8.4. How service-orientation principles inter-relate

When reading through the previous descriptions, a number of questions might come to mind, such as:

- What's the difference between reusability and composability? (Aren't you reusing a service when you compose it?)
- What's the difference between autonomy and statelessness? (Aren't both a representation of service independence?)
- What's the difference between loose coupling and the use of a service contract? (Doesn't a service contract automatically implement loose coupling?)

To answer these and other questions, this section revisits our service-orientation principles to explore how each relates to, supports, or is affected by others. To accomplish this, we abbreviate the original names we assigned each principle, as follows:

- Services are reusable = service reusability
- Services share a formal contract = service contract
- Services are loosely coupled = service loose coupling
- Services abstract underlying logic = service abstraction
- Services are composable = service composability
- Services are autonomous = service autonomy
- Services are stateless = service statelessness
- Services are discoverable = service discoverability

We intentionally prefix each principle with the word "service" to emphasize that the principle applies to the design of a service only, as opposed to our SOA characteristics, which apply to the design of SOA as a whole.

Note

Each relationship is essentially described twice within these sections. This repetitiveness is intentional, as this part of the chapter is provided more for reference purposes. Feel free to skip ahead if you are not interested in learning about each individual principle-to-principle relationship at this point.

8.4.1. Service reusability

When a service encapsulates logic that is useful to more than one service requestor, it can be considered reusable. The concept of reuse is supported by a number of complementary service principles, as follows.

- Service autonomy establishes an execution environment that facilitates reuse because the service has independence and self-governance. The less dependencies a service has, the broader the applicability of its reusable functionality.

- Service statelessness supports reuse because it maximizes the availability of a service and typically promotes a generic service design that defers activity-specific processing outside of service logic boundaries.
- Service abstraction fosters reuse because it establishes the black box concept, where processing details are completely hidden from requestors. This allows a service to simply express a generic public interface.
- Service discoverability promotes reuse, as it allows requestors (and those that build requestors) to search for and discover reusable services.
- Service loose coupling establishes an inherent independence that frees a service from immediate ties to others. This makes it a great deal easier to realize reuse.

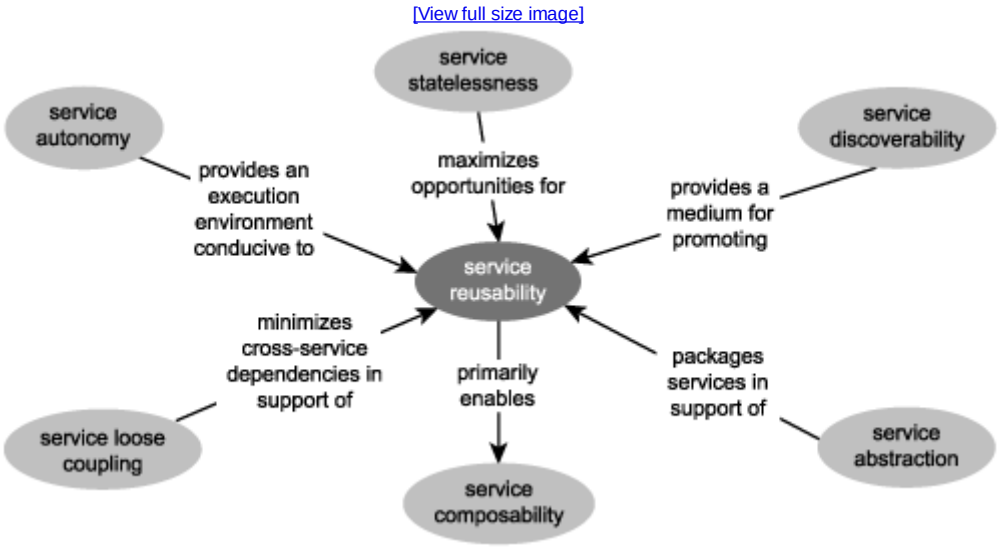
Additionally, the principle of service reuse itself enables the following related principle:

- Service composability is primarily possible because of reuse. The ability for one service to compose an activity around the utilization of a collection of services is feasible when those services being composed are built for reuse. (It is technically possible to build a service so that its sole purpose is to be composed by another, but reuse is generally emphasized.)

