IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 2, Ver. V (Mar – Apr. 2015), PP 88-92
www.iosrjournals.org

Continuous Testing of Service-Oriented Applications Using Service Virtualization

Shrimann Upadhyay[1], Hrishikesh Mukherjee[2], Arup Abhinna Acharya[3]
[1](School of Computer Engineering, KIIT University, Bhubaneswar, Odisha)
[2](School of Computer Engineering, KIIT University, Bhubaneswar, Odisha)
[3](School of Computer Engineering, KIIT University, Bhubaneswar, Odisha)

Abstract: *Service-Oriented Architecture (SOA) has changed the way business enterprises get aligned with technology with a very fast pace keeping the demand of re-alignment time very short. Service-oriented architecture (SOA) carries significant commitment to impart an effective version for bias, agile and componentized IT which can be modified quickly. SOA removes the gap between software and business. The service-oriented application can be expensive to test because services are hosted remotely, are potentially shared among many users, and may have cost associated with their invocation. As classical testing approach and tools are not able to fit well, we have evolved a new approach to confront the increased complexity of the software. SV (Service virtualization) is the recent idea in software industry that grasp throughout the SDLC, it grow vigorously due to its artistry to address numerous constraints faced while development and testing. Service virtualization removes system dependency deadlocks by virtualizing or capturing and configuring the objective of the system's dynamic behavior, performance and data which acts and replies in the same manner as live one. The main benefit of using service virtualization is that the cost and time for testing is dramatically reduced, and quality is improved.*

Keywords: *Service-oriented Architecture (SOA); Service virtualization (SV); continuous testing; software dependency; unavailable and inaccessible components; dependency constraints; infrastructure cost reduction; third party system dependency.*

I. Introduction

Service-Oriented Architecture (SOA) is a technology as well as a paradigm of designing a software system to provide services to either end-user applications or to other services distributed in a network. It is a way of designing, deploying and managing systems [1]. SOA has changed the way of business enterprise get aligned with technology with swiftness keeping the desire of re-alignment time very short. SOA makes promises that include increased agility, larger application lifecycles, better integration at lower costs. However, in reality these vow are difficult to deliver.

In SOA, services encapsulates reusable business functionality with platform-independent interface contracts. A service can be vigorously located and invoked. Service-oriented computing promises exceptional flexibility and efficiency in application development by enabling applications to be composed using third-party services [1]. The common misconception for testing SOA-based applications is that it is no different than testing non-SOA applications. However such applications are difficult to test. Hence, we have evolved newer methodologies and approaches to address these increased complexity.

Service virtualization is the latest idea in software industry that grasp throughout the SDLC. It can find delays, costs and risks enforced by dependent IT resources that are insufficient or unreachable for development and testing of any enterprise application development or integration project. Service virtualization abolish system reliance restraint by virtualizing or apprehending and replica of the objective applications zestful behavior, execution and statistics so that it behaves and reacts in the same way as the real one. Service virtualization ensures development and test teams have concurrent all time entry to logical test situation to contract their release cycle. Some parameters are decreased severely such as Costs for labs to perform test, repliers, and documents and quality is upgraded by examining more situations quick and untimely in the process of the lifecycle [2].

The action of service virtualization initiates with pointing-out constrained resources. The betrothal lead must point-out a recourse which is curb by cost or obtainbility. The procedures for pointing-out a well build objective methods for service virtualization(SV) are [2]-

- Identifying methods which requires distributed opportunity to use between multiple teams like database systems.
- Recognizing methods which have high use cost per operation like SaaS systems.
- Recognize methods which are not available for testing or have their autonomous phenomenon timelines and arrangements.

- Recognize methods having complicated data management problems, like organizing test data over a number of distributed systems to perform the desired test methodology.

Next a effective configuration of the method requires to be initiated. It can be done by arranging the virtual artifact to analyze the input and output of objective of the system for clear-cut modeling behavior. E.g: a typical procedure to begin a virtual-service is to scrutinize the WSDL.

The final step is the classification of the model in the practical(virtual) environment. Structural modification to the model is done, such as reforming the data or accomplishment characteristics as required. The model(virtual) will now act in the same method as its original world equivalent and may be used in case of testing.

