CHAPTER 1

INTRODUCTION

The dust has finally settled. The computer evangelist has envisioned. The strategist has defined. The manager, architect, developer, analyst, and modeler have taken note. The enthusiasm has diminished. The exuberance has turned into a more pragmatic course of action. But what have we learned? Almost a decade has passed since the service-oriented architecture (SOA) paradigm first appeared, extending a pledge to change, repair, enhance, and promote software development best practices. Has this computing trend influenced any of our customary software implementation approaches? Has this new paradigm fostered architecture best practices that have altered the old genre of software construction methodologies? Are we better off?

Yes, the mission to build enduring, reusable, and elastic services that can withstand market volatility and be effortlessly modified to embrace technological changes has been gaining momentum. Governance and best practices have been devised. Existing off-the-shelf products already embody fundamental SOA best practices, such as software reuse, nimbleness, loose coupling, adaptability, interoperability, and consolidation. From a project management perspective, stronger ties have been established between the business institution and information technology (IT) organization. The silo implementation paradigm has been rather weakened, and cross-enterprise initiatives have been moderately increased.

For those organizations that have not been influenced by these trends, the simple advice is to start from smaller "wins." Avoid grandiose SOA plans to change the world. Promote small-size projects, incremental initiatives that over time attain final objectives. Doing this will clearly reduce implementation risks and increase productivity and improve the chances for success. But now it is time to get to work. Roll up your sleeves and embrace a positive, creative, vibrant, and contagious spirit that encourages business and IT personnel to collaborate on fulfilling organizational aims.

The service-oriented discovery and analysis discipline[1] mirrors this approach: Smaller and incremental efforts to achieving success are preferred over large-size projects that introduce perils to business execution. Take tiny steps toward devising a solution to an organizational concern that simplify rather than complicate design and architecture models, clarify rather than obscure technical specifications, and focus service construction efforts rather than blurring implementation boundaries.

The service-oriented discovery and analysis discipline can indeed guarantee success if the goals are well established and agreed upon. Therefore, begin a service-oriented architecture project on a sound foundation of analysis that not only identifies the problem domain but also assesses the feasibility of a proposed solution. Verification of the constructed services against business requirements and technological specifications is another vital contribution of the analysis process. Furthermore, start the project with the service discovery process and justify a service's existence. Persist with service identification throughout the service life cycle, and always embrace new discovery opportunities as they come along. The final artifact of this exercise is an analysis proposition that calls for service modeling activities to deliver a superior solution.

Finally, adopt a personal as well as an organizational strategy to ascertaining services, analyzing problems, and modeling solutions. Formulate an individual plan that can assist with pursuing the service-oriented discovery and analysis venture. Leverage a systematic approach for service identification, examination, and modeling. Do not overlook perhaps one of the most imperative guiding principles: Foster transparency during the discovery and analysis endeavor. The term "transparency" is about tracing and reporting on decisions that are being made during the discovery, analysis, and modeling process in terms of their business practicality and technological feasibility.

WHAT IS SERVICE-ORIENTED DISCOVERY AND ANALYSIS?

Often mentioned in one breath, the discovery and analysis of services can be separable, yet these processes share interrelated milestones and goals that unite them, forming a single service-oriented life cycle discipline. The discovery effort typically is affiliated with service identification. This is a rudimentary and continuous pursuit that yields solutions, namely services, to address an organizational concern. The analysis venture, in contrast, is a study. It is practiced independently; exercised in different phases of a service's life cycle; conducted to understand a problem domain, to ascertain and assess a solution, to verify if service capabilities meet requirements, and to authenticate service operations for production.

SERVICE-ORIENTED ANALYSIS ENDEAVOR

Among other reasons for conducting a meticulous business and technology inspection, the analysis process is chiefly a *quest for a solution* to an organizational problem that is carried out by identified services. No matter when in the life cycle of a service it is pursued, the ultimate goal of the service analysis venture is to identify the best possible remedy to a *business concern* or a *technological challenge*. Obviously, a study of a wide range of artifacts and events that are about to affect a business or have already begun to influence its performance is the driving motivation of the analysis process. However, the engagement of analysts, modelers, architects, developers, or managers in rigorous examination of the business does not necessarily take place because of immediate influences on business execution. Assessment and inspection activities can also be launched to better comprehend success, foster growth, and, most important, prevent failure of business missions.

A METICULOUS STUDY: WHAT SHOULD BE ANALYZED? The pursuit of a solution that is presented by services to mitigate an impending problem, forecasted challenge, or already affected business or technological mission must be accompanied by a thorough analysis of business and/or technological artifacts. The two perspectives presented are the chief ingredients of almost any service-oriented analysis process. Thus the practitioner ought to focus on intrinsic questions to promote the inspection venture: What is the business or technological problem domain? What are the causes for the occurring challenge? Is a solution needed? Has a solution been proposed? How soon should remedy be applied? What aspects of the problem should be rectified? Are there any business requirements or technical specifications that offer a direction for delivering a solution? and more.

Business Perspective. To alleviate a challenge, the search for a solution may start with the analysis of the business. This direction advocates that business artifacts should be scrupulously studied to identify the cause of a concern. The term "artifacts" pertains to affiliated business articles that can provide clues to the state of the business, its goals, missions, and imperatives. This may include business strategies and the model of business execution, business processes and related underpinning activities, and even market and customer segmentation research.

If a business solution proposition has already been issued, deliverables such as business requirements and product descriptions should be available to the analysis process. These are

essential documents that embody the problems, concerns, and introduced remedies of the problem domain organization. The problem domain institution typically consists of business personnel, analysts, business architects, managers, budget experts, and executives that are assigned the duties of recommending an approach, a direction, and a solution to rectify an organizational concern.

Technological Perspective. The service-oriented analysis endeavor should not stop short of technological capabilities' inspection if the problem domain stems from applications, services, middleware, and infrastructure discontinuity of operations that ultimately impair transaction performance. Moreover, the examination of technical properties and facilities typically conduced by the solution domain organization (often affiliated with the IT personnel) focuses on two major investigation aspects: internal construct of a software executable and its hosting environment (external landscape). The former pertains to the study of an internal application or service design. This may include understanding internal components' relationships, interaction of operations, and assessment of offered capabilities. The technical environment in which applications and services operate is the second vital ingredient of the analysis process. This is related to the architecture that drives the distribution of software assets across the organization.

Additionally, the distribution and deployment of services in the enterprise broadens the analysis efforts. This may involve the investigation of a large number of technological concerns that are related to service ecosystem integration, collaboration, configuration, and interoperability. These subjects of exploration characteristically commission architects, developers, analysts, modelers, and managers to focus on the landscape and the production environment that hosts service capabilities. Hosting responsibilities include middleware and infrastructure platforms to enable message exchange and transaction execution. Service mediation and federation are other articles for investigation that practitioners should pay close attention to.