Figure 8.26 provides a diagram showing how the principle of service reusability relates to others.

Figure 8.26. Service reusability and its relationship with other service-orientation principles.

[View full size image]



8.4.2. Service contract

A service contract is a representation of a service's collective metadata. It standardizes the expression of rules and conditions that need to be fulfilled by any requestor wanting to interact with the service.

Service contracts represent a cornerstone principle in service-orientation and therefore support other principles in various ways, as follows:

- Service abstraction is realized through a service contract, as it is the metadata expressed in the contract that defines the only information made available to service requestors. All additional design, processing, and implementation details are hidden behind this contract.
- Service loose coupling is made possible through the use of service contracts. Processing logic from

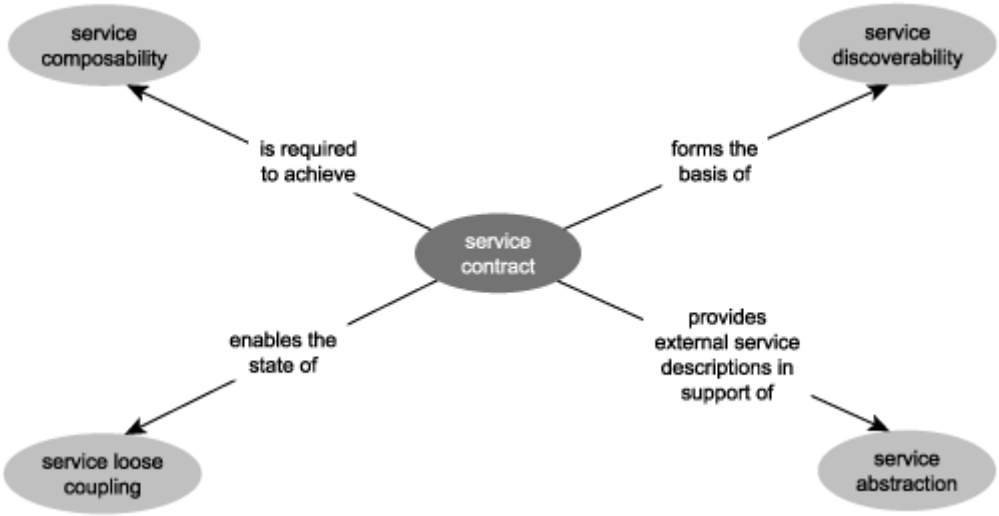
different services do not need to form tight dependencies; they simply need an awareness of each other's communication requirements, as expressed by the service description documents that comprise the service contract.

- Service composability is indirectly enabled through the use of service contracts. It is via the contract that a controller service enlists and uses services that act as composition members.
- Service discoverability is based on the use of service contracts. While some registries provide information supplemental to that expressed through the contract, it is the service description documents that are primarily searched for in the service discovery process.

The diagram in Figure 8.27 illustrates how the principle of service contract usage relates to others.

Figure 8.27. The service contract and its relationship with other service-orientation principles.

[View full size image]



8.4.3. Service loose coupling

Loose coupling is a state that supports a level of independence between services (or between service providers and requestors). As you may have already noticed, independence or non-dependency is a fundamental aspect of services and SOA as a whole. Therefore, the principle of persisting loose coupling across services supports the following other service-orientation principles:

- Service reusability is supported through loose coupling because services are freed from tight dependencies on others. This increases their availability for reuse opportunities.
- Service composability is fostered by the loose coupling of services, especially when services are dynamically composed.
- Service statelessness is directly supported through the loosely coupled communications framework established by this principle.
- Service autonomy is made possible through this principle, as it is the nature of loose coupling that minimizes cross-service dependencies.