II. Service Virtualization An Alternative To Mocking And Stubbing

Mocking or stubbing via injecting dependency is often a chaotiv attempt. Simplest of applications it often involves creating new interfaces, taking on dependency, and adds a lot of innecessary complexity. After that, one still have to write the mocks or stubs themselves, and many of that mocks or stubs were aren't useful in anything besides superficial unit tests. Service virtualization offer an alternative approach. Rather than mocking or stubbing individual classes, one can mock entire service. From the applications perspective it is talking to a real service, even though in reality that service may not even exist.

Two usual procedure of setting up a virtual service. First one is to initialize with the agreement (i.e. a WSDL or other protocol specific service descriptor) and create responses. This can be done manually using Java or .NET code. Advantages of this procedure is that the team doesn't have to wait for real service to be completed. The downside is the virtual service needs to actually match the real one. The other procedure is to use traffic recording. A tool sits between the component under test and its downstream dependencies, which basically acts like a proxy, collects information about component interact, their message request/response, performance attributes and data. Later on those recordings will be used to mimic the talk between the component and services.

III. Challenges Faced During Testing Phase

The unavailability of environment and components for testing purpose is common challenge faced by most testing teams, because a lot of delay is associated with the acquisition, installation, setup of testing infractures, and in gaining access to external and dependent systems. All massive companies which are dependent on IT must manage with the limitations of systems like mainframe, elements under processing, ERP's and delicate data sources which hold-up the projects and having cost associated with their invocation.

Dealing with out-of-scope test data structure throughout distributed methodology could take more time and exorbit cost. Service-virtualization(SV) ponders the concept of virtualizing all data which are missed and components to simulate their behavior with enough intelligence so that the application under test (AUT) believes that it is talking with a live system. It provides 24/7 availability to AUT like actual system for functional and performance testing purpose at lower cost.

IV. Capabilities Of Service Virtualization And Expected Benefits

Service virtualization gives number of capabilities to provide high standard implementations swiftly to the market, with competitive cost and modest risk.

1. Creation of virtual dependent production system in development and test environment

According to conventional approach or orthodox method, teams attempt to proceed forward having their private component progress by "stubbing" the next downstream method. At the time of developing a web UI, the team will build stub for same anticipated answers from the downstream surface (i.e. web service) after which the service developer may extinguish the principal downstream ESB layer, or endeavor to figure-out some user requirements from the web UI. Miserably, it is a non automatic process which is never enough to abbreviate the different types of connections and software architectures, and may be totally unattainable if a UI is not yet coded [3].

Teams working with real data scenarios and real behavior captured as virtual services, their productiveness is higher, because the resulting environment is far more realistic and current than set of stubs that are manually coded and maintained [3].

1.1. Expected Benefits

Even if some components are unavailable or down, development and testing will continue. Cycle time for test execution will be reduced. With reduced data dependency on other systems and applications improves test coverage in less time and improves code quality due to increased test coverage.

2. Simultaneous Development and Testing

The altogether software lifecycle may achieve a whole new degree of efficacy and efficiency when both the teams of testing and development works simultaneously or in parallel [3].

In case of parallel development and testing, effective(virtual)services behave as the "in-between" properties between the system which is under development and system which is under testing. For instance, suppose that team B is developing a inventory management service while team A at top is developing and testing an online shopping application. A virtual service is recorded from the existing inventory management service as an inceptive sub-structure for online shopping application's testing phase. Then, as we further move-on in testing, the online shopping application team A can provide feedback about any unexpected or new retaliation requirements as feedback. Those feedbacks were taken as input for virtual service requests that turns out to be the succeeding set of needs for the progress. Each parallel development and test cycle experience to speed up every emphasis of virtual service model(VSM) and feedback ensures that the updates happen with every new build. In case if the team lost access to the services, or the services would not help the element, one can change to virtual services.

2.1. Expected Benefits

Increased speed of test and development cycle. Continuous integration and testing around business requirements by avoiding deadlock conditions. Deliver function points at a fair speed, with higher quality and accuracy to specification.

