

White Paper



Service Virtualization for Modern Applications

Gaurish Vijay Hattangadi, Rajeev Gupta



Abstract

Disruptive innovations such as the internet have repeatedly forced businesses to change quickly or be relegated to history. Every such disruption asks a simple question of service functions in an organization - can they evolve quickly enough? Modern approaches such as service oriented architectures (SOA), along with other similar distributed application architectures, carry great promise and can provide an effective model for lean, agile and componentized IT that can adapt quickly. SOA is one of the most widely adopted enterprise architectural methodologies, but its implementation is anything but simple. Since classical testing methodologies and tools do not fit modern applications well, we have evolved newer more rigorous approaches and products to address increased software complexity and change. Service Virtualization is one of the latest ideas that can provide a faster time to market for software, with higher quality and lesser risk.

Service Virtualization simulates the behavior of software components, to remove dependency constraints on development and testing teams, so that they can deliver software faster, at lower risk and cost. System dependency constraints can severely limit your development and testing efforts, especially in complex, interdependent environments. Downstream components may be unavailable, poor performing, or unusable. In addition, the time and costs to set up test data scenarios and physical test labs, along with frequent contention for shared resources, can significantly limit the scope of testing and compromise application quality. Service Virtualization automatically captures and creates realistic virtual models of dependent systems, so that the real systems are no longer needed for development and testing.

The term "Virtualization" is also widely used in other contexts, such as storage virtualization and memory virtualization. This paper focuses on the use of virtualization as it applies to the problems of developing and testing modern SOA applications.

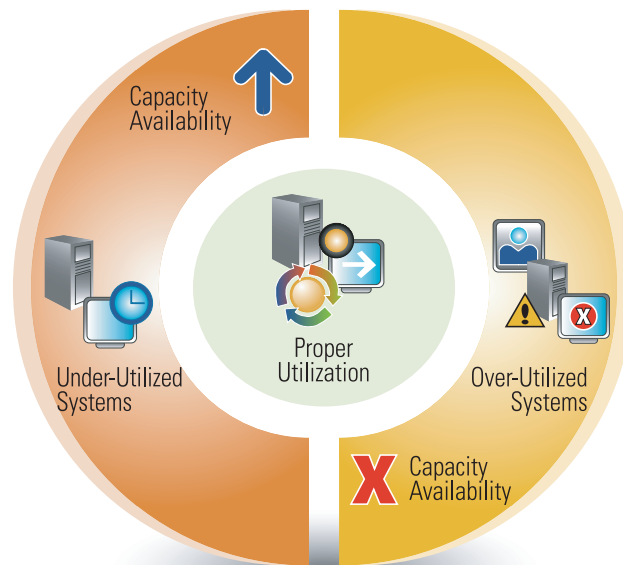
The Birth of Service Virtualization

The earliest exposure to virtualization for most IT practitioners is working with hardware virtualization systems such as Citrix or VMware. The premise of hardware virtualization is simple: underutilized hardware is common to IT infrastructure in most organizations. By pooling together hardware resources and unused capacity, and allocating this pooled infrastructure to application developers and testers, enterprises can drive costs out of the IT infrastructure. This is a great idea that has wide penetration throughout enterprises worldwide. But does it capture all of the value we expect for building today's modern application approaches such as SOA, BPM and Cloud-based environments?

In today's large enterprise, SOA itself is no longer a unique occurrence, in fact, service-orientation is the baseline for having a lean, agile IT process. The question isn't really who does SOA anymore, but who delivers quality services faster and cheaper: or, the most reliable modern implementation, with the fastest time-to-market. Service-oriented development and testing has evolved a great deal, we now have sophisticated new development methodologies, testing methodologies and tools that address the needs of these distributed, heterogeneous, multi-tier applications. While SOA speaks of autonomous systems that are integrated yet free to evolve independently, it fails to decouple the development timelines of teams that must share access to shared systems. Large integration and implementation projects are riddled with the sequencing of tasks that involve many shared system constraints that are resistant to hardware virtualization.

Over-Utilized Systems: When Service Virtualization Matters

The key value espoused by leading virtualization vendors (such as VMware, Microsoft, Citrix, etc.) is making better use of "under-utilized" servers that do not require dedicated system resources. But what are we supposed to do about the kinds of systems that are not good candidates for hardware virtualization? What about critical over-utilized applications that create bottlenecks among teams that need access throughout the software development and testing lifecycle, such as systems of record, mainframes, and other components that teams cannot access? Hardware virtualization has a less compelling value proposition in this scenario. Hardware virtualization focuses on making use of Under-Utilized system capacity, while the



Source: iTKO

Figure 1: Hardware virtualization focuses on making use of Under-Utilized system capacity, while the pressing constraints on software teams are Over-Utilized systems such as mainframes, and fee-based services

pressing constraints on software teams are Over-Utilized systems such as mainframes, and fee-based services.

A strategy to apply in this case is the use of Service Virtualization to reduce multiple team dependencies on over-utilized systems, allowing them to work in parallel. By observing the utilization patterns encountered in enterprise IT environments, IT teams can determine the mix of hardware virtualization and Service Virtualization which should be applied to receive the greatest value.

Ensuring a Responsive Environment

Service-based development approaches provide greater reuse of software components to flexibly meet business needs. However, when employed alone, SOA fails to address the issue of environmental costs expended in testing these integrated systems, due to factors such as the time expended due to complex data setup, time lost due to unavailable dependencies, and potential higher costs of leveraging per-use licensed systems (SaaS or Cloud-based systems). To realize the benefits of modern applications, teams must not only develop new functionality faster, but test this functionality earlier and more often.