Additionally, service loose coupling is directly implemented through the application of a related service-orientation principle:

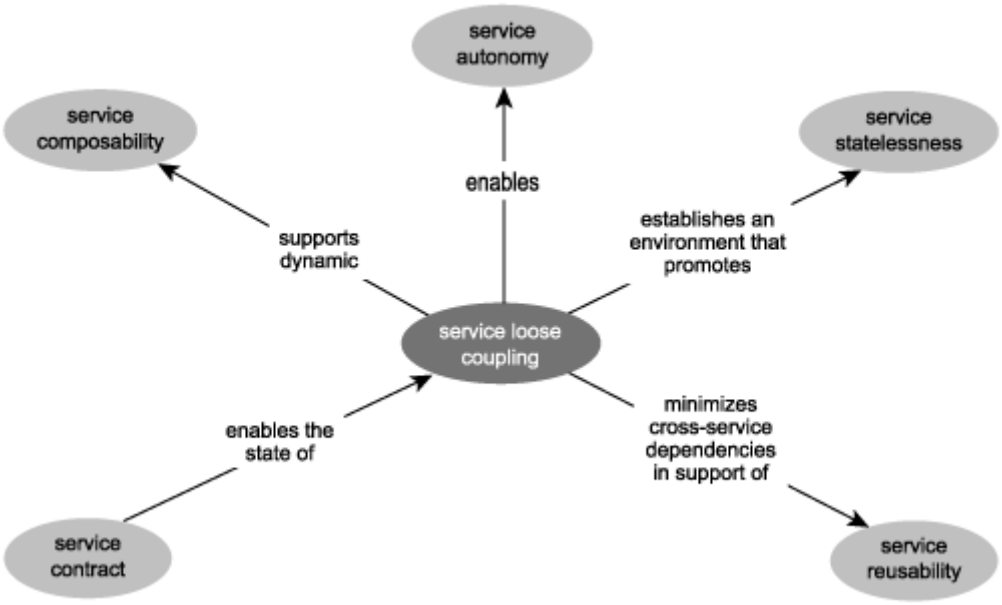
- Service contracts are what enable loose coupling between services, as the contract is the only piece of

information required for services to interact.

Figure 8.28 demonstrates these relationships.

Figure 8.28. Service loose coupling and its relationship with other service-orientation principles.

[View full size image]



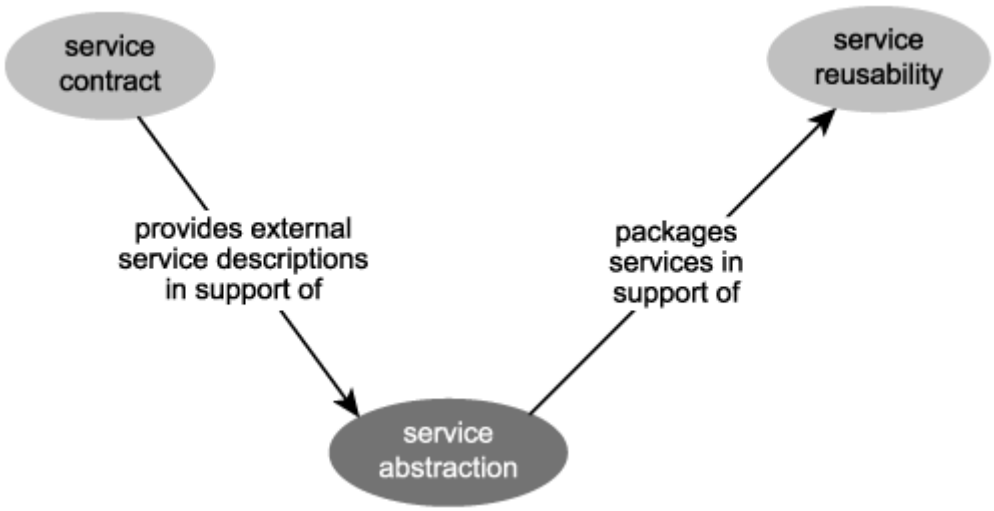
8.4.4. Service abstraction

Part of building solutions with independent services is allowing those services to encapsulate potentially complex processing logic and exposing that logic through a generic and descriptive interface. This is the primary benefit of service abstraction, a principle that is related to others, as explained here:

- Service contracts, in a manner, implement service abstraction by providing the official description information that is made public to external service requestors.
- Service reusability is supported by abstraction, as long as what is being abstracted is actually reusable.