3. Handling data for Out-of-Scope dependencies

In testing scenarios, enterprise business process which requires access to third party services or interfaces that are out of scope or out of reach of testing effort [3]. For instance, consider a online ticket booking application is under testing which depends on online payment validation or processing service. The test team focuses on testing the developed application, there is no practical or cost effective way to test the application with the online payment validation service integrated. The online payment service is not possess or dominated by testing team, but there might be some business processes in the application that may depend on it. The most obvious solution in this stage is to stub the expected functionality and simply skip it, which reduces the testing scope, accuracy, etc. But the goal is not to test the third party service, with suitable situation is to have it engage in the functional/performance test in support of enterprise business process that would be of interest in testing. Stubs are too steady, which requires additional development time and effort.

Service virtualization has the ability not only to virtualize the functional behavior of the interface but also the performance behavior. Downstream out-of-scope component related scene are captured as virtual services. Hence, service virtualization eliminates the problem of missing or unavailable data behind the in-scope components.

3.1. Expected Benefits

Eliminates delay due to dependency on third party services or systems, data which is not available for testing. It gives 24/7 availability of test scenarios for development and test teams. Impact on live system is minimal. Test execution timeline is reduced as less time is required to setup the data. No conflicts over test data with other testing teams as the data input be provided by independent test teams.

4. Heterogeneous technology and Platform support

Most in the cases project team spends a lot of money on creating the infrastructure in development, testing, pre-production, and production environments. Many applications of financial sector were hosted on multiple or heterogeneous environments [3].

The greatest challenge faced by many organization and project teams is that the pre-production environments are never the complete systems.

Heterogeneous systems are used in business IT surroundings. So, service-virtualization(SV) can be used to conceptualize any and all reliance that would have effect on system under test (SUT). The system includes web traffic (HTTP), web services (SOAP,XML), integration layer and ESB and other third party services.

4.1. Expected Benefits

Reduced cost in implementation or setting-up of the pre-production environments.

V. Approch To Implement Service Virtualization

Service virtualization creates the virtual mock of software service behavior, and those mocks will stand in as real services during development and testing cycles.

Benefits of service virtualization and virtualizing any service will be taken care before starting the execution (implementation). Most of the obstacles and bottlenecks can be solved by attentive and well marked implementation approach. Service virtualization supports inherent agile lifecycle of current composite application approach. The virtualization implementation approach starts with requirement management phase followed by analysis and design phase, implementation (development and test) phase and end with deployment and management phase [4].

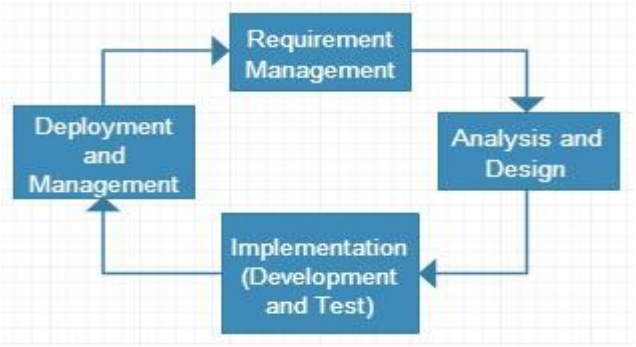

Fig. 1- Service Virtualization Lifecycle

A. Requirement Management

The virtualization team will need to recognize the proposed functional and non-functional aspects. Teams have to believe the running IT-landscape and wishing development and test peripherals. The teams at this phase will need to understand all the use-cases and clearly demonstrate the ideal target for service virtualization.

Systems and services having access constrained which are available at determined times but play an fundamental role in the process of business for development and enhancement (i.e. mainframe) virtualization is a solution to improve and intensify the availability and reduce the total elapsed time [4].

The teams understand well with the requirements for virtualization targets. In case, the developing service is completely new and having no pre-existing data then teams may require to know how data are authorized and the artificial data creation.

B. Analysis And Design

On the basis of collected requirements, the teams plan, prioritize and document the virtual services. First, the teams concentrate on services(utility) and elements pointed-out for virtualization and the use-cases associated with them. A variety of design artifacts can be used to link the use case with the services which include sequence diagrams and service definitions. The services to be virtualized would be prioritized based on development and test teams needs. This prioritization will help development and test teams timeline/schedule to ensure improved agility [4].