Initial attempts to resolve the above problems typically involved teams of developers coding fake "stubs" or "mocks" to attempt to provide some sort of responder for systems under development, but these ultimately turn out to be costly and maintenance intensive and fail to deliver an effective environment for development and testing.

Service virtualization is the capability to emulate the dynamic behavior and performance characteristics of a service in a virtual environment. Service virtualization allows for the replication of behavior for an IT asset which is very valuable when the target system has restricted availability or high usage cost. To testing, development and performance teams, the Virtual Service responds to the system under test just as the live components would - except with the added benefit of 24/7 availability and configurability of data, functionality, and response times.

Service virtualization has evolved today to encompass not just services but also database systems, mainframes and a wide range of other IT assets. By automatically capturing the behavior of a given system by listening to its traffic, a Virtual Service can simulate almost any constrained asset in software development. Service Virtualization has progressed to a point where it can significantly improve the predictability and speed of a project's critical chain in both testing as well as development.

The Methodology of Service Virtualization

The process of service virtualization starts with the identification of a constrained resource. For a given project, the engagement lead must identify a resource that is constrained by cost or availability. The steps for identifying a strong target resource for service virtualization is -

- Identify resources such as database systems that require shared access between multiple teams. An ideal target scenario would be a system that multiple teams need to sequentially develop.
- Identify resources such as SaaS systems that have a high per use cost
- Identify systems that are unavailable for testing or have their own independent development timelines and schedules.
- Identify systems with complex test data management challenges, such as synchronizing test data across multiple distributed systems to conduct the required test scenarios.

Next a virtual model of the target system needs to be constructed. This is mostly done by configuring the virtualization product to examine the inputs and outputs of the target system for accurately modeling behavior. Configuration information can also be obtained by examining metadata provided by real world services and components. For example a common approach to initially generate a virtual service is to examine the WSDL (interface definition of a web service).

The last step is the deployment of the model in a virtual environment. Configuration changes to the virtual model can be made, such as modifying the data or performance characteristics as needed. The virtual model will now behave in the same way as its real world counterpart and can be used in development and testing.

APPLYING SERVICE VIRTUALIZATION

A PRACTICAL EXAMPLE

Service Virtualization provides a versatile range of capabilities to address scenarios where dependent systems are unavailable, inaccessibility, or poor performing.

Imagine a situation where an order fulfillment system needs to be integrated by two teams with varied test data setup and other requirements. Classically you would need to provide sequential access to the system thus approximately doubling development and testing time. With service virtualization both teams can have their own virtual service to develop and test independently of each other. Service virtualization now allows the project manager to conduct concurrent development and testing.

Service virtualization has another closely related use that can be applied to many common IT projects. Let's assume that a project plan calls for a mainframe system to be exposed through services - these services are then to be used by a modern .NET application. A traditional approach would be to conduct the project in two phases - phase one executing the mainframe services development and phase two would be development of the .Net application. With virtualization an architect can model the services to be developed by the mainframe team and create a virtual service for the testing team thus allowing both development teams to work concurrently.

Service virtualization also addresses the costs associated with accessing third-party services. Let's assume that your requirements mandate the integration of a service that cannot be exposed easily for testing. This might be a third party or partner service that carries a per-use cost making testing very expensive. Without virtualization, developers would have been forced to manually create stubs and hope that all would work well in production. With virtualization you can now have a realistic replica that can be hosted in your virtual environment, improving the quality of the application while simultaneously minimizing or eliminating third-party access fees.

These different use cases for virtualization are all born of one necessity - easing resource constraints. If resource optimization and targeting resource bottlenecks is the key to project management then service virtualization is the answer. Parallel development and testing can lead to shorter development and testing time.

S

Summing up

By breaking the constraints around over-utilized systems, service virtualization can enable your organization to obtain a service layer that is more cost effective, flexible, and able to deliver applications faster to market than your competition. Practitioners can achieve effective virtualization for modern applications through a judicious mix of both hardware and service virtualization for tackling problems posed by underutilized as well as over utilized systems. However, the challenge remains in-terms of which service virtualization tool fits requirements perfectly and how do I implement it effectively?

About the Authors

[Gaurish Vijay Hattangadi](mailto:Gaurish_Hattangadi@infosys.com) (Gaurish_Hattangadi@infosys.com)

Gaurish is a Solution Consultant with the Independent Validation and Testing Services Practice at Infosys.

[Rajeev Gupta](mailto:Rajeev.Gupta@itko.com) (Rajeev.Gupta@itko.com)

Rajeev is a Director of Solution Architecture and leads the Value Delivery practice at iTKO.



About iTKO

iTKO helps customers optimize their software development and delivery lifecycle for greater quality and agility in an environment of constant change. iTKO's award winning [LISA™ product suite](#) can dramatically lower quality assurance costs, shorten release cycles, reduce risks, and eliminate critical development and delivery constraints by virtualizing dependent IT resources to provide on-demand accessibility and capacity for teams across the software lifecycle.

For more information, visit <http://www.itko.com> or check out our blog at <http://blog.itko.com>.



About Independent Validation Solutions (IVS)

Infosys' Independent Validation and Testing Solutions practice (www.infosys.com/testing-services) provides third-party validation solutions and services that help global 2000 companies build tomorrow's enterprise.

The 10,200 people strong practice offers end-to-end validation and consulting services which range from custom validation to specialized testing ensuring minimal business disruptions. A strong complementary IP-based solutions program ensures significant improvement in testing efficiency and cycles. Established best practices further help clients achieve operational superiority and improve time-to-market, while retaining a customer-centric focus.

With Infosys' Independent Validation and Testing Solutions, clients are assured of a transparent business partner, world-class processes, speed of execution and the power to stretch their IT budget by leveraging the Global Delivery Model that Infosys pioneered.



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.