is Overrated “September 25, 2012

[3] SOA Magazine http://www.servicetechmag.com/I10/0907-2

[4] Service virtualization” http://en.wikipedia.org/wiki/Service_virtualization”

[5] Service Virtualization https://www.mulesoft.com/resources/esb/service-virtualization

[6] CA LISA Virtualization http://itroisolutions.com/wp-content/uploads/2014/02/acs2717-ca-lisa-service-virtualization-ds-08121.pdf

[7] Service virtualization is helping organizations realize business value from testing http://www.ovum.com/service-virtualization-is-helping-organizations-realize-business-value-from-testing/

[8] Service Virtualization: The Road to Simplification http://www.developer.com/tech/article.php/3678296/Service-Virtualization-The-Road-to-Simplification.htm