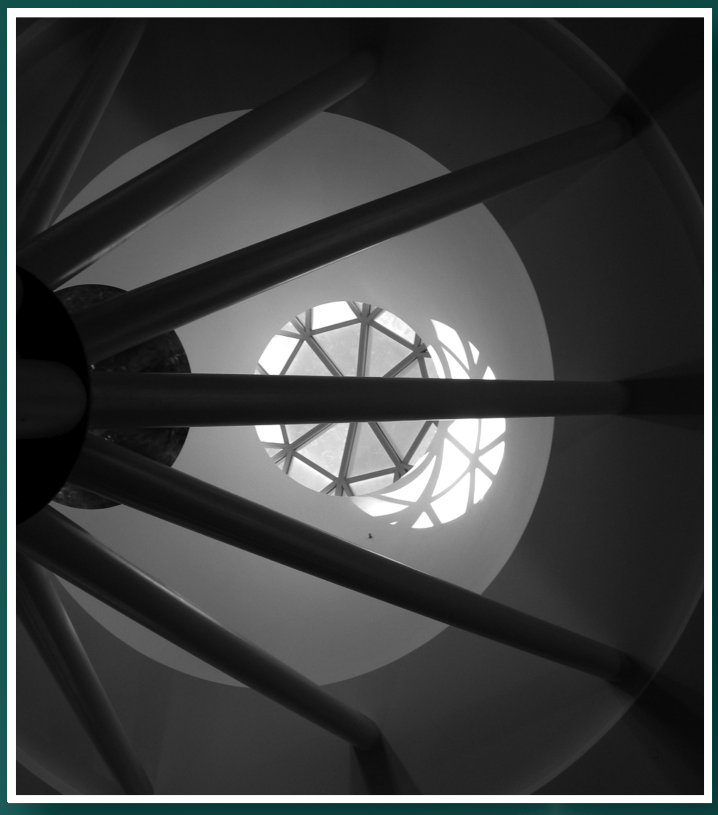

Chapter 15

Supporting Practices

15.1 Service Profiles

15.2 Vocabularies

15.3 Organizational Roles

Each of the following recommended practices can be considered an additional "best practice" in its own right in that each provides a proven approach or consideration in support of applying service-orientation design principles.

The practices associated with vocabularies and roles in particular raise issues that can be addressed during SOA planning stages in preparation for subsequent service analysis and design projects.

15.1 Service Profiles

When collecting discoverability-related meta information, it is helpful to use a standardized template or form that ensures the same type of data is documented for each service. This can be especially useful during the early analysis stages, when service candidates are just being conceptualized as part of the service modeling process. The document used to record details about a service is the *service profile*.

Figure 15.1 illustrates how the service profile emerges from the initial analysis phases, but can then continue to accompany a service as it progresses through subsequent design, delivery, and governance stages. A service profile can very much become a living document that is owned and maintained by service custodians.

Once a service is finally deployed, some organizations transpose the contents of the service profile to the service registry, whereas others choose to keep the service profile as a separate document (in which case the service's registry record may include a pointer to the location of the profile document).