At the end of this phase a design document is prepared contains information-Details of protocols used by virtual services, port details of virtual services, request/response details, list of request/response fields identified for parameterized and test data management, and test data for virtual services.

C. Implementation (Development And Test)

The implementation phase includes formations of effective methods(virtual services) and categorized them with data. On the basis of development/test requirements, teams start implementation of virtual services. There are different perspective of service virtualization rely on types of services and their behavior.

Now a days, many service virtualization tools are available in market, with their benefits and support different numerous ways for creation of virtual services.

For instance, the open source soapUI tool creates mock stubs or services using service interface or WSDL (i.e. web service description language) and CA LISA creates virtual services by placing a recorder which works as proxy and records the transactions of the desired objected service and upload the pairs of request and response [5].

D. Deployment And Management

The virtual services created or implemented are now designed and positioned in virtual containers and may be used in development and testing when needed.

The virtual services and images are used in iterative and incremental lifecycle projects just like SOA. So, the virtual services commonly requires to be tweaked to reach the rigors integrated co-relations.

Virtualization teams expects to have a look in the change in virtual images as new behavior are taken in to consideration with proceeding through every iteration.

VI. Service Virtualization Tools

Tools which are available in market for service virtualization are listed [6]-

Table 1- Tools for Service Virtualization

S.no.	Tool Name	Vendor	Supported Technologies	Supported Protocols
1.	LISA SV	CA Technolog-ies	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.	Parasoft Service Virtualize	Parasoft Technologies	Web service, XML, TIBCO, SAP PI/XI, J2EE container, etc.	HTTP(s), JMS, TCP/IP, JAVA, SOAP, etc.
3.	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.
4.	HP SV	HP	Web Services, MQ, JMS, TIBCO EMS, IMT Connect, CICS, SAP (XI/PI, RFC and IDoc), etc.	HTTP(s), JMS, JDBC, SAP, MQ, ORACLE, SOAP, etc.
5.	soapUI	SmartBear	Web Services	SOAP, JSON

VII. Conclusion

Many times while testing service-oriented application as it is iterative and incremental development and relay on agile methodology the teams depends on other teams for completion of development and testing activities. A lot of dependencies on various teams and third party services while testing service-oriented applications.

Basically, a team cannot complete its task until the other team finish their tasks. Every team is having their own set of tasks when they are developing a complex and distributed application. In the situation, every teams must be able to virtualize their own virtual components from the infrastructure. This is how, service virtualization is important. Each team can take interface specification document or design document and virtualize the downstream components or build the expected responses of the downstream components, even before the first component or iteration is ready for testing.

Hence, service virtualization(SV) helps to provide a necessary environment for business application development and testing. Using service virtualization in the development and testing lifecycle will decrease the overall time spent in development and testing of service-oriented application.

References

[1]. Thomas Erl, "Service-oriented Architecture: Concept; Technology and Design", Pearson Education Publishing, 2nd Edition, 2013.
[2]. Gaurish Vijay Hattangadi, Rajiv Gupta, "Service Virtualization for Modern Application", Infosys White Paper, April 2011.
[3]. John Michelsen, "Key capabilities of a Service Virtualization Solution", ITKO-Infosys White Paper, July 2011.
[4]. Gaursih Vijay Hattangadi, "A Practitioner's Approach to Successfully Implementing Service Virtualization", Infosys White Paper, Sep. 2011.
[5]. CA LISA Virtualization http://itroisolutions.com/wpcontent/uploads/2014/02/acs2717-ca-lisa-servicevirtualization-ds- 08121.pdf
[6]. Dharmalingam Subbuah, Balaji Arulmozhi, Hariharasudhan Maruthamuthu, "Constraint free testing using service virtualization", Internatioanl Journal of Computer Applications, vol. 105- no. 17, November 2014.
[7]. Gaurish Vijay Hattangadi, "A Practitioner's Guide to Modern SOA Testing", Infosys White Paper, July 2011.