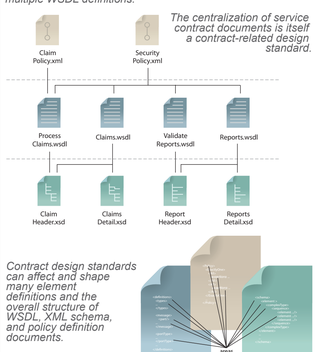
Standardized Service Contract

"Services within the same service inventory are in compliance with the same contract design standards."

When a service is implemented as a Web service, the service contract can be comprised of a WSDL definition and multiple XML schema and policy definitions, as well as supplementary documents, such as an SLA.

This principle preaches a "contract first" approach to service delivery, whereby contracts are custom-developed (prior to the development of the service logic) according to design standards that apply to all services within a given service inventory.

Standardized policies and schemas can be centralized so that one definition represents an "official" set of policy assertions or complex types that can be referenced by multiple WSDL definitions.



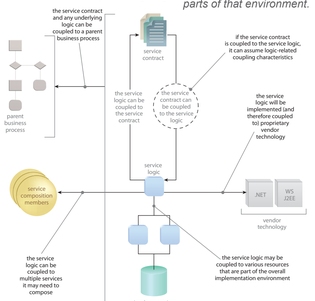
The centralization of service contract documents is itself a contract-related design standard.

Contract design standards can affect and shape many element definitions and the overall structure of WSDL, XML schema, and policy definition documents.

Service Loose Coupling

"Service contracts impose low consumer coupling requirements and are themselves decoupled from their surrounding environment."

A service contract that is derived from its underlying environment can end up forming negative types of coupling upon parts of that environment.



Logic-to-contract coupling is considered a positive form of coupling because it represents the independent creation of a contract that is decoupled from the service environment.

Service consumer programs are required to couple themselves to a service's contract. As a result, they inherit whatever forms of negative or positive coupling that reside within the service contract.

This can lead to various forms of "indirect" or unintentional coupling.

Proliferation of negative coupling is undesirable because it leads to a fragile and inflexible service inventory reminiscent of past integration architectures.

This principle relates to the Contract Centralization pattern which dictates that the service contract be the sole means of accessing service logic and resources.

Service consumer designers may not be aware of the fact that the contract their program is forming a dependency on is negatively coupled.

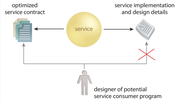
Proliferation of positive coupling is desirable so as to allow service implementations to evolve without impacting service consumers.

Service Abstraction

"Service contracts only contain essential information and information about services is limited to what is published in service contracts."

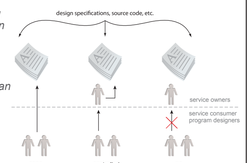
When determining what information about a service should be abstracted, it is helpful to categorize service meta data into distinct categories.

The application of this principle can affect the abstraction of each of these meta information types differently.



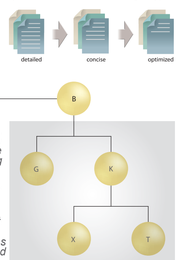
This principle advocates the deliberate hiding of service meta data so that a minimal amount of information about a service is accessible to the outside world.

Access control procedures can therefore become a requirement that may need to be addressed on an organizational level via the introduction of new or modified processes.



The application of this principle can effectively turn a service into a "black box" where the only information made available about the service is what is published in its contract (which may encompass what is also published in a service registry).

Therefore, the content of the service contract itself is a primary focal point for which different abstraction levels exist.



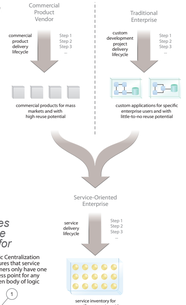
As a result of this principle, service consumer program designers may be unaware that a service is composing others.

This places a great deal of emphasis on the reliability and the predictability of a service, regardless of what it may be encapsulting (which also raises issues as to what should be published within service SLAs).

Service Reusability

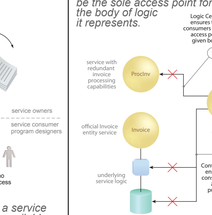
"Services contain and express agnostic logic and can be positioned as reusable enterprise resources."

Within service-orientation reusability represents a core target design characteristic that is tied to the goal of achieving repeated ROI for agnostic services.



This principle combines techniques from traditional commercial product design with traditional enterprise project delivery.

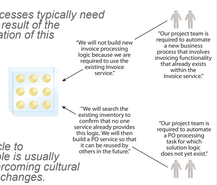
Positioning services as reusable enterprise resources relates to the Logic Centralization design pattern that dictates that each reusable service be the sole access point for the body of logic it represents.



When combined, Logic and Contract Centralization result in a highly standardized and normalized service inventory in full support of maximizing reusability potential and loose consumer coupling.

Project delivery processes typically need to be changed as a result of the consistent incorporation of this principle so as to ensure that Logic Centralization is always respected and that reuse potential of agnostic services is maximized.