SERVICE ANALYSIS PROCESS. During the service-oriented development life cycle, the opportunities of conducting analysis sessions are vast. "Analysis paralysis" situations that pertain to long studies of business or technological artifacts that do not yield tangible and practical deliverables should be avoided. Moreover, excessive investigations of product artifacts, business requirements, service capabilities, environment configurations, architecture and design blueprints and deliverables alike can immensely prolong the software development life cycle.

Therefore, the rule of thumb suggests that practitioners should initiate an analysis session for each stage of a service life cycle. This is an iterative examination approach that commences during a project's inception, continues through design time, persists throughout service construction, and concludes when a service is deployed and managed in a production environment.[2] In each of these project phases, a distinct type of analysis should be launched. For example, during project inception, a practitioner should study requirements and identify the motivation for employing a service to provide a solution. Conversely, the analysis process that takes place during the service contraction addresses different concerns: It is devised both to verify if a service satisfies business or technical requirements and to guarantee high quality of operations.

Exhibit 1.1 illustrates four analysis iterations that can be conducted during a service's life span:

1. The *inception analysis* iteration characteristically takes place before a service has been proposed.
2. *Assessment analysis* is conducted after business and technical requirements propose a solution.
3. *Verification analysis* is pursued after a service has been constructed.
4. *Authentication analysis* is required before a service is deployed and persists during service operations in production.

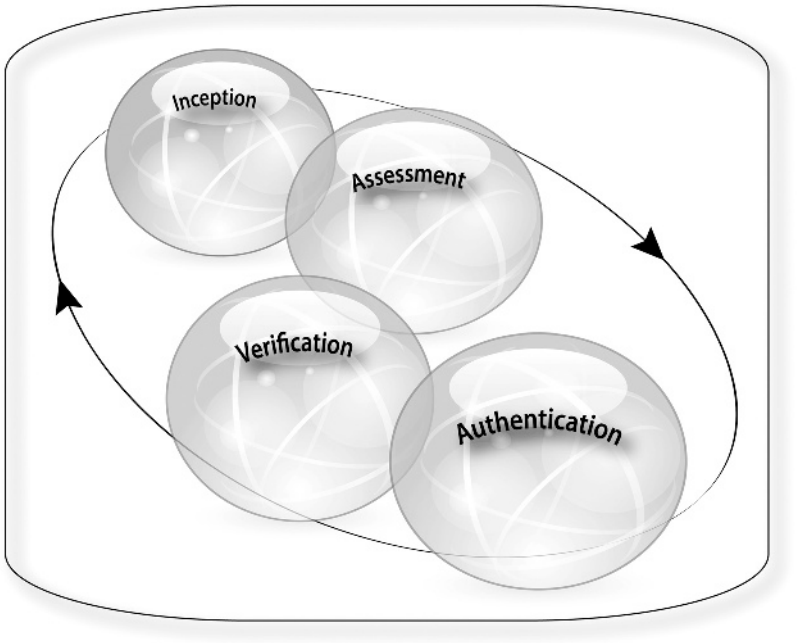

EXHIBIT 1.1 SERVICE-ORIENTED ANALYSIS ITERATIONS

The sections that follow depict an iterative process of service analysis during which an examination of a service and its hosting environment is conducted to achieve distinct analysis goals.

Inception Analysis Iteration: Analyze Prior to Service Proposition. The analysis process that is pursued before a service has been proposed is driven by strategic imperatives that call for investigating the state of the business, tracing events that have influenced business execution, and identifying the chief concerns of an organization. This venture typically occurs before or at some point in the *inception* phase of a project, during which ideas are gathered and discussions take place to establish a firm direction for addressing problems and identifying *motivations* for instituting services. Furthermore, an analysis session that is performed before a service begins its life cycle or during its *inception* stage is typically dedicated to founding service concepts that eventually will materialize and be transformed into tangible services.

Assessment Analysis Iteration: Analyze after Service Proposition. Once a service has been proposed and a solution is at hand, a different analysis iteration can begin. This is the time to study business requirements and learn about a software product description, features, attributes, and capabilities. However, if a technological proposition advocates creation of a service and expanding its operating environment, technical specifications should be studied and analyzed. This may include the inspection of service *design and architecture* artifacts and analysis of the introduced technological landscape. Therefore, the *assessment* of the proposed business or technical solutions should be focused on a service's feasibility to rectify an organizational concern and offer practical remedies.

Verification Analysis Iteration: Analyze after Service Construction. Once a service has been contracted, it should be analyzed to assess the quality of its capabilities, functionality, and operations.

This service *verification* process must also confirm if the service design and architecture model meets business requirements or technical specifications. In other words, the major queries that typically should be asked during this evaluation activity are related to a service's capabilities to satisfy its corresponding consumers' necessities: "Can a service's offerings alleviate the identified business challenges?" "Does the service provide a technological solution?" "Does the service design model adhere to organizational best practices?"

Authentication Analysis Iteration: Analyze before and during Service Operations. To ensure flawless execution of a service in a production environment, a service should be certified. This is another analysis effort that should take place before a service is ferried out to production and assumes the responsibilities that it was designated. Thus the *authentication* process should be broadened to include the inspection of internal service functionality as well as external vital aspects of service integration, collaboration, interoperability, and configuration. Furthermore, the certification of a service should not conclude with its deployment effort to a production environment. This process must persist to meet the challenges that a service might face in production, during which volatile market conditions, changes to the business mission, technological discontinuity, and disruptions to transaction processing might occur.

SERVICE ANALYSIS APPROACH. Finally, when pursuing the service analysis process to assist with carving out a solution, adhere to a methodical approach that not only offers direction and milestones to attain, but also underlines the importance of achieving predefined goals. The quest for feasible services that can provide a satisfying remedy to an organizational concern then must be driven by an analysis *road map* that is lucid and unambiguous to analysts, architects, developers, modelers, and managers. The starting point, midpoints, and end point of the service inspection endeavor must be crystallized by employing repeatable *patterns*. These are guiding templates that are accepted by the governance body of an organization whose duties include carving out best practices and standards to streamline the analysis process.

In the sections that follow we introduce a general template for pursuing a service-oriented analysis venture.[3] This is a guiding road map that identifies the four major perspectives of service inspection process that covers vital aspects of internal and external service design and architecture. Exhibit 1.2 illustrates this concept. Note that in addition to the four recommended analysis orientations, the practitioner should adhere to the overarching organizational practices that help shape a service-oriented analysis venture:

1. Look in the Box.
2. Look out of the Box.
3. Look above the Box.
4. Look below the Box.

Look in the Box. The solution that we are commissioned to propose is obviously executed by internal service operations. This service internal functionality embodies business and/or technical processes and is the manifestation of business requirements and technical specifications. Therefore, the practitioner should look in the "solution box" to analyze the service's internal affairs and the contribution of its capabilities to rectifying an organizational problem. While analyzing a service's internal construct, service interfaces, communication mechanisms, and binding contracts with corresponding consumers are also vital considerations that must be examined. Remember, these internal service examinations that should take place during the inception, assessment, verification, and authentication service-oriented analysis iterations should persist during a service's life cycle to perfect its operations, boost its reusability rates, and increase its performance abilities.