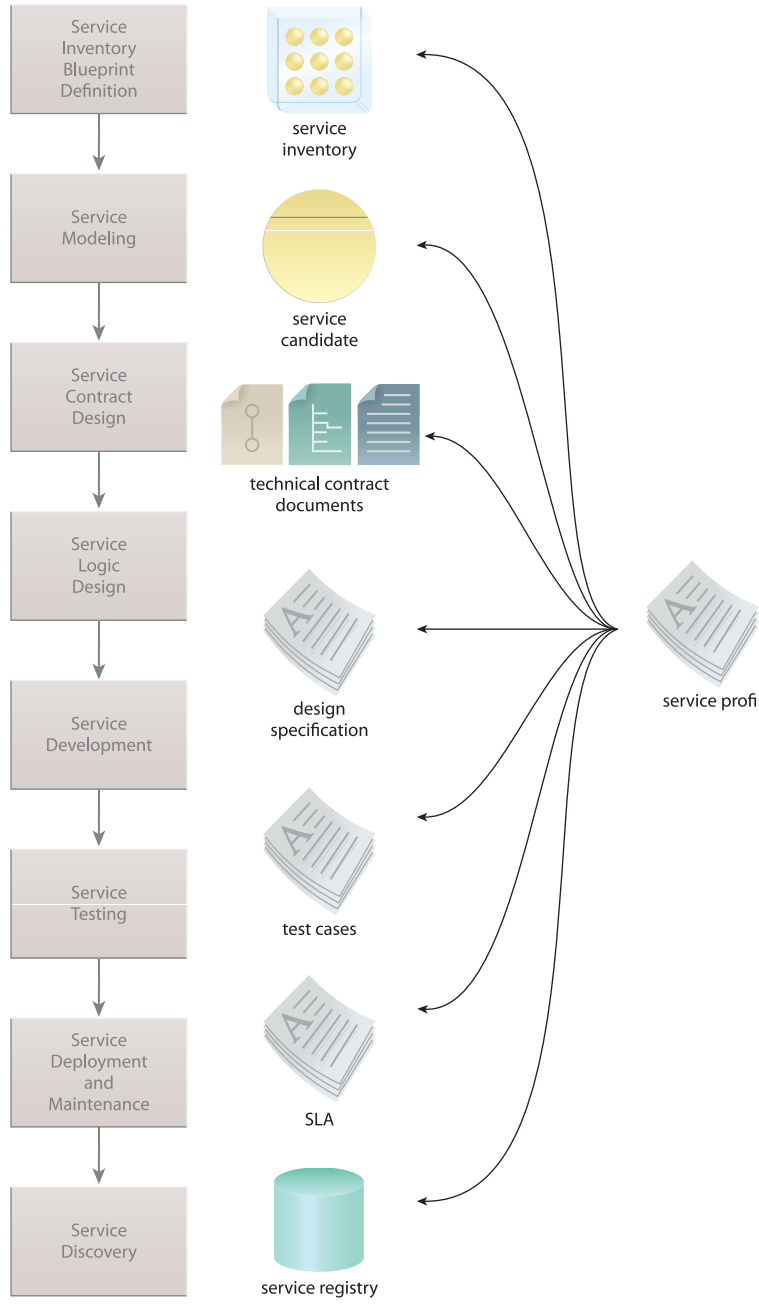

Figure 15.1
A service profile initially acts as a repository of meta information when a service is first conceptualized during early analysis stages and then provides valuable details for design and delivery-related documents used during later lifecycle phases.

Service-Level Profile Structure

There is no one official profile format for the service profile. However, with an understanding of how services typically evolve throughout project delivery lifecycles, the following baseline fields are recommended: -

- *Service Name*
- *Purpose Description (Short)* - A concise, one sentence description of the service context and purpose.
- *Purpose Description (Detailed)* - A full explanation of the service context and its functional boundary with as many details as necessary.
- *Service Model* - Entity, Utility, Task, Orchestrated Task, or a custom variation.
- *QoS Requirements* - This field captures various anticipated quality of service requirements, characteristics, or limitations that affect the service as a whole. Examples include security, performance, availability, and transaction requirements (each of which could easily justify its own field in the profile).
- *Capabilities* - The profile should document capabilities that exist and are in development, as well as those that are only planned and tentatively defined. Color coding is often useful to make these distinctions as is the use of the capability "status" field (described shortly).
- *Keywords* - This field can contain one or more keywords ideally taken from an official service inventory-level taxonomy or vocabulary. Service profile keywords should correspond to the keywords used by a service registry.
- *Version* - The version number of the service currently being documented is noted here. Depending on the version control system in use, version numbers may only be applicable to service capabilities.
- *Status* - The development status of the service (or service version) is expressed in this field using standard terms identifying a project lifecycle stage, such as "analysis," "contract design," "development," or "production." If the service is not in production, it can be helpful to include an estimated delivery date.
- *Custodian* - Details on how to reach the official service custodian or owner, as well as others that contributed to this documentation.

Capability Profile Structure

Because a service acts as a container for a collection of capabilities, additional "sub-profiles" need to be established to represent each individual capability separately, as follows:

