

SLIDE ONE - TITLE

First, I want to thank you for attending and for your interest in this subject. Clearly you have your company's success in mind as do I. Your company should be glad they have someone like you keeping their ear to the grind stone. And whether your industry is in the energy, financial, real estate, or the poultry sector this type of architecture will directly affect your bottom line; as well as your team's ability to sleep at night.

SLIDE TWO - AGENDA

The agenda is straight forward, we will stay focused on moving our attention to what is SOA / why it works / why it doesn't work / what is service virtualization and here we will get into some detail, hopefully illuminating how Service Virtualization is beneficial.

SLIDE THREE – GOAL OF THIS TALK

This talk is geared toward companies either knee deep in current SOA projects or companies currently just evaluating SOA as a potential architecture style for one or more of their projects.

Whether you are heavily or lightly involved in the SOA architecture processes, there typically are some misunderstandings about SOA that I would like to address right out of the box. The goals of this presentation are to make you aware of these often overlooked concepts. Some may seem like common sense to most, but they also are easily overlooked when the engineers start pounding the code out the door in order to meet the dead lines set by management.

It's easy for everyone to say, lets create a flexible, agile, and manageable system; but when you ask an engineer "what is the foremost underlying tactic to produce such a system?" they rarely have an answer.

And this is because they are focused on delivering 'the Service' to 'the Business'. And 'that' is how success is measured. How the service gets packaged, deployed, provisioned, discovered, consumed, audited, instrumented, etc... are rarely captured in the business requirements. Even if they are, no two engineers perceive it the same way.

Upon the engineer learning the formal framework of Service Virtualization, they are then armed with a consistent methodology; as we will show in the following slides.

SLIDE FOUR – SOA OVERVIEW

A **service** is an **encapsulated business software component** that is rendered as a pair of separately defined elements —**service interface** and **service implementation**. The interface is the fundamental, definitive component of a service. A service is always intended for programmatic access from another software component (a service consumer). The service interface thus defines the programmatic access "**contract**" of the service. Like any contract, its terms and conditions must be carefully considered, and changes become more difficult once it is in effect. However, the interface provides much latitude for the service implementation to adapt and change over time, so long as it continues to fulfill the terms of the interface contract.

Examples of services include: "Check Customer Credit", "Deny Account", "Open New Account", and "Transfer Funds". All of the listed services provide a unique set of functionality with predictable and repeatable accessible to any client that provides the correct input and expects a given output as defined by the interface contract.

SOA brings forth the exposure of all your services into business level functions your clients can utilize.

SLIDE FIVE & SIX– WHAT IS SOA

FIVE: The underlying principals described in all of these definitions are Service Virtualization. If you overlook this or do not fully understand it, your SOA solution will not meet your business expectations.

As exhibited by each of these definitions, all thought leaders describe SOA through terms such as reusable, abstraction of complexity, separate, discreet, centralized management, distributed implementation and providing business agility.

SIX: Each definition also alludes to the fact that businesses are investing in SOA so they can react to shifting market demands and provide a supportive infrastructure that enables them to make more money

SLIDE SEVEN – WHY IS SOA FAILING YOU

SOA is failing you and me today, because most IT groups are delivering on the promise of SOA through providing Business units with the required services. Simply put, they have tunnel vision. They are so caught up in the sheer responsibility of delivering the right services, in the right granularity, that this consumes their entire focus. Most end up providing what the business requires the day they asked for it, but by the time they deliver, it's typically not adequate because things have changed in the interim.

Because the developer in the middle is focused on doing nothing but creating these services -- nothing else... and in a lot of cases they are only focused on meeting today's requirements, with no forethought of tomorrow... the result is the next generation of hard coded interfaces...

SLIDE EIGHT – WHY IS SOA FAILING YOU

The way in which new customers are going to require items such as security, auditing, authorization, transport protocols, and exposed meta data cannot be accounted for at the time of service development,. We don't know what we don't know. Nor is it possible to:

-- understand how different clients will require different flavors of the same service, possibly requiring some of the services to be rolled up into higher level interactions

-- nor to foresee the requirements to have 100 instances of the same service on the network, each configured slightly differently for each customer; only differing by a configuration value

-- nor to reliably provide the ability to audit or monitor all service interactions as services and consumers evolve over time

All of these things have nothing to do with the core service the engineer is creating. But the business will not succeed without the capabilities to adapt and provide such items.

SLIDE NINE – WHAT IS SERVICE VIRTUALIZATION

I presume everyone on this call is familiar with hardware virtualization. VMware / Virtual Server (hyper-V) / and Xen are all talked about in almost every magazine, book and/or conference today. Everyone in our industry is either using them, or is considering using them. The ability to separate your operating system from specific hardware and literally have it move from machine to machine is very attractive.

