Building
Tomorrow's Enterprise

White Paper

A Practitioner's Approach to Successfully Implementing Service Virtualization

The Final Piece of the Puzzle

Gaurish Vijay Hattangadi

Executive Summary

Service virtualization envisions a promising solution to alleviate dependency constraints. However, its implementation can pose confusing questions for IT architects. Success in virtualization initiatives comes from understanding the three W's (Why virtualize, What is the virtualization process and When to virtualize) and one H (How to virtualize). Previous papers in this white paper series have already helped us address the questions that pertained to the "Why" and "When" of Service Virtualization[1]. This paper focuses on the "What" of the virtualization process and also lays out a practitioner's approach/ methodology to successfully implementing Service Virtualization.

Sep 2011

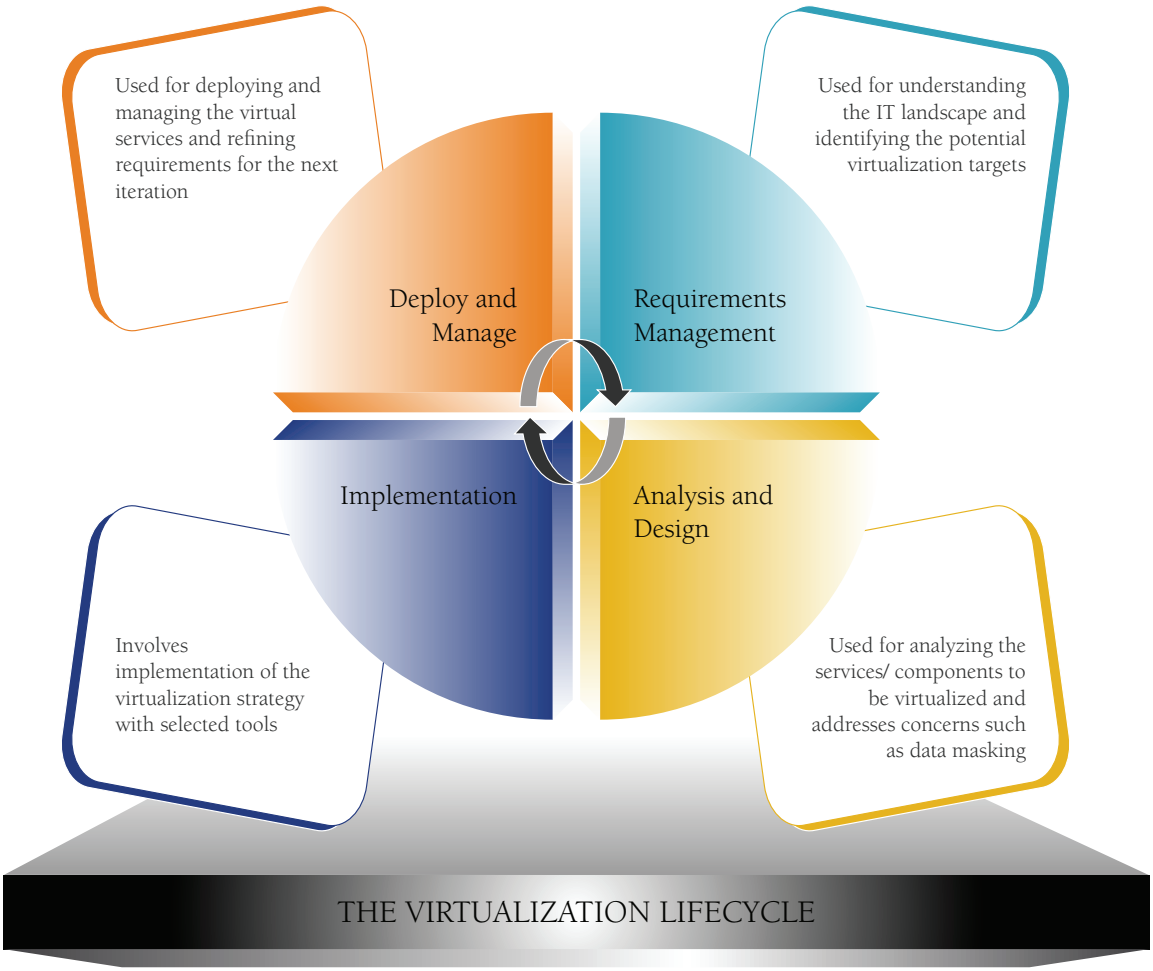
The Virtualization Methodology

In earlier papers, we discussed the factors that lead to the creation of service and hardware virtualization[2]. These complimentary solutions are designed to alleviate problems posed by over utilized and underutilized systems. Service Virtualization is a promising solution; unfortunately IT architects face numerous hurdles in its implementation. Most of the hurdles can be resolved by a rigorous and a well-defined virtualization lifecycle. The virtualization lifecycle suits the inherently agile lifecycle of today's composite application approaches, which includes SOA, BPM and Cloud-based apps, since it is iterative and incremental. Similar to a SOA development lifecycle, the virtualization lifecycle starts with the requirements management phase followed by analysis and design, implementation (develop and test) and ends with deployment and management. Figure 1 illustrates this concept – the inner pie displays the phases of the SOA lifecycle and the outer boxes describe the intent of the virtualization teams at each phase of the lifecycle.