These relationships are shown in Figure 8.29.

Figure 8.29. Service abstraction and its relationship with other service-orientation principles.



8.4.5. Service composability

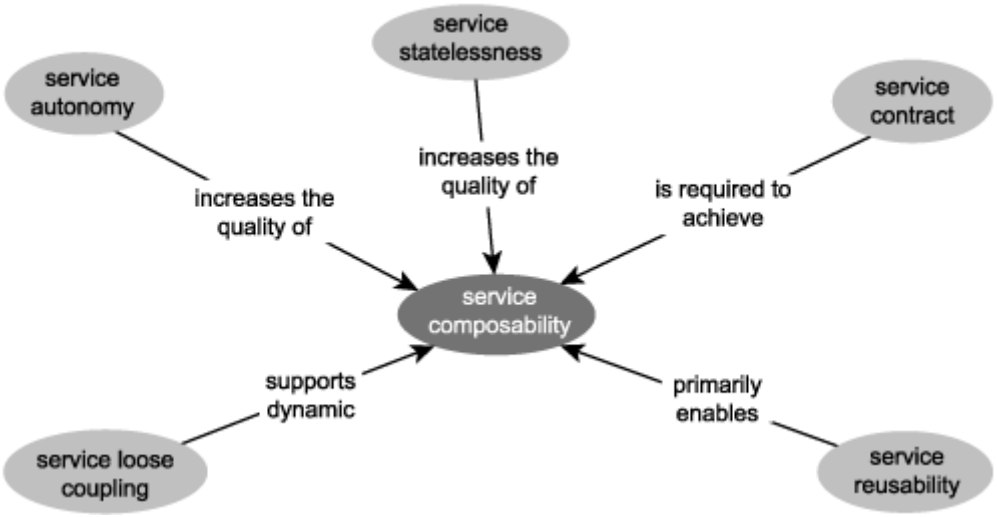
Designing services so that they support composition by others is fundamental to building service-oriented solutions. Service composability therefore is tied to service-orientation principles that support the concept of service composition, as follows:

- Service reusability is what enables one service to be composed by numerous others. It is expected that reusable services can be incorporated within different compositions or reused independently by other service requestors.
- Service loose coupling establishes a communications framework that supports the concept of dynamic service composition. Because services are freed from many dependencies, they are more available to be reused via composition.
- Service statelessness supports service composability, especially in larger compositions. A service composition is reliant on the design quality and commonality of its collective parts. If all services are stateless (by, for example, deferring activity-specific logic to messages), the overall composition executes more harmoniously.
- Service autonomy held by composition members strengthens the overall composition, but the autonomy of the controller service itself actually is decreased due to the dependencies on its composition members.
- Service contracts enable service composition by formalizing the runtime agreement between composition members.

Figure 8.30 further illustrates these relationships.

Figure 8.30. Service composability and its relationship with other service-orientation principles.