- *Capability Name*
- *Purpose Description* - A concise explanation of the capability's overall purpose and functional context (similar to the short service description).
- *Logic Description* - A step-by-step description of the logic carried out by the capability. This can be supplemented with algorithms, workflow diagrams, or even entire business process definitions, depending on what stage the capability definition is at.
- *Input/Output* - These two fields provide definitions of a capability's allowable input and/or output value(s) and associated constraints. It can be helpful to describe these in plain English during the service modeling phase. The details established here can make reference to existing schema types.
- *Composition Role* - The execution of capability logic can place a service into various temporary runtime roles, depending on its position within service composition configurations. This field can be filled out with a description of the capability's role, or it can simply contain a term used to identify predefined roles, such as those introduced in Chapter 13.
- *Composition Member Capabilities* - A list of services (and specifically their capabilities) composed by the capability logic. This provides a convenient cross-reference to other service capabilities on which the current capability has formed dependencies. Ideally, identified composition member capabilities are mapped to the portions of the business process logic (documented in the *Logic Description* field) so that delegated logic is clearly indicated.
- *QoS Requirements* - As with the corresponding field in the service-level profile structure, this field is dedicated to collecting quality of service details. However, the information documented here pertains specifically to the service capability, which means it may need to be derived from or correlated with the service-level quality of service details in some cases.
- *Keywords* - Often the same keywords that apply to the service can be carried over to the capability. But it is not uncommon for additional keywords to be added to

individual capabilities so as to better classify their purpose. Keywords for services and capabilities should originate from the same parent vocabulary.

- *Version* - Depending on the versioning system in place, capabilities themselves may be versioned with a number, or new capability versions may be added with the version number appended to the capability name.
- *Status* - The same lifecycle identifiers used for services can be applied to the status of individual capabilities. However, this field can also be used to earmark capabilities that were identified during the modeling stage but for which no specific delivery date exists.
- *Custodian* - More often than not, the custodian of the service will be the custodian (or one of the custodians) of the related capabilities. However, when multiple business and technology experts collaborate on a given service, some are only there to assist with the definition of one service capability (or a subset of service capabilities). In this case separate custodians may need to be associated with individual capabilities.

Additional Considerations

Customizing Service Profiles

What we've established so far is fundamental profile documentation. Organizations are encouraged to customize and extend this to whatever extent required. Each of the principles covered prior to this chapter provided the option of identifying additional types of meta information, primarily associated with the extent to which principle characteristics were implemented.

Therefore, when documenting a service at various lifecycle stages, its profile can be further appended with levels, such as those summarized in Table 15.3 in the upcoming *Vocabularies* section.

Service Profiles and Service Registries

Much of the information assembled into service profiles will form the basis for service registry records. Depending on whether a service registry exists within an organization at the time the profile is being defined, it is advisable to become familiar with the registry product's record format. This will allow you to better align the service profile template with how the profile information may need to be represented within the service registry.

Service Profiles and Policies

While the WSDL and XML schema definitions will often naturally emerge from existing data models, design standards, and the interoperability requirements documented as part of the service-oriented analysis phase, policy definition is not always as straight forward. Much of the information collected in the service profile document (especially as part of the QoS fields) can form the basis for policies.

It is up to those that shape the full service contract (which, as originally illustrated in Figure 6.2, is comprised of technical and non-technical documents) to decide whether a given policy should be expressed via a technical syntax, such as the WS-Policy language, whether it is better represented within an SLA, or whether it should be part of the service contract at all.

Of course another important piece of information that needs to be kept within service profiles is any *existing* policies that are identified as pertaining to the service or any one of its capabilities. An additional field dedicated to providing a link to relevant (technical or non-technical) policies may be warranted within enterprises that have several centralized policies already in use.

Service Profiles and Service Catalogs

The structure of a service profile is ideally standardized so that different project teams consistently document the services they deliver. As more service profiles are created, they can be assembled into a *service catalog*. A service catalog is essentially a documentation of the services within a service inventory (much the same way a product catalog may describe the inventory of items a company has in its warehouse).

If an organization is creating multiple domain service inventories, each with its own design standards and governance processes, then service profile structures may vary. Therefore, a separate service catalog is generally created for each service inventory.

SUMMARY OF KEY POINTS

- As services move from concept to candidate to physical design, it is important to consistently document them using standardized service profiles.
- The use of service profiles is most effective when combined with a standardized vocabulary or taxonomy.
- Service profile documents can be compiled into an inventory-specific service catalog.

15.2 Vocabularies

When services are delivered by various project teams, the need for consistency in how service characteristics, contexts, keywords, and other forms of meta information are labeled and classified is paramount. If different teams use different conventions, it can jeopardize the potential for services to be repeatedly composed and can further burden the governance of service inventories.