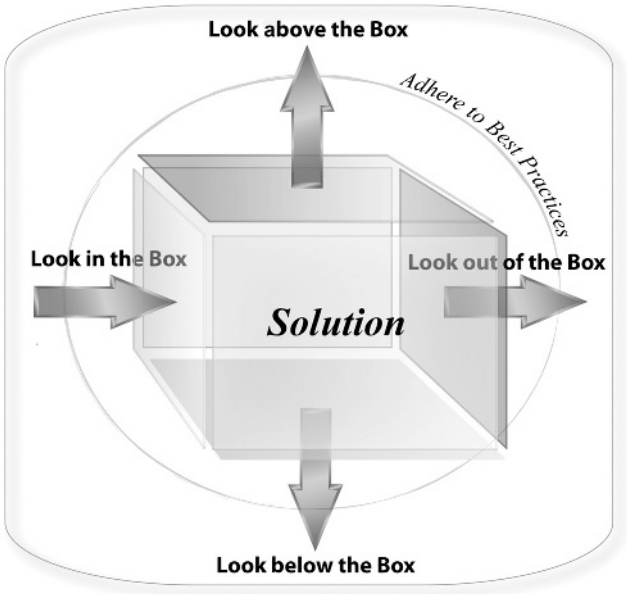

EXHIBIT 1.2 SERVICE-ORIENTED ANALYSIS ROAD MAP

Look above the Box. A practitioner should also look above the solution box to study the overarching governance standards and policies that influence service design and architecture. This analysis effort should also focus on understanding documented and undocumented business processes and exploring the manner by which they can be translated to service capabilities. To synchronize these processes and harmonize transactions, it is time to start planning an orchestration model for governing business activities across the organization. Another goal that should be attained by looking above the solution box is to get acquainted with the organizational service life cycle stages and comprehend the various transformations that a service must undergo. Among these metamorphosis activities is broadening service capabilities and functionality. This can be fulfilled by embarking on an analysis effort to help practitioners generalize and abstract services beyond their existing boundaries.

Look below the Box. The quest for a solution to an organizational concern should carry on by looking below the solution box. The term "below" identifies another perspective that must be inspected to ensure a thorough analysis process, during which the fundamental aspects of service design and architecture are scrutinized and justified. One of the leading practices that should be exercised when looking below the solution box is separation of concerns. This is an investigation effort that focuses on breaking up organizational problems into manageable units of analysis to enable loosely coupled solutions that ultimately are carried out by services.

Another important analysis activity that should take place when pursuing the below-the-box direction is the evaluation of service granularity. This exercise fosters a balanced construction of services across the organization by standardizing an allowable service size and the magnitude of their operations. In other words, striking a balance between fine-grained and coarse-grained services by limiting their growth or contraction is the art of granularity analysis. This policy eventually promotes service decomposition activities that boost reuse, nimbleness, and software elasticity.

Look out of the Box. Finally, the out-of-the-box analysis approach engages a holistic view of a service ecosystem. This method of investigation involves a service's hosting environment and the underpinning pillars of its architecture model: service integration, interoperability, mediation, deployment, configuration, distribution, and federation. Furthermore, to better understand a service's landscape and help in carving out a message exchange and transaction model for a service provider and its corresponding consumers, the out-of-the-box direction focuses on two major scenarios of service-oriented implementation: contraction and expansion. The former identifies the necessities of an organization to minimize or limit the scope of architecture, while the latter addresses requirements for expanding service boundaries and offerings, and increasing a service's consumer base.

SERVICE-ORIENTED DISCOVERY ENDEAVOR

The quest for a solution brings us to the point of service discovery. This is the process of identifying, reusing, or collecting ideas, processes, and capabilities to alleviate an organizational concern. "Identifying" means that the practitioner discovers a solution to the problem by proposing a remedy, namely a service. The term "reusing" pertains to leveraging the functionality of an already existing service to resolve a problem. Finally, "collecting" implies that architects, analysts, developers, modelers, and managers are commissioned to devise a solution by employing a number of software entities. These may include concepts and existing legacy services and applications. Thus the software assets that are selected to participate in a solution may not be found only in a production environment: They can be in different life cycle stages, such as service conceptualization, service design and architecture, service construction, and service operations.

WHAT IS A SERVICE? Before identifying and justifying the existence of a service, an organization must define its essence and agree on what the term "service" represents. This effort is somewhat subjective because service definitions vary from one organization to another. For some, a service is just a business function that originates from a concept. According to others, a service is merely a technical term that constitutes a physical executable that is deployed to a production environment. Therefore, lack of consensus and industry standards encourage organizations to define a service based on their necessities and business or technological imperatives.

In the context of service-oriented discovery and analysis discipline, the term "service" is a holistic definition, a universal expression that depicts any software entity that offers business or technical capabilities. This can include a wide range of "things" that can be considered services, such as a concept, business process, software component, an application, a Web service, a virtual service that operates in a virtual computing environment (such as a Cloud), a COBOL program, a database, or a middleware product. Moreover, a service that is deployed to a production environment must consist of interfaces that enable it to communicate with the outside world and meet its corresponding consumers' requirements. The binding agreement between message exchange parties is known as a contract that depicts the types of offerings a service extends to the information-requesting entities.

Remember, during the service discovery process, all discovered services are merely candidate entities that may materialize into future tangible solutions. There is no guarantee that an identified service may indeed find its way to a production environment. Therefore, the process of service discovery yields an *analysis service* that represents a temporary capability employed to offer a solution.