Imagine the flexibility you get from hardware virtualization, applied to your business services... this is what Service Virtualization is providing for SOA solutions.

Simply put, Service Virtualization picks up where Hardware Virtualization stops and with the combination of hardware virtualization and service virtualization we are getting one step closer towards the ultimate goal, Utility Computing.

SLIDE TEN – WHAT ARE SERVICE POLICIES

Policies are discrete items: autonomous and reusable – designed and coded outside the scope of your core business service, dynamically providing functionality outside the scope of the business logic required.

Service Policies provide much more of a formal framework simplifying the overall service coding effort. Without service policies, developers try to make their services flexible either through configuration or the dynamic loading of rules... ending up increasing both management and maintenance costs...

Policies permit the introspection of your SOA environment – reporting on what is being done / where / by who / what is doing it / what version / who is using it / etc

This allows the business analyst to drive what policies are adorned onto what service - on a service by service basis. Taking the hassle of the service interactions and configurations away from the developer; who generally did not want to do it to begin with.

SLIDE ELEVEN – EXAMPLE SERVICE POLICIES

Here are some examples of service policies one would expect to apply to a given service depending on how the service is utilized by a given client. Knowing these items are available and extendable allows the service developer to remain focused on their services, allowing them to add such complexities as required...

SLIDE TWELVE – HOW CAN IT IMPACT YOUR BUSINESS

A great number of development teams start with a service - they then run the service on a machine and then end up running a new instance of that service for each customer. This clearly doesn't scale well - isn't a wise use of resources and costs a great deal of time and money to maintain and manage. With service virtualization - one is able to free them from the one to one relationship, free them and do more work with less hardware. The hardware is then treated as a computing commodity and is utilized appropriately. The developers aren't concerned over the protocol, security or hardware in which their service runs - since Operations applies the policies. Thus the people responsible for those aspects are given control of the listed aspects.

As of today, we have numerous customers which were able to take their existing hardware - minimum change in actual service codebase - but still increase the capacity for their services. By simply making certain their solutions keep the service virtualization concepts in the forefront.

SLIDE THIRTEEN – SEPEARATION OF RESPONSIBILITIES

Before Service Virtualization you had to make sure you programmed for extensibility in each service; increasing both upfront costs as well as complexity. Trying to gather 'all' requirements from all the right parties, before the business might even know what they are. Now with Service Virtualization, you are choosing to directly involve the individuals or teams who should be making the decisions when they need to make the decisions; hence aligned with the business. And as your business changes the right people are enabled to dynamically react to those needs.

POLICIES

Dynamic and run time | Security | SLA / QOS | Application Routing | Versioning

CORE SERVICES

Static and design time | Data Binding | Transport Handling | Localized Routing

SLIDE FOURTEEN – LIFE BEFORE SERVICE VIRTUALIZATION

The environment for services normally exists as discrete point solutions exposing specific business functionality for an application that previously lived in isolation. Normally a business will attack each individual isolated application, ending up with many exposed service endpoints, sometimes aggregated together or directly exposed to their respective clients.

Solutions like this make any kind of SLA for the system as a whole unmanageable, because there is no composite application to manage or monitor.

In the move to combine what may have been multiple services, into a single service endpoint, you also create administration and monitoring quagmires. With situations where your clients are calling you asking why the application is not working and you pause, quickly trying to discover the status of the twenty different systems making up your solution. Normally, ending with, “will call you right back...”

And needless to say each application provides its own configuration for

1. Exposing static endpoints
2. Its own security requirements
3. Logging formats
4. Monitoring its own health

How do you keep up with all these configuration files?

How do you manage and monitor all these services and their given dependencies...?

SLIDE FIFTEEN – LIFE WITH SERVICE VIRTUALIZATION (SECURITY)

Now we are moving into the section I’ve named “Life with Service Virtualization”. Here the first examples of applied policies are oriented around Security. When IT decided to expose a given set of core services to their first customer they kept it simple using basic authentication with SSL. The second client, being a more enterprise client, wanted to utilize digital certificates. Well, without service virtualization and policies, you would tell your client to wait three months while you go enable all your solutions. With Service Virtualization, you simply bring up the graphical administration tool for your services; add another endpoint for the service using the required protocol and authentication scheme. If this policy doesn’t exist, you then simply focus on creating the required policies, not changing your core business logic. This provides a much quicker development life cycle, hence equating to less money spent on meeting customer demands.