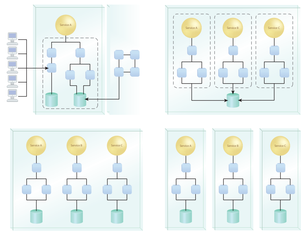
The greatest obstacle to realizing this principle is usually associated with overcoming cultural resistance to these changes.



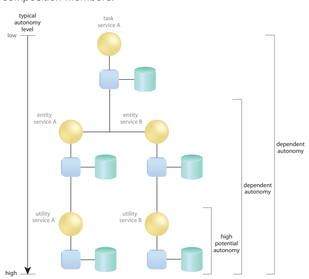
Service Autonomy

"Services exercise a high level of control over their underlying runtime execution environment."

The more control a service has over its underlying runtime implementation, the more predictable its runtime behavior will be. Reducing shared access to service resources and increasing physical isolation can raise a service's ability to function autonomously.



The autonomy of individual services is especially important to the effectiveness of service compositions. Because a service composing another automatically loses autonomy, the level of autonomy a composition controller can attain is often limited to the collective autonomy levels of its composition members.



SOA: Principles of Service Design

by Thomas Erl

Copyright © 2008 SOA Systems Inc.
ISBN: 0132344823, Prentice Hall
(purchase this poster at www.soaposters.com)

www.soabooks.com
www.prenhall.com
www.soasystems.com
www.soaschool.com
www.soamag.com

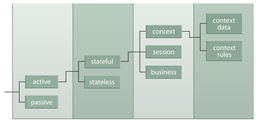
www.whatissoa.com
www.soaprinciples.com
www.soapatterns.com
www.soaspecs.com
www.soaglossary.com

Prentice Hall Service-Oriented Computing Series from Thomas Erl

Service Statelessness

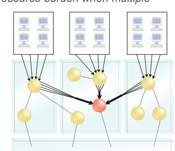
"Services minimize resource consumption by deferring the management of state information when necessary."

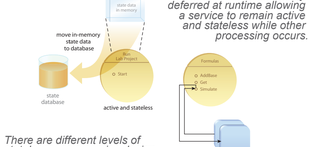
Depending on the nature of its logic and its role within a composition, a service may need to transition through different states and may need to manage different types and amounts of state data.



State data management consumes system resources and can result in a significant resource burden when multiple instances of services are concurrently invoked, especially with agnostic services that are involved in the automation of multiple business processes.

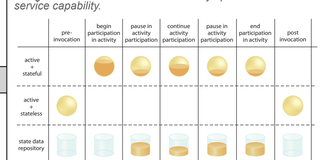
Therefore, the temporary delegation and deferral of state management can increase service scalability and support a wider range of reuse and recomposition over time.





State data is commonly deferred at runtime allowing a service to remain active and stateless while other processing occurs.

There are different levels of statelessness a service design can achieve, depending on the frequency of state deferral and the quantity of state data being deferred. These levels are usually specific to each service capability.

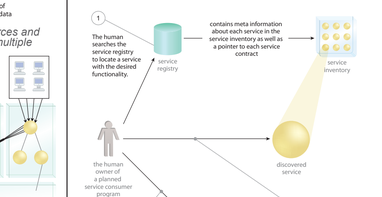


Service Discoverability

"Services are supplemented with communicative meta data by which they can be effectively discovered and interpreted."

Of the four service meta information types functional and quality of service data are most relevant when focusing on a service's communications quality for discoverability and interpretability purposes.

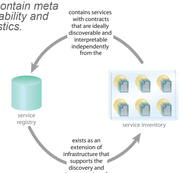
The application of this principle supports a standardized process of service discovery and inerpretation within an organization through the use of a service registry as the central repository of service meta data.



A key goal of this principle is to enable a wide range of project team members to effectively carry out the discovery process and not to limit it to those with technical expertise.

Both service contracts and records within a service registry contain meta information with discoverability and interpretability characteristics.

Much of this information relates to and originates from the service profile document that may have been created and maintained since the service was first conceptualized during the service modeling phase (see Chapter 15 and Appendix B).

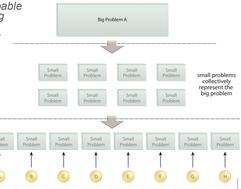


Service Composability

"Services are effective composition participants, regardless of the size and complexity of the composition."

Service-orientation is a design paradigm with a distinct approach to carrying out a separation of concerns.

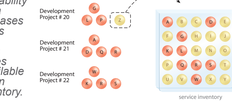
Services capable of addressing agnostic or cross-cutting concerns can be repurposed to solve multiple problems.



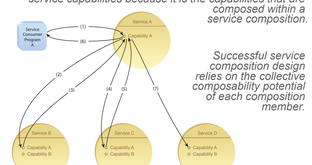
This requires an effective means of decomposing solution logic and repeatedly recomposing it to solve new problems.

This principle is primarily concerned with a service's ability to act as an effective composition member so that it can support the realization of new business requirements that can be fulfilled by the assembly of service compositions.

The composability potential of a service increases and becomes increasingly important as more services become available within a given service inventory.



A key aspect of this principle and service compositions in general is that individual concerns are, in fact, solved by service capabilities because it is the capabilities that are composed within a service composition.



Successful service composition design relies on the creative composability potential of each composition member.