SERVICE DISCOVERY PROCESS. Akin to the service analysis process that is discussed in the Service-Oriented Analysis Venture section, the service discovery effort should take place in different phases of a service life cycle.[4] This is an iterative process that commences at a service's inception, persists through service design, architecture, and construction, and continues when a

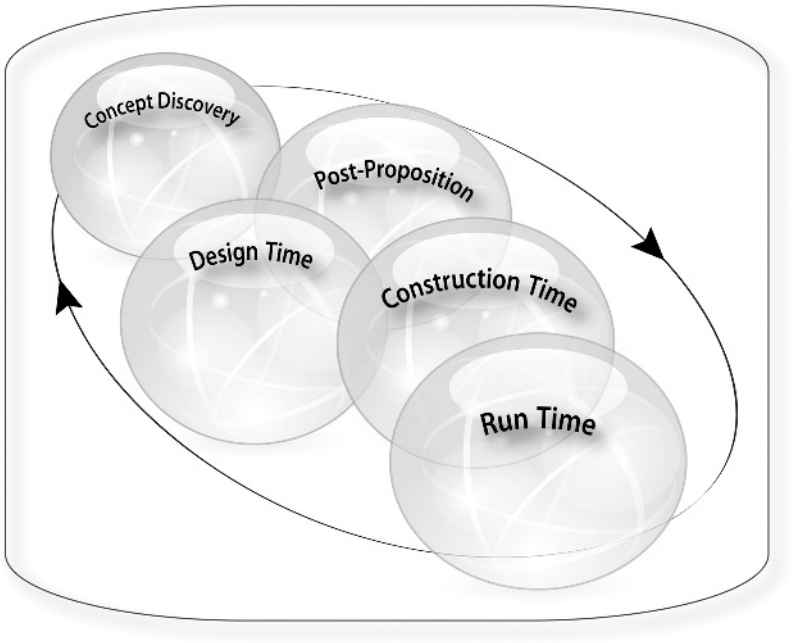

EXHIBIT 1.3 SERVICE-ORIENTED DISCOVERY ITERATIONS

service is deployed to production. This cycle of discovery can go back over the same ground if a service must undergo enhancements, redesign, or rearchitecture efforts.

Exhibit 1.3 illustrates this chain of discovery activities and identifies the five major iterations that are conducted in a service life cycle:

1. **Concept discovery** typically occurs before a solution is accepted and established, during which conceptual services are identified.
2. **Post-proposition discovery** is pursued after a solution to a problem has been proposed and adopted.
3. **Design-time discovery** is conducted during the design and architecture service life cycle phase.
4. **Construction-time discovery** obviously occurs during the construction phase of a project, during which source code is written and tested.
5. **Run-time discovery** takes place after a service has been certified and deployed to production environment.

The sections that follow elaborate on the five service discovery iterations and identify the underpinning activities that lead to identification of service capabilities.

Concept Discovery Iteration: Discover before Solution Proposition. Even before a solution has been proposed and institutionalized, the practitioner should start formalizing organizational ideas and concepts that can later serve as candidate services. This concept recognition effort is typically conducted during the service conceptualization phase, during which the problem domain is studied, remedies are discussed, resolution avenues are explored, and obviously *conceptual services* are discovered. Furthermore, the identified conceptual services typically promote a common

communication language between business and IT organizations and help establish an organizational glossary that consists of concepts that can be reused across the enterprise. Finally, from a project initiation and management perspective, the service conceptualization phase also contributes to scoping a project and setting its timeline boundaries.

Post-Proposition Discovery Iteration: Discover after Solution Proposition. The service discovery process should persist after a solution to an organizational problem has been proposed. But at this stage a different service identification approach should be applied. This new discovery iteration calls for ascertaining service capabilities and functionality that meet business requirements or technical specifications to satisfy consumers' imperatives. In other words, at this stage in a service life cycle, focus on discovery of services that are derived from business requirements and/or technical specifications. Once discovered, these offered service solutions, often referred to as candidate services, should be subject for design and architecture in the next iteration, which is discussed in the sections that follow.

Design-Time Discovery Iteration: Discover during Service Design and Architecture. Once the candidate services have been introduced, the service design and architecture phase continues. The design venture not only focuses on firming up a service's internal composition, it is also about addressing external environment challenges, such as service integration, message exchange mechanisms, and service interoperability. During this ongoing effort, new services can be identified to complement the previous service conceptualization and service proposition efforts that are discussed in the preceding sections. These newly discovered services offer capabilities to enable a wide range of solutions, such as data transformation, mediation, security, user interface, implementation of business logic, and more.

Again, during the service design and architecture stage, the opportunities for service discovery are vast and crucial to the solution because of the technological concerns that require more attention at this stage. Finally, to document the design and architecture phase and depict the discovered services, architects, developers, analysts, modelers, and managers deliver design and architecture blueprints and modeling artifacts (also widely known as technical specifications).

Construction-Time Discovery Iteration: Discover during Service Construction. The service construction stage again introduces ample opportunities for service discovery. This service development initiative is typically pursued by developers, architects, and technical managers by following design and architecture blueprints and adhering to organizational best practices and standards. Obviously, design artifacts are the driving aspects of service construction, but the developer may come across scenarios that call for the establishment of new services to fill in the gaps of the technical specifications that were previously delivered. To comply with enterprise best practices, again, the developer may find additional service identification opportunities. For example, to promote loose coupling standards, a bulky service should be broken up into finer-grained new services to carry out business or technical missions.

Refactoring is another development practice that characteristically introduces additional opportunities for service discovery. This is an exercise that is pursued to optimize, stylized, and modularize the source code to boost service performance and reusability factors. The developer is then commissioned not only to enhance operations and service interfaces; source code componentization is also required to group functionality in a logical manner. This effort to bundle logic into distinct executables typically leads to the identification of new services.

Run-Time Discovery Iteration: Discover after Service Deployment. Finally, while a service operates in a production environment, additional service discovery opportunities may arise. Here the deployment and configuration of services and their adaptability to a functioning service ecosystem is road-tested. This may call for the optimization of service integration and collaboration

mechanisms, enhancements to service mediation facilities, or even the improvement of a security model. This typically introduces an array of new service responsibilities and the commencement of capabilities that can fill in the gaps that are found after deployment and distribution of services.

SERVICE DISCOVERY APPROACH. The process of service discovery should be guided by organizational best practices and standards. Without a lucid governance direction, the service identification venture may take superfluous turns that typically prolong the service development effort and hinder vital deliverables that are required for service deployment to production. Therefore, a discovery method should be carved out to channel the service identification efforts and encourage practitioners to focus on the keystone solutions. Moreover, this how-to instruction manual should identify the starting point, milestones, and end point of the discovery process. The method of ascertaining new services or employing existing legacy applications to offer a remedy to an organizational concern should also elaborate on the imperative artifacts that are needed for the service discovery process. To learn more about these approaches, refer to the Service Identification Patterns section later in this chapter, and for an elaborated discussion study Chapters 5 through 10.

Exhibit 1.4 illustrates the five different paths for service discovery. These are orientations devised to facilitate efficient service identification and help practitioners to broaden the search for services that offer solution capabilities to business or technical problems. As apparent, the depicted five directions represent distinct *patterns* for service discovery (note the indicated page number that is positioned in parenthesis next to each pattern):

1. **Top Down (71,89).** This discovery approach advocates deriving new services from business processes and their underpinning activities. As an alternative, this method of service identification also calls for ascertaining conceptual services that stem from solution ideas and software product attributes.