[View full size image]



8.4.6. Service autonomy

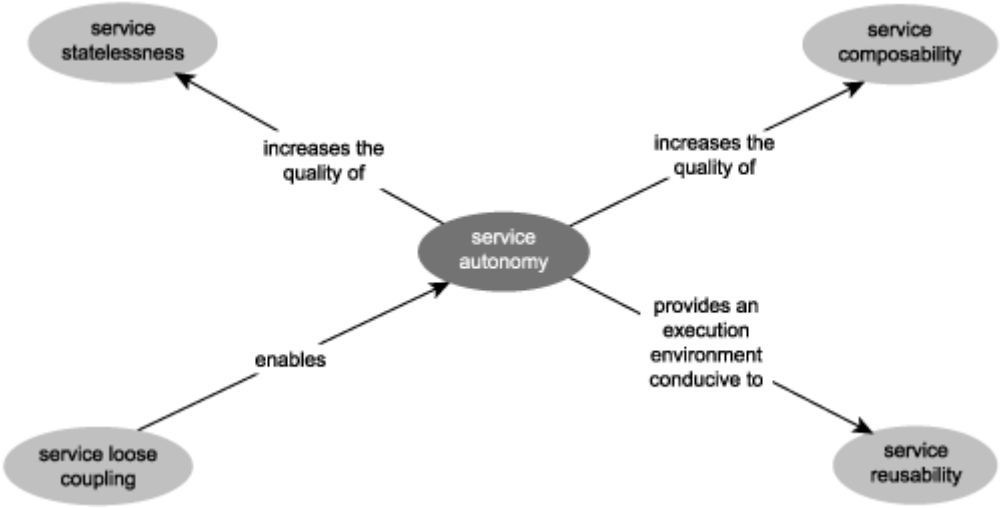
This principle applies to a service's underlying logic. By providing an execution environment over which a service has complete control, service autonomy relates to several other principles, as explained here:

- Service reusability is more easily achieved when the service offering reusable logic has self-governance over its own logic. Service Level Agreement (SLA) type requirements that come to the forefront for utility services with multiple requestors, such as availability and scalability, are fulfilled more easily by an autonomous service.
- Service composability is also supported by service autonomy—for much of the same reasons autonomy supports service reusability. A service composition consisting of autonomous services is much more robust and collectively independent.
- Service statelessness is best implemented by a service that can execute independently. Autonomy indirectly supports service statelessness. (However, it is very easy to create a stateful service that is also fully autonomous.)
- Service autonomy is a quality that is realized by leveraging the loosely coupled relationship between services. Therefore service loose coupling is a primary enabler of this principle.

The diagram in Figure 8.31 shows how service autonomy relates to these other principles.

Figure 8.31. Service autonomy and its relationship with other service-orientation principles.

[View full size image]



8.4.7. Service statelessness

To successfully design services not to manage state requires the availability of resources surrounding the service to which state management responsibilities can be delegated. However, the principle of statelessness is also indirectly supported by the following service-orientation principles:

- Service autonomy provides the ability for a service to control its own execution environment. By removing or reducing dependencies it becomes easier to build statelessness into services, primarily because the service logic can be fully customized to defer state management outside of the service logic boundary.
- Service loose coupling and the overall concept of loose coupling establishes a communication paradigm that is fully realized through messaging. This, in turn, supports service statelessness, as state information can be carried and persisted by the messages that pass through the services.

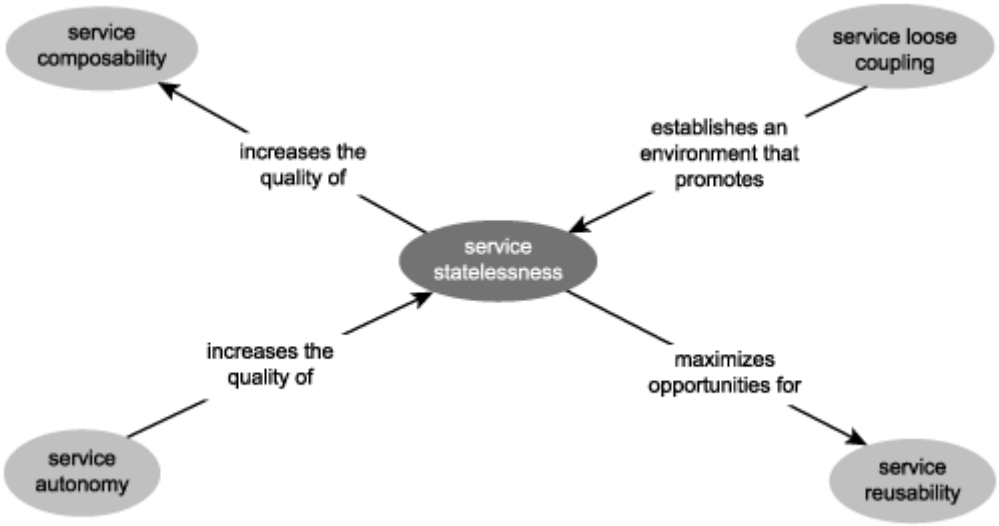
Service statelessness further supports the following principles:

