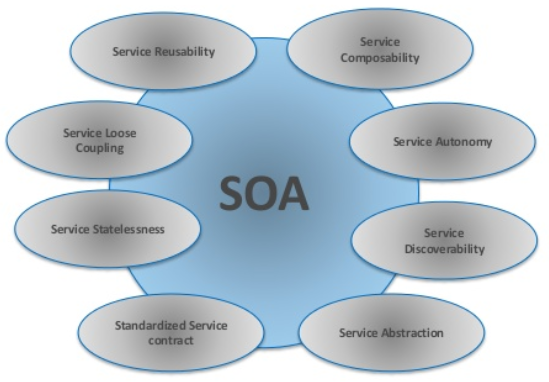
<https://www.slideshare.net/ExigenServices/service-design-principles-and-patterns>



Standardized Service Contract

* We want reusable service to be agnostic. If this logic is sufficiently generic and common, the service can be classified as reusable.
* A strong naming convention, for example, can immediately convey the purpose and requirements of individual capabilities belonging to a service (discoverability)
* Individual services need to be capable of acting as effective composition members in order for a service inventory to facilitate repeated compositions in support of fulfilling a wide range of automation requirements.
* Overly coarse-grained capabilities are often less suitable for when the service needs to participate in a larger composition

Service Coupling

* A service should not be designed to support a particular (usually pre-existing) service consumer program.

Service Abstraction

* hide private function
* Programmatic Logic Abstraction : hide algorithms, exception handling and logging routines, and other logic associated with how the program is constructed

Service Reusability

* The service logic can be accessed concurrently
* Current business requirements and the definition of common usage scenarios
* The application of the Service Reusability principle often leads to the necessity of defining coarse and fine-grained versions of similar capabilities
* The agnostic focus of entity and utility services is clearly intended to provide a functional context suitable for the encapsulation of reusable logic.
* The greater the reuse potential, the greater the opportunities for the service to be repeatedly composed

Service Autonomy

* A distributable deployment environment, so as to allow the service to be moved, isolated, or composed as required.

Service Discoverability

* If service does not have the necessary capabilities but still provides a suitable functional context, it can be identified as the location to which add the required functionality (as an extension to the service)

Service Statelessness

* Reduce state into