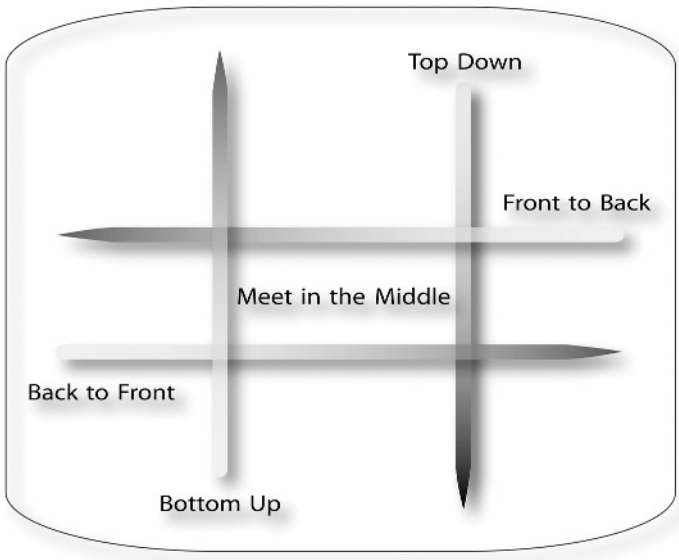

EXHIBIT 1.4 SERVICE-ORIENTED DISCOVERY PATTERNS

2. **Front-to-Back (105).** This service discovery track calls for discovering services by studying user interface deliverables, such as presentation layer logic, content rendering mechanisms, page layouts, and field validation mechanisms.
3. **Back-to-Front (123).** This service identification course takes into consideration data artifacts, such as data schemas, entity diagrams, and data structures.
4. **Bottom-Up (145).** This approach for service discovery capitalizes on existing technologies, architectures, and business processes to derive services.
5. **Meet-in-the-Middle (165).** Finally, this track for service identification shifts the focus to the environment where services operate and the various facilities that link services to each other, such as integration mechanisms, orchestration capabilities, and message exchange platforms.

SERVICE-ORIENTED DISCOVERY AND ANALYSIS PROPOSITION

The service analysis and discovery processes discussed hitherto elaborate on the approach and the road map that each effort is driven by. As the reader may recall, the analysis venture is a study, devised to explain the nature of a problem, assess a proposed solution, and certify services for production. In contrast, the discovery endeavor pinpoints new or existing services that offer capabilities to alleviate an organizational concern. These discovery and analysis initiatives take place in different stages of a service life cycle. Their yielding deliverables are distinct, yet they are interconnected because of the inherent dependency between the two processes. Therefore, the service-oriented discovery and analysis discipline as a whole combines these initiatives and calls for an analysis proposition delivery. This is a chief milestone in the service-oriented life cycle because the yielded services and their hosting environments are the enduring units of analysis for service life cycle stages.

ANALYSIS PROPOSITION. As discussed, the final product of the service-oriented discovery and analysis is an analysis proposition. This is an *offer*, typically delivered to the business and IT organizations for study and approval. Once endorsed, these analysis findings and recommendations are utilized in the various service life cycle stages, such as conceptualization, design and architecture, construction, and operations. So what does the analysis proposition consist of? An analysis proposition does not constitute formal and final design and architecture blueprints; those are provided during the design and architecture service life cycle stage. In contrast, the analysis proposition is merely an examination of service internal and external constructs, an assessment of feasibility, and recommendations for impending service implementations.

Therefore, the chief deliverable is an analysis modeling diagram that illustrates a proposal for a solution in which services, their corresponding consumers, and their hosting environments are identified and modeled for a solution. The term "modeled" pertains to modeling efforts that are conducted by analysts, architects, modelers, developers, and managers employing an analysis modeling language and notation. Furthermore, this proposition characteristically includes *recommended* service internal and external design and architecture, such as service structures, distribution of services, integration of services, and a model for binding contracts. To read more about the modeling language and its notation, refer to the Service-Oriented Discovery and Analysis Modeling section later in this chapter and to Chapters 14 through 22.

The sections that follow depict the three major deliverables of the service-oriented discovery and analysis proposition (illustrated in Exhibit 1.5). These include the ultimate artifacts that the practitioner should consider delivering:

1. Proposing an enhanced solution
2. Proposing an alternative solution
3. Proposing a new solution

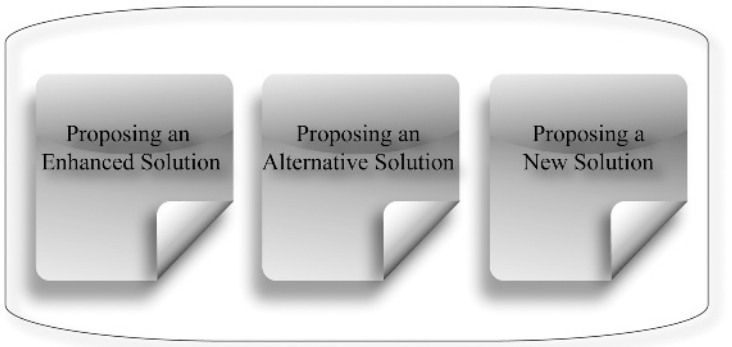

EXHIBIT 1.5 SERVICE-ORIENTED DISCOVERY AND ANALYSIS PROPOSITION DELIVERABLES

PROPOSING AN ENHANCED SOLUTION. Obviously, an analysis proposition that is devised to enhance a solution to a problem must follow an offered remedy that is extended by business or IT personnel during a service life cycle. This pertains to four major service life span stages: inception, design and architecture, construction, and operations.

1. In the project inception phase, the service-oriented discovery and analysis process should focus on evaluating a business or technical remedy, assess its practicality, and introduce a superior solution proposition, if appropriate. The phrase "business or technical solution" mostly pertains to business requirements or technical specifications that embody the problem domain and identify the necessary solution for implementation. From a business perspective, business requirements are fundamental artifacts that the IT organization depends on when constructing services. Technical specifications, however, are affiliated with technological and architectural capabilities of a remedy. Therefore, during the service-oriented discovery and analysis process that takes place in the project inception phase, the practitioner must examine these requirements and offer an analysis proposition if they do not fully meet organizational concerns and do not offer an adequate remedy to a problem.
2. During the design and architecture project stage, the practitioner is commissioned to evaluate design and architecture blueprints, assess their feasibility, and propose an improved scheme. This may include internal service design and external hosting environment.
3. The service construction phase also introduces opportunities for analysis and perfection activities. The devised tangible services and the constructed integrated environment should also be subject for optimization and improvement.
4. When services are deployed, configured, monitored, and managed in a production environment, the opportunities for proposing enhanced service integration and collaboration models are vast. This may include improving service reusability rates, tackling interoperability challenges, and firming up message exchange routes.

PROPOSING AN ALTERNATIVE SOLUTION. As discussed in the previous section, an analysis proposition to enhance an offered remedy can be delivered in different project life cycle stages. In contrast, an analysis proposition can be extended to explore a different avenue, an alternative solution that deviates from the original solution that was introduced in one of the project stages: inception, design and architecture, construction, or operations. The chief reasons for disagreeing with a business or technological remedy and offering an alternative proposition is typically due to misinterpretation of requirements and specifications or simply rejecting a proposed solution approach.