In relation to service-orientation design principles, the following vocabularies are relevant and should always be standardized:

- Service-Oriented Computing Terms
- Service Classification Terms
- Design Principle and Characteristic Types, Categories, Labels
- Design Principle Application Levels
- Service Profile Keywords

The next set of sections revisits some of the terms, labels, and categories described in earlier chapters to provide an overview of the vocabularies established in this book. Each of these vocabularies can be further customized and extended for specific enterprise environments. The key is to do so consistently and make the official vocabularies widely available to all relevant project team members.

Service-Oriented Computing Terms

The following set of terms represents the fundamental taxonomy that establishes the core elements and parts of a typical service-oriented computing platform:

- Service-Oriented Architecture
- Service-Orientation Design Paradigm
- Service-Orientation Design Principles
- Service-Oriented Solution Logic
- Service
- Service Model
- Service Composition
- Service Inventory
- Service Inventory Blueprint

These terms are defined and described in Chapters 3 and 4.

Service Classification Terms

Table 15.1 lists the core service models referenced throughout this book and also provides alternative industry terms. (Service models were first introduced in Chapter 3.)

Service Model	Classification	Alternative Terms	Corresponding Service Abstraction Layer
Entity Service	Business, Agnostic	Entity-Centric Business Service Business Entity Services	Entity Service Layer
Utility Service	Non-Business, Agnostic	Application Service Infrastructure Service Technology Service	Utility Service Layer
Task Service	Business, Non-Agnostic	Task-Centric Business Service Business Process Service	Task Service Layer
Orchestrated Task Service	Business, Non-Agnostic	Process Service Business Process Service Orchestration Service	Parent Business Process Layer Orchestration Layer

Table 15.1
The terms used to represent these fundamental service models also carry over to how the corresponding service abstraction layers are labeled.

Types and Associated Terms

Various terms were established in Chapters 5 through 13. Some defined types of design characteristics, whereas others provided categories of relevant information, as listed in Table 15.2.

Design Principle	Types
All	Service Granularity Capability Granularity Data Granularity Constraint Granularity
Standardized Service Contract	Functional Service Expression Standardization Data Representation Standardization (or Data Model Standardization)
Service Loose Coupling	Logic-to-Contract Coupling Contract-to-Logic Coupling Contract-to-Technology Coupling Contract-to-Implementation Coupling Contract-to-Functional Coupling Consumer-to-Implementation Coupling Consumer-to-Contract Coupling
Service Abstraction	Technology Information Abstraction Functional Abstraction Programmatic Logic Abstraction Quality of Service Abstraction
Service Reusability	n/a
Service Autonomy	Runtime Autonomy Design-Time Autonomy
Service Statelessness	Active and Passive (Primary States) Stateful and Stateless (Primary State Conditions) Context, Session, and Business (State Information Types) Context Data and Context Rules (Context Data Types)
Service Discoverability	Design-Time Discovery Runtime Discovery Functional Meta Data Quality of Service Meta Data

Design Principle	Types
Service Composability	Primitive Composition Complex Composition Service Activities Composition Controller Composition Sub-Controller Designated Controller Composition Member Composition Initiator Composition Instance Composition Member Capability Point-to-Point

Table 15.2

Collections of related terms used to classify various types of characteristics and information. (Note that the granularity types listed in the first row were introduced in Chapter 5.)

Design Principle Application Levels

Several of the chapters in this book provided suggested labels to communicate to what extent a principle was applied to a service capability or to the service as a whole. Table 15.3 summarizes these levels.

Design Principle	Levels
Standardized Service Contract	Levels Dependent on Design Standards
Service Loose Coupling	Non-Centralized Consumer Coupling Centralized Consumer Coupling (plus numeric rating)
Service Abstraction	Detailed Contract Abstraction Concise Contract Abstraction Optimized Contract Abstraction Mixed Detailed Contract Abstraction Open Access Controlled Access No Access
Service Reusability	Tactical Reusability Targeted Reusability Complete Reusability

Design Principle	Levels
Service Autonomy	Service Contract Autonomy Shared Autonomy Service Logic Autonomy Pure Autonomy
Service Statelessness	Non-Deferred State Management Partially Deferred Memory Partial Architectural State Management Deferral Full Architectural State Management Deferral Internally Deferred State Management
Service Discoverability	Custom Rating System
Service Composability	Custom Rating System (for composition design, composition runtime, and composition governance stages)

Table 15.3
Some design principles provide specific, measurable application levels, while others provide suggested rating systems that depend on environment-specific factors.