IMPORTANT

We assume that your virtualization and SOA consultants have already helped you select the right virtualization solution*. This along with the usage of a mature SOA testing process and good SOA testing tools are prerequisites for an effective virtualization. Most of these aspects have been discussed in the previous whitepapers of the series.

Figure 1: The Virtualization Lifecycle



Requirement Management Phase – "What to Virtualize"

This phase of the software development lifecycle is used to understand the proposed functional and non-functional requirements. Virtualization teams use this phase to understand the current IT landscape and the desired development and test environments. Once infrastructural requirements have been understood, the virtualization leads then identify the environment build strategies. There are always a few components in any environment which have very simple behaviors with low test data management complexities and negligible access constraints. These have low ROI and are therefore not ideal targets for service virtualization (except as stubs for "happy path" technology demonstrations). Experiences with large engagements clearly demonstrate that ideal targets for virtualization are:

1 Services and systems that are access constrained

- Systems available only at specific times but are an integral part of the business process being developed or enhanced. An often quoted example here is that of a mainframe based underwriting engine that is part of a pre-sales quote generation process undergoing enhancement. The underwriting engine is only available for testing at specific times and is also expensive to refresh (data refresh). Virtualizing this component is a tempting solution for its increased availability.
- Shared component dependency: This is an often cited problem which involves N-Tier architectures that exposes domain models (business logic) through services for consumption in user interfaces. This component dependency forces project managers to adopt a critical path where the business logic and associated services are developed before the user interfaces. Creating a virtual service changes the critical path chain as UI development teams can begin working earlier and in parallel with the service development teams. The reduction in the elapsed project time can be a significant advantage as well.

2 Complex Enterprise Services

- Enterprise SOA initiatives lead to the creation of enterprise IT architectures where master data is exposed through services (examples include customer, product and order history services). The development teams then integrate these services into their applications to access vital information for entities than span the enterprise. Such deeply integrated systems have complex and expensive test data and test environment requirements. For example, development teams spend a lot of time understanding the data relationships and pushing the correct data into environments for validating even trivial business processes. It is often easier to virtualize these enterprise services than spending time in buying software licenses and recreating dependent systems in multiple environments.
- Large programs often require changes to complex enterprise services and the development teams need to wait for these changes to be implemented before they can begin working on the main solution. These shared enterprise services are usually managed by a centralized "enterprise services" team. These teams have their own timelines and schedules which sometimes lead to very expensive delays. This problem can be mitigated by utilizing service contract documents to virtualize the services in the development environments.
- Mainframe based services can be the lifeline of modern IT. However setting up mainframe environments for testing is not simple. These expensive systems are often shared between teams which can make their availability and other environmental tasks such as data refreshes difficult. Virtualization technologies that are mature can resolve this problem.

While we have examined a few low hanging virtualization fruits many more exist in the form of cloud development and performance testing.

Subsequently, the teams dive into a deeper set of requirements for these virtualization targets. If the service being built is completely new (or is substantially modified) and has no preexisting data that can be found in production, then the teams will need to determine how data would be sourced and ensure that artificial data can be created effectively. Ensuring coverage for a planned service requires the virtualization leads to examine all the processes that consume the virtual service. Next, all the used cases that belong to identified processes are collated. These are used in the design phase to ensure coverage. If the service that needs to be virtualized already exists in production then the data can be recorded/ sub-setted. For such services, the major concerns are security areas such as masking and ensuring that the subset of data is adequate.

Virtualization teams will then need to incorporate their findings (virtualization targets) into the environment building guide or other artifacts which document how environments will be built or enhanced for the current iteration of the lifecycle.

Analysis and Design Phase

Based on the requirements gathered from the previous phase, the virtualization teams plan and document the process for building the virtual services in the analysis and design phase.

In this phase, virtualization teams focus on services and components identified for virtualization and their associated use cases. For linking the use cases to the services, virtualization teams use a variety of design artifacts that included sequence diagrams and service definitions. This is needed to ensure complete coverage (also means that the virtual service behaves like its real world counterpart). Virtualization products can generate empty virtual services (with no data) which are based on service contracts such as WSDL documents. Populating virtual services with data requires either subsetting and masking (for pre-existing services) or artificial data creation. Although in this phase a virtual service is not populated with data, virtualization teams must ensure that they generate valid input & output pairs for all scenarios (use cases) in a given virtual component. In other words, at the end of the analysis and design phase, a virtualization lead must be able to identify the specific response a virtual service creates at a given request for all possible use cases. Many new virtualization teams ask the development teams for the request/response documents which enable them to quickly jumpstart their virtualization initiatives. However, some targets for virtualization can be very complex and change rapidly. They may require scripting to generate the right behavior and may also need extensive templating to achieve the desired fidelity.