For example, misunderstanding of the problem domain at the project inception phase can trigger an alternate solution that is offered by the analysis proposition. Discrepancies between business requirements and architecture blueprints during the design and architecture project phase can also result in an alternative solution. Finally, inconsistencies between the constructed services and the architecture model may occur during the construction phase. This may result in disagreements about the original development direction and approach that may call for an alternative solution proposition.

PROPOSING A NEW SOLUTION. Occasionally an analysis proposition offers a new solution that has never been proposed before. This occurs not because of misinterpretations of business requirements or noncompliance with design and architecture blueprints. An analysis proposition can be crafted because an organizational concern has not been tackled, requirements were not provided, design and architecture deliverables did not address a problem, or the development team has never been commissioned to construct the required tangible services. The responsibility of a practitioner is then to find these gaps in each of the project phases and conduct analysis and discovery sessions to tackle unaddressed issues and concerns.

DRIVING PRINCIPLES OF SERVICE-ORIENTED DISCOVERY AND ANALYSIS

The service-oriented discovery and analysis process is a repeatable discipline that offers best practices and patterns to enable practitioners identifying and inspecting a service's internal and external design and architecture. From a service-oriented analysis perspective, this venture calls for a continuous investigation and assessment of service feasibility and contribution to a business or a technological problem. This endeavor also takes place during all service life cycle stages, during which the subject for inspection differs in each step of a service's evolution. As we learned in the Service-Oriented Analysis Endeavor section, different artifacts are examined in each analysis iteration. The same applies to the service discovery process and its related iterations, when service identification opportunities vary in each service life cycle stage, as discussed in the Service-Oriented Discovery Endeavor section.

This brings us to the chief principles that should be pursued to guarantee a successful discovery and analysis effort, simplify its processes, and maximize its efficiency. The sections that follow introduce these tenets and elaborate on their contribution to the service discovery and analysis venture.

SERVICE-ORIENTED DISCOVERY AND ANALYSIS TRANSPARENCY MODEL. Transparency is one of most intrinsic tenets of the software development life cycle initiative, and in particular it should apply to the service-oriented discovery and analysis discipline. Transparency is about the justification that practitioners must provide when making important decisions about service design, architecture, construction, deployment, or operations. In other words, every significant activity during any of the service life cycle stages should be rationalized by architects, developers, modelers, analysts, and managers. They must not only clarify the reasons that led to certain business or technical resolutions, but also indicate the cost, effort, and time of a solution implementation. Therefore, the service-oriented discovery and analysis discipline calls for the establishment of the service-oriented analysis transparency model that advocates architectural, technological, business, and operational traceability during the service life span. The term "traceability" pertains to the process of tracking the evolution of a service ecosystem and its business and technological drivers. Exhibit 1.6 illustrates these transparency components:

1. **Architectural traceability.** Architectural traceability is related to decisions that must be reported and justified in terms of design practicality, cost effectiveness, and return on investment. These must be evaluated against organizational best practices, such as software efficiency, reusability, agility, performance, elasticity, loose coupling, and interoperability.

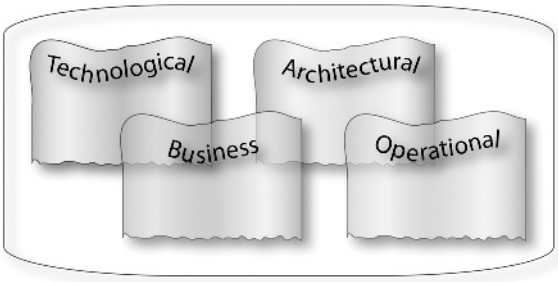

EXHIBIT 1.6 SERVICE-ORIENTED DISCOVERY AND ANALYSIS TRANSPARENCY MODEL

2. **Technological traceability.** This aspect of transparency is about technological decisions that must be reported and evaluated, including selection of commercial off-the-shelf (COTS) products, such as language platforms, messaging facilities, and metadata repositories. Service deployment and configuration mechanisms and service mediation and distribution techniques are other concerns that should be traced.
3. **Business traceability.** Business traceability is about identifying the business value proposition of a service, the contribution of a service hosting environment to organizational concerns, and return on investment when it comes to launching service-oriented projects. It is also about evaluating the feasibility of business strategies, practicality of business requirements, and how effective services satisfy business imperatives.
4. **Operational transparency.** This traceability aspect is related to the assessment of service operations in production. It is about the evaluation of a service's capability to interact with its consumers, exchange messages with its peer services, sustain high consumption rates, uphold high transaction volumes, and offer technological and business continuity with minimum disruption to its operations.

SERVICE MODELING: A VIRTUAL VENTURE. "Virtual modeling of services" is a phrase that depicts the design and architecture of an intangible operating environment, simulated service processes and capabilities, and emulated service relationships with corresponding consumers. This paradigm obviously defines the manipulation of the nonphysical computing landscape that imitates a deployed production environment and the consumers it serves. The major players that participate in the design and architecture venture that takes place in a virtual service world are virtual services, consumers, networks, servers, platforms, repositories, and more. Again, this modeling environment is devised to mimic the real computing world and the transactions that are exchanged.

Therefore, service modeling is a requirement that drives the service-oriented discovery and analysis process. To virtualize a service internal structure, functionality, and its external hosting environment, leverage the proposed modeling language and notation to promote business and technological goals, solve problems, and offer efficient solutions. By modeling a proposed remedy, we not only simulate a tangible deployment landscape and its technologies and architectures, we also contribute to organizational expenditure reduction and minimize project risks. To learn more about modeling techniques, processes, and patterns that can be employed during the service-oriented discovery and analysis process, refer to Chapters 14 through 22.

PATTERNS OF IMPLEMENTATION: BEST PRACTICES AND OUT-OF-THE-BOX SOLUTIONS. During service-oriented discovery and analysis, employ the provided *patterns* for service identification and inspection. These are the best practices and predefined solutions that can reduce development cost, analysis and discovery time, and ultimately organizational expenditure. Use these repeatable templates to:

- Shape the solutions for a project or a business initiative
- Govern the overall service discovery and analysis process
- Model a virtual service environment, mimic service functionality, and simulate service integration

These patterns of design and implementation are vital to service-oriented discovery and analysis because they reduce implementation risks and introduce road-tested, out-of-the-box solutions that have been applied in similar circumstances. They are the guiding pillars of the analysis process and major contributors to shaping a service-oriented analysis proposition. An overview of the service-oriented discovery and analysis patterns is provided later in the Service-Oriented Discovery and Analysis Patterns section later in the chapter.