SUMMARY OF KEY POINTS

- Establishing a standard vocabulary of terms used for classification and communication purposes can streamline the delivery of services.
- This book provides numerous terms and classifications that can be further extended or customized.
- Vocabularies are ideally distributed to and used by all project team members.

15.3 Organizational Roles

As explained in Chapter 4, applying service-orientation design principles on a broad basis changes the complexion of an IT enterprise. Organizational structures and project delivery lifecycles and processes are affected and subjected to changes, as are ownership and governance responsibilities and priorities.

Changes on an organizational level result in changes to those who work within the organization.

Traditional IT positions are impacted as the need for new roles emerges in response to the distinct requirements associated with the delivery, deployment, and maintenance of services, service inventories, and service-oriented technology architecture implementations (Figure 15.2). It is important to gain an understanding of these new roles as early on in the delivery lifecycle as possible so that project teams are fully prepared.

Provided in this section are descriptions for the following set of common roles:

- Service Analyst
- Service Architect
- Service Custodian
- Schema Custodian
- Policy Custodian
- Service Registry Custodian
- Technical Communications Specialist
- Enterprise Architect
- Enterprise Design Standards Custodian (and Auditor)

Note that this list is limited to roles associated specifically with the application of service-orientation design principles as they relate to the aforementioned deliverables and delivery stages.