SLIDE SIXTEEN – LIFE WITH SERVICE VIRTUALIZATION (PROVISIONING)

Service Virtualization at its core is policy driven. These policies don't always just describe the interaction with the services directly. They also focus on how, when, and where the services should be deployed, provisioned, and utilized. These types of policies are commonly known as provisioning policies.

These types of policies allow system operations the ability to define:

- What services should be deployed
- How each service should be exposed (protocol policies)
- What resources a service is allowed to consume (resource policies)
- When should a service be accessible (scheduling policies)
- Configure services for specific users with specific needs

This allows your IT department the ability to condense their hardware into a common commodity pool. Allowing resources to be allocated to the services which require it at the moment they need it, versus, requiring your IT department to statically allocate virtual / real hardware resources. [Hence, reducing overall management costs]

SLIDE SEVENTEEN – LIFE WITH SERVICE VIRTUALIZATION (DISCOVERY)

Another critical aspect to Service Virtualization is dynamic discovery of services. With the advent of enabling system operations to provision services dynamically, having the ability discover, consume, and execute services becomes very important. Each service as it comes online will register itself in the global registry, allowing its requests to route appropriately upon demand. This allows your services to exist anywhere on your network without having to completely reconfigure each client as requirements change.

As well, this architecture supports the ability to publish these dynamic endpoints into existing service registries, allowing traditional clients the ability to utilize these dynamic services as easily as they use traditional services.

SLIDE EIGHTEEN – LIFE WITH SERVICE VIRTUALIZATION (OPERATIONAL)

With Service Virtualization also comes Operational Readiness. This brings awareness to:

- Each message coming into and out of the system (auditing, utilization, status)
- Each service running (utilization, dependencies)
- Each service endpoint (protocol, security, status)
- Each server's status (utilization, health, etc)

So when an error does occur, you immediately know where, when, and by who; as well as you don't have to wait for a customer to call you to let you know it occurred. You have the capability to deal with operational events using the same orchestration and service platform as you used to build your core services. Hence, making operational readiness as important as the services themselves; true SLA support.

SLIDE NINETEEN – LIFE WITH SERVICE VIRTUALIZATION (VERSIONING)

Let's say you design, implement and deploy a new service for customer A. Everything works great, until Customer B needs the same service, just a bit different. In most systems this requires a great deal of work / hard coding the new endpoints with the new services, normally on new hardware; or at least a new VM.

Service Virtualization provides the capability for systems to either route or transform/translate older versions versus having to upgrade all the systems / customers. Allowing you to provide services to either run side-by-side, or transform the older messages into the new formats.

SLIDE TWENTY – LIFE WITH SERVICE VIRTUALIZATION (COMPOSITION)

Now when just versioning your services becomes common place, some solutions might require the ability to provide simplicity to your end clients. I know exposing twenty systems to your end clients might bring joy to your faces, letting them just figure out how to utilize your systems. But some clients might demand simplicity and thereby require you to provide custom orchestrations of your services for each client. This will mean having the capability to expose the same service to multiple clients with different orchestrations for each. With service virtualization this should be as easy as routing each client's request to the right orchestration. And providing the ability to adapt as new clients come online.

This also provides the ability to update your backend at will without your clients having to change.

SLIDE TWENTY ONE – LIFE WITH SERVICE VIRTUALIZATION (LOAD BALANCING)

Last but not least is load balancing! If you are going to provide the ability to dynamically provision, discover, orchestration, version, and the like... Then you have to provide a way to utilize all those services on your network. Whether you tend to use hardware load balancers or your solutions built in algorithms, properly utilizing the allocated hardware is indeed important. And providing all these metrics to the dynamic monitoring allows the system to know when it's appropriate to allocate either more or less hardware based on true utilization, versus just allowing system administrators or developers to guess upon deployment. As well as providing the ability to send notifications to your IT staff when your systems are approaching peak capacity, so they can go purchase more hardware to accept all the new orders...

And with Service Virtualization, choosing which methodology to load balance should be as easy as just selecting the right policy.

SLIDE TWENTY TWO – SUMMARY

So in closing, Service Virtualization is really focused on enabling SOA solutions by separating the responsibilities into business logic - and everything else. Allowing you to expose your services to the world, presenting whatever perspective your clients require in order to properly utilize the services they requested. As well as providing the infrastructure needed reducing the load from the development staff, and providing a dynamic environment for the system administrators allowing them to constantly adjust the solution to stay in line with the business's demands.

SLIDE TWENTY FOUR – QUESTIONS?

Robert E. Brooks

Chief Technologist

Robert.Brooks@gridGISTICS.net