SERVICE-ORIENTED DISCOVERY AND ANALYSIS MODELING

An analysis proposition is articulated by an analysis proposition modeling diagram that depicts vital design perspectives of a service and its hosting environment. These views include information about *internal* service composition, such as aggregated components, service operations and interfaces, service internal boundaries, service specialties, service capabilities, and more. Service ecosystem *external* aspects, such as deployment, service coupling, contract model, distribution model, and interoperability mechanisms should also be illustrated in an analysis proposition diagram.

Furthermore, a proposed analysis solution must be presented in a formal manner, in which the design of services and their corresponding distributed environment adheres to a prescribed notation that captures relationships between services and associated consumers. This language also is devised to describe the evolution of a service and it transformation stages, starting at its inception, tracing its development phases, and depicting its maturity in production. Chapters 14–22 discuss in details modeling techniques, notations, and modeling operations.

MODELING LANGUAGE WITH TRANSPARENCY CAPABILITIES. Because analysis aspects dominate this exercise, the language that is employed to present the analysis proposition must trace decisions that are made during an analysis process. These questions include: Why should a service be decomposed? Why should a service be retired? Why should two or more services be unified? Why should a service be further abstracted? What is the driving motivation for decoupling two or more services? In which circumstances should a binding contract be established between a service and its corresponding consumers?

Furthermore, as we learned in the previous section, the service-oriented analysis and discovery transparency model also calls for an elaborated analysis proposition that traces business decisions, technological and architectural best practices and implementations, and even events during run-time in production. From a modeling perspective, these may have occurred in the past, be taking place at the present time, or be planned for the future. This analysis *traceability* approach is tuned to the time aspects of a service life cycle. That is, it describes the metamorphosis of a service since its inception, identifies its existing state, or provides an architecture plan for the future.

Exhibit 1.7 illustrates the three service-oriented discovery and analysis modeling methods, each of which presents a different state in the service life cycle: to be, as is, and used to be. These are explained in detail in the sections that follow.

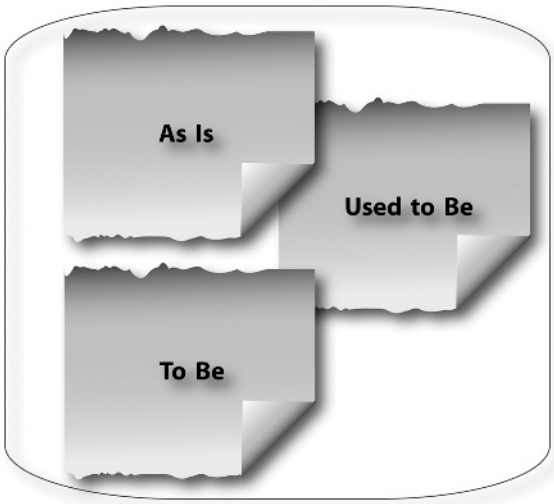

EXHIBIT 1.7 SERVICE-ORIENTED DISCOVERY AND ANALYSIS MODELING STATES

TO-BE MODELING STATE. Nearly all of today's modeling languages offer design capabilities that reflect a futuristic state of software architecture. These languages illustrate future scenarios; some even introduce a road map for implementation. This modeling approach is focused on how a service and its hosting environment "ought to be." Thus past or current design considerations are disregarded. In other words, since the to-be modeling paradigm does not "remember" past or current service life cycle states, the only documented aspect of the service design is the future or end state of architecture formation.

The service-oriented discovery and analysis modeling venture obviously capitalizes on the to-be view. Future depiction of a solution and implementation is vital to the analysis proposition since it recommends nonexistent avenues for tackling organizational concerns.

USED-TO-BE MODELING STATE. Unlike the to-be modeling state, which is focused just on future service architecture, the used-to-be method looks at the past. Therefore, the analysis proposition diagram depicts the past state of service design, identifies best practices that led to architectural decisions, describes technological preferences that were popular earlier in the service life span, or elucidate business strategies that influenced service design. These aspects of analysis offer a great deal of information about a service's evolution and its hosting environment, business incentives, return on investment, architecture strategies, and more.

AS-IS MODELING STATE. The as-is modeling state that is depicted in an analysis proposition diagram illustrates the current state of a service architecture. This is an intermediate condition of the design that is clearly different from past and future planning. Thus the diagram represents a service's transformation between a selected point in the past and current implementation. This feature enables analysts, architects, managers, developers, and modelers to understand the fundamental changes that a service and its operating environment have undergone.

Such valuable information can be leveraged for learning from mistakes and errors that have occurred due to vast reasons, such as misconstrued business requirements, miscalculated

capacity and consumption planning, inadequate computer resources assignment that influenced service performance, and/or unsuitable design and architecture implementations. From a return on investment perspective, the as-is modeling approach can identify the actual cost of a project that has been launched in the past. This vital information can shape future service-oriented development initiatives and serve as an essential use case study to improve future projects.

SERVICE-ORIENTED DISCOVERY AND ANALYSIS PATTERNS

By and large, patterns are intrinsic to software development disciplines since they simply offer solutions. These are proven best practices and a recommended "how to" set of remedies that can be applied to resolving problems. They are also conceived of as *repeatable methods* that have been road-tested and applied to rectify similar software development problems.

Design patterns are perhaps the most well-known guiding solutions that originated in modern computing times both to assist with understanding a problem and, most important, to offer avenues of implementation that developers and architects can employ. Moreover, these patterns are positioned to help alleviate challenges on different levels of software implementation: Some are devised to provide improved source code algorithms, others address application architectural issues, and several tackle enterprise architecture solutions.

The service-oriented discovery and analysis patterns introduce a wide range of repeatable solutions that *not only* address design and architecture challenges but also represent a new breed of patterns affiliated with the process of implementation. The processes of service identification and inspection, of discovery, of categorization, and of course of analysis and modeling can be leveraged by a practitioner to devise superior solutions to organizational concerns. These how-to best practices and policies are accompanied by what-not-to-do recommendations, often referred to as anti-patterns. Service-oriented discovery and analysis anti-patterns are also necessary to foster service reuse, consolidation, loose coupling, and business agility.

The five different discovery and analysis pattern categories that are illustrated in Exhibit 1.8 can help carve the analysis proposition that architects, developers, modelers, analysts, and

EXHIBIT 1.8 SERVICE-ORIENTED DISCOVERY AND ANALYSIS PATTERNS

managers are commissioned to deliver. These constitute the recommended approaches for accomplishing the service identification and inspection venture.

DISCOVERY AND ANALYSIS ROAD MAP PATTERNS. Governing best practices and implementation directions are imperative to service identification and inspection processes that take place during a service life cycle. Therefore, the discovery and analysis road map patterns offer orientations, overarching guiding paths[5] for ascertaining new services or engaging legacy software assets for a proposed solution. These are also repeatable analysis procedures that practitioners can employ to inspect a service's internal design or external architecture. In addition, the road map patterns set priorities and define the boundaries for service discovery and analysis activities that are practiced during a project.

