Service Virtualization Maturity Model

	Ad-Hoc	Reactive	Proactive	Managed	Optimized
Focus	Individual: One-off attempts to bridge gaps obstructing an individual's ability to complete a specific development or test task.	Project: Service Virtualization (SV) emulates dependent system components and allows the project's development or testing tasks to "shift left."	Environment: SV provides consistent access to dev/test environments that involve difficult-to-access, inconsistent, or unreliable system dependencies.	Scenario: Environments are coordinated to rapidly exercise different scenarios (performance, security, error conditions, etc.) in order to achieve better testing outcomes.	Enterprise: Provides optimized and secured environment access across and beyond the enterprise—including portals for business partners.
Characteristics	Dev/test scenarios need to execute across complex, dependency-rich environments, but access to a staged test environment is constrained. Developers react by creating stubs to pry the test or scenario out of the constrained environment. This is an 'inside-out' approach. QA/performance test engineers react by waiting for access to a complex staged test environment (if available) or using stubs to bypass critical dependent systems.	A single group/project drives the creation and management of virtual assets that mimic behavior of incomplete or unavailable dependent components. Virtual assets are created for specific use cases and are augmented when needed for alternative cases. The extension of data sets or performance profiles is reactive based on specific testing needs.	A more holistic approach; accommodates a broader enterprise audience. SV is leveraged to provide continuous access to realistic dev/test environments (rather than simply alleviate project-specific access pains). Virtual assets are created, accessed, and managed in the context of environments. Policies, procedures, and standards exist around the application of SV. Consistent, continuous environment access enables more extensive and accurate testing to occur with or without access to a staged test environment.	Environments are governed by business rules that not only dictate what components are available, but also specify what permutations are valid under various contexts. Since business rules automate environment access and control, users can rapidly "self-provision" test environments. Configurations are accessed as 'disposable software' with zero risk. Lays the foundation for goal-oriented business-driven scenarios.	Provides the appropriate level of environment access to each constituency. A Center of Excellence is established to optimize and manage policies, procedures, and standards. Optimized environment for goal-oriented, business-driven scenarios significantly reduces application risk.

Service Virtualization Maturity Model

	Ad-Hoc	Reactive	Proactive	Managed	Optimized
Process Fit	Any pockets of maturity are based on the experience and initiative of individuals. No centralization of assets; every man for himself.	Enables earlier, easier testing, but does not necessarily diminish the need for staged test environments. A net new test environment is available by the use of SV; this is an initial step for facilitating Agile/parallel development.	Creates more sophisticated and flexible dev/test environments. Promotes a level of interconnectedness between SV and virtual test lab management systems.	Facilitates more mature coordination between SV and virtual test lab management systems.	Seamless integration and orchestration of SV with virtual test lab management systems. The unified solution establishes a single entity that allows for regression test suites to automatically call complex environments.
Environment Management	Assets are typically created as one-off solutions and stored on a local machine, inaccessible to anyone but the creator. The 'stub' is created without consideration of the environment and serves only the individual test.	Virtual assets might be evolved if needed to bridge project-specific gaps, but no overarching change management policies or processes exist.	Change is managed from the environment perspective. Users are notified of new virtual asset versions upon accessing the environment; change-impacts are highlighted, and users have the option of accessing the required version.	Robust change management and scenarios. Automated business rules drive the evolution of changing environment components.	The SV environment is governed by differentiated states associated with how various entities are accessing SV assets and environments.
Maturity Drivers	The application is not adequately tested or integrated due to limited access to environment conditions. The time and complexity of managing stubs outweighs the value they provide. Constrained access to staged test environments results in unacceptable test coverage or time-to-market delays.	Additional groups request access to the virtual assets with varied configuration options. Increasingly comprehensive scenarios vs. multiple dependent systems need to be tested. Project experience exposes the need for common, proactive processes for reuse and management.	Increased need to access multiple on-demand, "disposable" test environment configurations tailored for specific application or project demands.	Controlled access to sophisticated test environments is needed internally across the enterprise and externally with strategic business partners.	