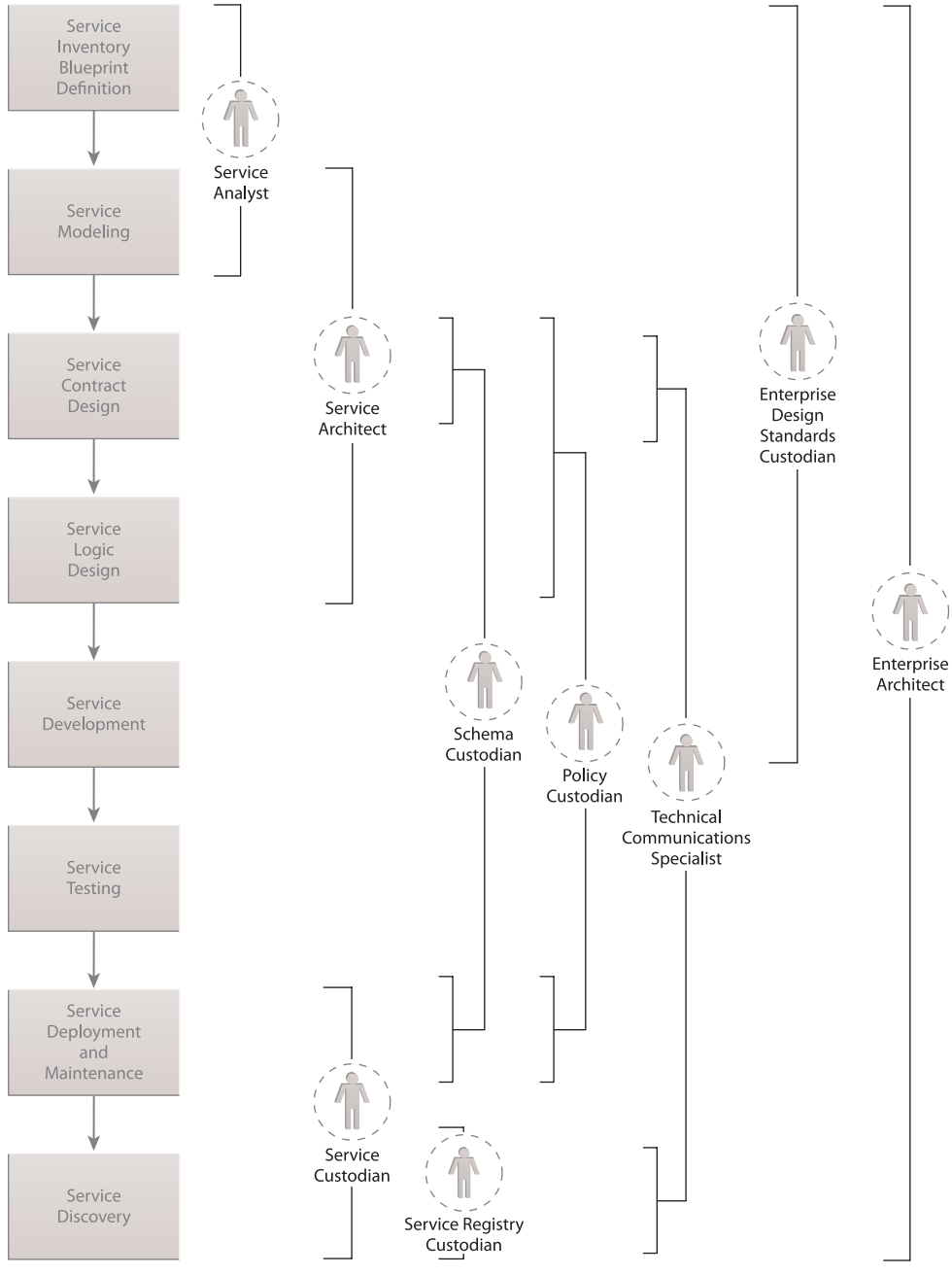

Figure 15.2

Common roles associated with service-orientation can be required in various stages of a typical service delivery lifecycle.

Service Analyst

This role requires expertise in the definition of service candidates, service capability candidates, and service composition candidates. A service analyst is therefore proficient in all aspects of the service-oriented analysis process, including the delivery of service candidates through the service modeling process.

The service analyst role can be assumed by architects and business analysts that participate in a project's service-oriented analysis phase. Alternatively, it can form the basis of a team leader role within this process, essentially a specialist in service-oriented analysis that coordinates and leads architects and business analysts throughout all process steps. The latter variation can be very effective in larger enterprise environments where every iteration through a business process can require the participation of different business and technology subject matter experts.

Principles most associated with this role: Service Reusability, Service Autonomy, Service Discoverability

Service Architect

The service architect is primarily concentrated on the physical design of services. Therefore, this role is more associated with the service-oriented design process and the various service model-specific service design processes an organization may be using.

Service architects are enlisted when an organization is ready to proceed to the design and development stages of an SOA initiative. They essentially use the service candidate definitions as a starting point, apply related design standards and conventions, and deliver service contract and logic designs.

The actual development of the contract and logic may be carried out by development teams. However, service architects proficient with contract technologies may assume the responsibility of delivering the technical contracts themselves. Furthermore, service architects may be required to contribute to service design standards as well.

Depending on the scope of a service delivery project, the same individual may be able to assume both service analyst and service architect roles.

Principles most associated with this role: All

Service Custodian

A service custodian owns the governance responsibilities of one or more specific services. These duties do not just revolve around the extension and expansion and maintenance of service logic, but also include having to protect the integrity of the service context and its associated functional boundary. Therefore, a service custodian can take ownership of a service as early as when its context is defined (and verified) during the service-oriented analysis stage.

Service custodians are important to the evolution of agnostic services. Their involvement ensures that no one project team inadvertently skews the design of an agnostic service in favor of their requirements. They are furthermore responsible for hiding non-essential information about service designs from the outside world (as per the access control levels established by the Service Abstraction principle). As a result, service custodians often require a good amount of authority.

Note that depending on how service details are documented, a service custodian may author, own, and maintain a service's corresponding profile document.

Principles most associated with this role: All

Schema Custodian

Originally established in the book *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services,* this role is still very much required for the governance of environments where services are delivered as Web services. The flexibility provided by the Web services framework to allow a data representation architecture (comprised of XML schemas) to be created and standardized independently from the service layer enables schemas to be separately defined and maintained. Ideally, this role is assumed by data analysts or other types of specialists with an intimate knowledge of an organization's information architecture.

The need for XML schema language expertise is a key prerequisite of this role. Not only are schema custodians often called upon to deliver new standardized XML schemas, they are also responsible for augmenting or extending schemas in response to changing business requirements (which also leads to the need to manage schema versions).

In support of realizing service-orientation, schema custodians ensure that service contract schemas are properly positioned as standardized and centralized parts of service inventories. Schema custodians may even own design standards pertaining to data representation.

Principles most associated with this role: Standardized Service Contract, Service Loose Coupling, Service Abstraction

Policy Custodian

Although this role can be assumed by the same person acting as the Schema Custodian, it is not uncommon for different individuals (or even different groups) to be responsible for defining and maintaining policy assertions for Web service contracts. Often these technical policy expressions are tied to existing security polices, in which case their need may not actually be identified until later in the project delivery lifecycle when the actual service logic is being designed. Other forms of policies, such as those that express a proprietary assertion syntax to represent specific business rules and policies, may be defined and owned by a combination of technical and business professionals.