The essential challenges that are addressed by the road map patterns typically are conveyed by fundamental governing questions, such as: Where do we begin with the analysis process? How do we start with service identification activities? What are the chief best practices that motivate the service discovery and analysis process? What type of inputs does the service discovery for? The answers to these questions and an elaborated plan for conducting service identification and examination are introduced in Chapters 2 through 4.

SERVICE IDENTIFICATION PATTERNS. As the reader may remember, a model for service-oriented discovery is revealed earlier in the Service-Oriented Discovery Approach section that elaborates on the five major tracks for discovering services: Top-Down (71,89), Bottom-Up (145), Front-to-Back (105), Back-to-Front (123), and Meet-in-the-Middle (165). These service identification directions are established in Chapters 5 through 10 as formal patterns, methods of discovery that can be employed to ascertain new services or engage legacy software in a proposed solution. Furthermore, a remedy to an organizational concern typically includes services that are being developed and maintained in different life cycle stages, such as concepts, services that are being designed and constructed, or even deployed services to production. To assist with the collection of services, the introduced discovery patterns provide distinct paths for service identification. Each of these tracks guides practitioners how to inspect a diversity of project artifacts and derive services for a business or technological mission.

The patterns for service discovery also introduce methods and processes for implementation. These can be employed to maximize identification opportunities and leverage a wide range of input deliverables that can be collected to broaden the discovery venture. Consider the chief artifacts that are utilized to conduct a successful service derivation and collection initiative:

- **Business processes.** Documented and undocumented business processes, business requirements, business modeling diagrams, and business analysis documents
- **Product specifications.** An assortment of product descriptions that identify software product attributes and features
- **User interface articles.** Screen design artifacts, storyboards, page layout design requirements, and user interface scripts
- **Data and content artifacts**. Data structure specifications, data schemas, and data concepts
- **Existing technologies.** Existing technologies and technologies, such as empowering platforms, middleware, existing services, and more

SERVICE CATEGORIZATION PATTERNS. The service categorization effort is about classifying discovered services by employing three major perspectives. The first is related to service contribution to different business and technological disciplines (also referred to as contextual contribution). This venture is chiefly about grouping services by their common specialties and the type of offerings extended to consumers. For example, a service can be affiliated with business processes that execute an equity trading order. A car insurance premium calculator is another function that can be designated

to a service. From a technical perspective, content download capability or audit trail functionality can be assigned to utility services that fall under the technical category.

Second, services can be categorized by their origin, or the sources that a service stems from. For example, a service may originate from a concept found in an organizational software portfolio or even spotted in a production environment.

Third, from a structural perspective, a service can be classified by its internal formation type or external distribution method.

These categorization approaches call for the institution of patterns that can help practitioners classify a service by various aspects that define its identity. As discussed, these may be structural considerations, contextual affiliations, and even origins. To learn more about these provided service categorization patterns, refer to Chapters 11 through 13. The practitioner should leverage these methods of classification to establish an organizational taxonomy that promotes service reuse, consolidation, reduction of expenditure, and nimbleness.

CONTEXTUAL ANALYSIS AND MODELING PATTERNS. Discussed in detail in Chapters 14 through 17, the contextual analysis and modeling patterns embody guiding principles for modeling services to offer solutions based on their semantic affiliation. The term "semantic" describes a service's association with a business or technical specialty, a field of expertise that can alleviate an organizational concern. Accounting, home insurance, and trading are examples of specialties that can drive the contextual modeling process. Therefore, the contextual modeling approach enables the manipulation of a service's functionality scope and the magnitude of its offerings in a hosting computing environment. This approach typically affects the extent of a service's capabilities and ultimately affects the dimension of its consumers' base.

To attain these goals, the contextual analysis process offers patterns to meet four different objectives for a project:

1. **Contextual generalization patterns.** A group of patterns that increase the abstraction level of a service to widen the boundary of its capabilities
2. **Contextual specification patterns.** Patterns employed to reduce service functionality and abstraction level
3. **Contextual expansion patterns.** An assortment of contextual patterns that enable the expansion of a distributed service environment
4. **Contextual contraction patterns.** A collection of patterns that assists with the contraction of a distributed service landscape and reduction of architecture scope

STRUCTURAL ANALYSIS AND MODELING PATTERNS. In contrast to the contextual analysis and modeling patterns, which merely focus on semantic affiliations and manipulation of service functionality and capabilities, the structural patterns are devised to shape internal and external service formations to provide a superior solution. The term "structure" clearly pertains to two different aspects of service design: internal and external. The former corresponds to an internal service construct. This includes aggregation of service components, granularity level of a service, and even establishment of contracts between a service's constituents with their corresponding external consumers. The patterns that help shape service external structures, however, address different concerns: service distribution, service mediation, service federation, service integration, interoperability, and more. Consider the four different structural analysis and modeling patterns that are discussed in detail in Chapters 18 through 22:

1. **Structural generalization patterns.** Patterns that help widen a service's internal structure and as a result increase the boundary of its capabilities
2. **Structural specification patterns.** A group of patterns that trim down a service internal formation to reduce the overall scope of its operations

3. **Structural expansion patterns.** A collection of structural patterns that expand a distributed service environment and widen a deployed architecture scope
4. **Structural contraction patterns.** An assortment of structural patterns that facilitate the reduction of a distributed service landscape and contraction of architecture

SUMMARY

- The service-oriented discovery and analysis discipline consists of two major processes: service identification and analysis.
- The service-oriented analysis process is repeated in four major analysis iterations: inception, assessment, verification, and authentication.
- The discovery process calls for service identification in five different iterations: concept discovery, post-proposition, design time, construction time, and run-time.
- The analysis proposition is the final artifact of the service-oriented discovery and analysis.
- There are three major service-oriented discovery and analysis principles that should be embraced by practitioners: transparency, virtual modeling, and adoption of patterns.
- The service-oriented discovery and analysis modeling language support three major states: as-is, used-to-be, and to-be.
- The service-oriented discovery and analysis offers five pattern categories: discovery and analysis road map patterns, service identification patterns, service categorization patterns, contextual analysis and modeling patterns, and structural analysis and modeling patterns.

Notes

1. The service-oriented discovery and analysis discipline is a part of the service-oriented modeling framework (SOMF) that is featured in Michael Bell, *Service-Oriented Modeling: Service Analysis, Design, and Architecture* (Hoboken, NJ: John Wiley & Sons, 2008).
2. The service-oriented analysis process can be embedded in any existing organizational development and operations life cycle stage
3. The service-oriented analysis governing process discussed is inspired by Richard Veryard, an author on business modeling and SOA governance. Refer to Chapter 2 for a detailed discussion of his contribution.
4. The service-oriented discovery process can be incorporated in any organization development and operations life stage.
5. Start the service-oriented discovery and analysis process by studying the road map patterns offered in Chapters 2 through 4. Carving a personal and an organizational strategy should then follow.