- Service composability benefits from stateless composition members, as they reduce dependencies and minimize the overhead of the composition as a whole.
- Service reuse becomes more of a reality for stateless services, as availability of the service to multiple requestors is increased and the absence of activity-specific logic promotes a generic service design.

Figure 8.32 illustrates how service statelessness relates to the other service-orientation principles.

Figure 8.32. Service statelessness and its relationship with other service-orientation principles.

[View full size image]



8.4.8. Service discoverability

Designing services so that they are naturally discoverable enables an environment whereby service logic becomes accessible to new potential service requestors. This is why service discoverability is tied closely to the following service-orientation principles:

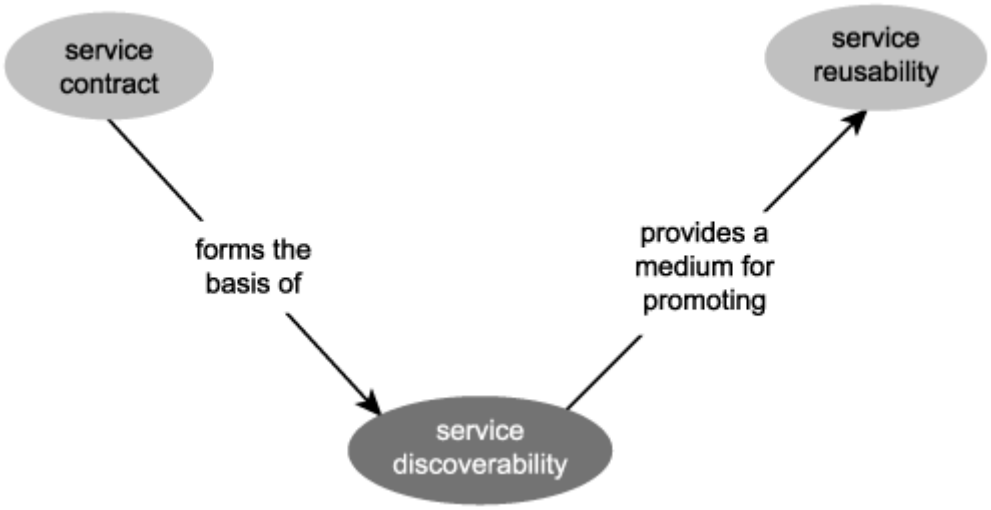
- Service contracts are what service requestors (or those that create them) actually discover and subsequently assess for suitability. Therefore, the extent of a service's discoverability can typically be

associated with the quality or descriptiveness of its service contract.

- Service reusability is what requestors are looking for when searching for services and it is what makes a service potentially useful once it has been discovered. A service that isn't reusable would likely never need to be discovered because it would probably have been built for a specific service requestor in the first place.

The diagram in Figure 8.33 shows how service discoverability fits in with the other service-orientation principles.

Figure 8.33. Service discoverability and its relationship with other service-orientation principles.



SUMMARY OF KEY POINTS

- Service-orientation principles are not realized in isolation; principles relate to and support other principles in different ways.
- Principles, such as service reusability and service composability, benefit from the support of other implemented principles.
- Principles, such as service loose coupling, service contract, and service autonomy, provide significant support for the realization of other principles.