Because service polices can be tied to existing corporate policies, they may be subject to more change than other parts of the service contract. Therefore, their initial definition is important to avoid embedding too much potentially volatile policy logic in the service contract. Similarly, their subsequent governance is also important to ensure they are kept in alignment with the actual polices they may have been derived from.

Overall, policy management can turn into a significant responsibility that can involve subject matter experts representing various IT departments. The document *Guidelines for Policy Assertion Authors* is a useful resource published by the W3C as a supplement to the WS-Policy specification (see `www.soaspecs.com`).

Principles most associated with this role: Standardized Service Contract, Service Loose Coupling, Service Abstraction (Note that other design principles can be affected when policies are used to express details about a service's underlying logic and behavior.)

Service Registry Custodian

Once a service registry is introduced into an enterprise, it will need to be religiously administered by one or more qualified individuals. If the content in the registry is ever allowed to go stale or somehow becomes inaccurate, the registry itself loses significance as a central part of the SOA infrastructure.

The service registry custodian is tasked with the overall administration of one or more private service registries. This goes beyond the installation and maintenance of the registry product, it encompasses the constant responsibility of ensuring a high quality of registry record content, which ties directly into how discoverability-related meta information is defined and recorded for individual services.

Although service registry custodians will typically not author discoverability content themselves, they will often own standards or conventions that dictate the nature of meta data used to populate service registry profile records.

Principles most associated with this role: Standardized Service Contract, Service Discoverability

Technical Communications Specialist

As explained in Chapter 12, the communications quality of service meta information can often be questionable. Although technically and business-wise accurate, comments, annotations, and general information within the service profile document can lack the clarity required for discovery and interpretation by a broader audience.

A technical communications specialist is usually someone with a background in technical writing who is enlisted to refine initial drafts of service profiles and associated meta data. The responsibility of this individual is to express discoverability information in plain English, using standard vocabularies so that a range of project team members can effectively query and understand service contracts and associated profiles.

Principles most associated with this role: Service Discoverability

Enterprise Architect

Although this is not a new role by any means, it represents a position that is greatly emphasized by the cross-application (cross-silo) scope of service inventory delivery projects.

Technology architects with an enterprise perspective are expected to:

- author or contribute to enterprise design standards
- become involved in service delivery projects to ensure that agnostic services are properly positioned
- assess service runtime usage and determine required infrastructure
- evaluate security concerns of individual service capabilities
- help define and perhaps even own service inventory blueprints

As discussed in the *Governance Concerns* section of Chapter 9, the demand for enterprise-centric resources can dramatically increase in service-oriented environments. This may very well require that existing enterprise architecture groups be expanded.

In larger organizations there may also be the need for *enterprise domain architects*—a variation of this role that specializes in a particular segment of the overall enterprise. These architects would then be focused on the definition and governance of domain-specific service inventories.

Principles most associated with this role: All

Enterprise Design Standards Custodian (and Auditor)

As explained in the *Using Design Principles* section of Chapter 5, it is beneficial to derive design standards from service-orientation design principles so that the principles are consistently realized across all services.

Furthermore, as enterprise architecture groups grow in response to the changes incurred by an SOA transition, design standards can be authored by multiple experts, each contributing conventions associated with a particular aspect of service design (such as security, performance, transactions, etc.).

To ensure that design standards are kept in alignment and used wherever appropriate, it may very well be necessary to establish an official custodian. This individual would be responsible for the evolution of the design standards but also for their enforcement. Therefore, this role often involves performing audits of proposed service or service-oriented solution designs.

The authority required to carry out auditing responsibilities can sometimes raise concerns within IT environments not accustomed to such formal use of design standards. Therefore, this role can sometimes be more successfully established within the boundaries of a specific enterprise domain, where a given set of standards applies only to a specific domain service inventory, not the enterprise as a whole.

Principles most associated with this role: All

NOTE

This list does not represent all possible roles associated with an SOA transition initiative. A title dedicated to SOA Governance is planned for the *Prentice Hall Service-Oriented Computing Series from Thomas Erl* in which organizational roles will be comprehensively explored and defined and also associated with appropriate governance processes.

SUMMARY OF KEY POINTS

- Service-orientation brings with it a shift toward an enterprise-centric perspective when it comes to delivering solution logic.
- Various new roles can be defined in support of applying service-orientation principles in analysis, design, and governance capacities.