We have mainly focused on the service virtualization for middleware and SOA components, however many products can virtualize databases. The lifecycle methodology is mostly the same for both, database and service virtualization. Leads may prefer to virtualize databases if they believe that the data population and subsetting requirements are easier at the database level than for a larger component. This is more commonly seen when a service derives limited data from a simple target database, derives complex data and composes or orchestrates it across other enterprise services. It is easier in such scenarios to virtualize the simpler and smaller target database than the more complex composite services. This is why you should look into automating the process of virtualizing with "just the test data you need" by capturing these scenarios from pre-production or production databases.

Implementation Phase

The implementation phase involves the creation of virtual services and populating them with data, ensuring accurate performance. The data induced virtual images are then ready for deployment to virtual environments.

Based on the environment and testing/ development requirements, a virtualization team may begin implementing virtual services, even if the larger SOA project is still in the analysis and design phase. As a matter of fact this is needed if the virtual services will be deployed in the development environment for alleviating the development resource constraints.

In this phase virtualization teams populate empty (no data) services with data. Virtual images can derive test data by recording and then masking the production information or through artificial data creation. Steps for capturing production data varies across products but for the most part resemble:

- Configure the recording components to point to the right targets for observing traffic.
- Modify the captured data to create an appropriate subset for usage in virtual images.
- Mask sensitive data.
- Ensure that all scenarios are covered by the test data.

For creating artificial data, virtualization teams often start by collating data into a simple worksheet for stubbing the image. This is then available for import into the virtualization product. There are more advanced ways to automate this import dynamically from live transactions or data sources (for more information see the series of joint papers between Infosys and ITKO and blogs on Test Data Management (TDM) to know more about production level and DevTest environments).

Virtualization teams exit the implementation phase after ensuring that a virtual service can indeed perform for all the identified use cases. The virtual images created as a result of this phase are now ready for use in the next phase.

Deploy and Manage Phase

Virtual images created in the previous phase are now configured and deployed in virtual service containers by the virtualization teams.

Most virtual images require minor corrections as they are used in their target environments. The architects may change the intent of certain parameters or minor service behaviors. Well-designed virtualization tools allow this by incorporating the changes to the configuration or data without requiring the recreation of the images from scratch. Data in a virtual database often needs to be tweaked to meet the rigors of federated relationships. Without this, fine tuning the process models will not work as expected. Given the iterative nature of the SOA lifecycle, virtualization teams also expect to see changes in the virtual images as new behaviors are added with every iteration. Contract versioning can be handled by copying the old image and making changes. Therefore evaluating the reusability factor of a well-designed virtualization tool is indispensible for such scenarios.

In many cases, developers could benefit from virtual services by supporting the development activities right from the start. Then the "deploy and manage" phase may begin as early as the beginning of the implementation phase in the larger SOA lifecycle. Project managers should also expect to expend large efforts in managing image data for performance testing scenarios if there is no robust automation in place (same is the case while preparing data for systems in performance testing environments).

The usage data collected from virtualization tools can provide ROI information. Usage data collated against environment build guides can help managers evaluate the dollar savings and quantify the benefits accrued by service virtualization.

Conclusion

Disciplined and rigorous adherence to a well-defined virtualization lifecycle can alleviate many problems faced by implementation teams when they adopt Service Virtualization. As mentioned in the beginning, a solid adoption methodology/approach to Service Virtualization must be complemented with the right selection of the Service Virtualization tools/products and the implementation expertise of the product. This is the only way organizations can maximize benefits from their Service Virtualization implementations. Further, the use of mature SOA testing processes and good SOA testing tools help increase the effectiveness of the overall Service Virtualization implementations.

About the Author

Gaurish Vijay Hattangadi (Gaurish_Hattangadi@infosys.com) is a Test Solution Consultant with the Independent Validation and Testing Services Practice at Infosys.

References

1. Refer "Driving Better Business Process Scalability with Modern Software Quality"
http://www.infosys.com/IT-services/independent-validation-testing-services/white-papers/Documents/driving-better-business-process-IT.pdf

   and "Service Virtualization for Modern Applications"
http://www.infosys.com/IT-services/independent-validation-testing-services/white-papers/Documents/modern-application-testing.pdf

2. Refer "Driving Better Business Process Scalability with Modern Software Quality"
http://www.infosys.com/IT-services/independent-validation-testing-services/white-papers/Documents/driving-better-business-process-IT.pdf

   and "Service Virtualization for Modern Applications"
http://www.infosys.com/IT-services/independent-validation-testing-services/white-papers/Documents/modern-application-testing.pdf

   *Refer "Key Capabilities of a Service Virtualization Solution"
http://www.infosys.com/IT-services/independent-validation-testing-services/white-papers/Documents/key-capabilities.pdf

Infosys® | Building Tomorrow's Enterprise

For more information, contact askus@infosys.com

About Infosys

Many of the world's most successful organizations rely on Infosys to deliver measurable business value. Infosys provides business consulting, technology, engineering and outsourcing services to help clients in over 30 countries build tomorrow's enterprise.

For more information about Infosys (NASDAQ:INFY), visit www.infosys.com.