SOA Principles and Design

The common semantics should (Enterprise ID's):

- Identify information that must be shared across the enterprise and between services
- Define the meaning and context of that information
- Identify techniques for mapping enterprise semantics to existing application data models

CHALLENGE	ARCHITECTURAL REQUIREMENT
Reuse	Ability to publish, search for, evaluate, and register as a consumer of a service. Capabilities for managing and maintaining a service life cycle across organizational boundaries. Ability to guarantee availability and lifetime of a service version. Mechanisms for decoupling the consumer's life cycle from the provider's.
Efficient Development	Have a reference architecture that guides the development of services. Use BPM to define business processes, based on service composition and a layered set of services. Have efficient processes that manage the integrity of the total set of services for both providers and consumers in accordance with the overall vision and business and information models.
Integration of Applications and Data	Have an enterprise, common semantic model for the shared information. Have a reference architecture that differentiates between business services and integration services. Have a reference architecture that describes common patterns for integration. Have infrastructure capabilities that enable semantic transformation between existing systems and the enterprise model.
Agility, Flexibility, and Alignment	Have a reference architecture that defines the business and information aspects of SOA and their relationship to the enterprise. Have an enterprise, common semantic model that is used to inform the service interface design. Use model-based development techniques to ensure

the traceability between the business models and the implemented systems. Have processes that enable and validate conformance.

SOA needs to describe the following aspects of services within an enterprise

1. A definition of services, the granularity, and types of services
2. How services are constructed and used
3. How existing packaged and legacy systems are integrated into the service environment
4. How services are combined into processes
5. How services communicate at a technical level (i.e., how they connect to each other and pass information)
6. How services interoperate at a semantic level (i.e., how they share common meanings for that information)
7. How services align with the businesses strategy and goals
8. How to use the architecture

Modularity and granularity: Coarse-grained services provide a greater level of functionality within a single service operation. This helps to reduce complexity and network overhead by reducing the steps necessary to fulfill a given business activity.

Fine-grained service operations: provide the exchange of small amounts of information to complete a specific discrete task.

Encapsulation

— Services exhibit a strict separation of the service interface (*what* a service does) from the service implementation

Loose coupling

— Coupling describes the number of dependencies between a service consumer and provider. Loosely coupled services have few, well-known and -managed dependencies.

Isolation of responsibilities — Services are responsible for discrete tasks

or the management of specific resources. A key characteristic of service design is the isolation of responsibility for specific functions or information into a single service.

Autonomy— Autonomy is the characteristic that allows services to be deployed, modified, and maintained independently from each other and the solutions that use them. An autonomous service's life cycle is independent of other services.

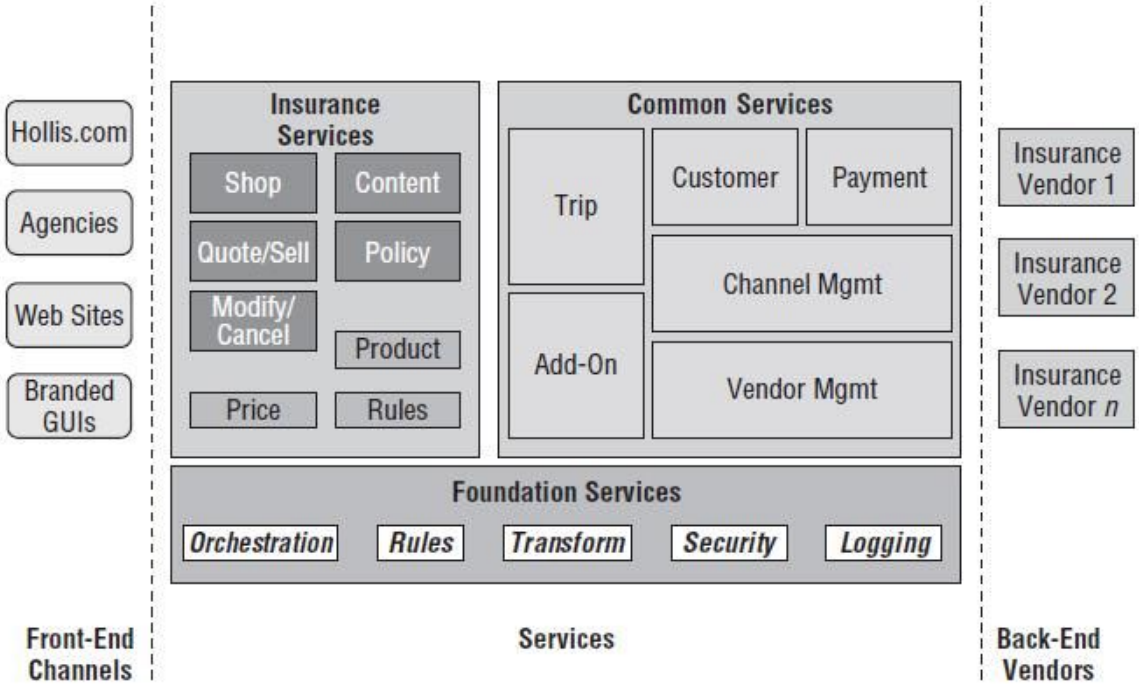

Figure 13-2 Conceptual architecture