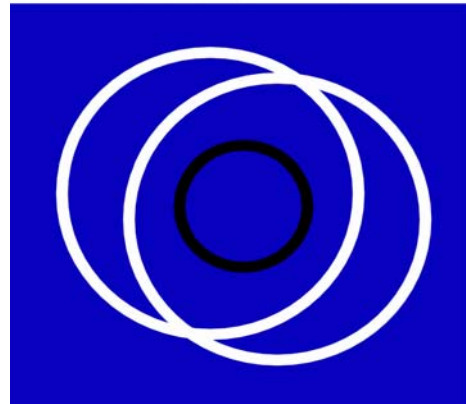
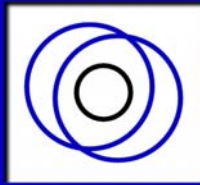


SOA Systems





SOA Methodology (r. 2.0)

The following document contains revised versions of the Service-Oriented Analysis and Service Modeling processes that are part of release 2.0 of the SOA Systems Mainstream SOA Methodology. These and other processes in the methodology are regularly enhanced to reflect industry developments and trends.

For more information, visit: <http://www.soamethodology.com>

Note that for the convenience of readers, this document has been structured so that the sections provided replace the corresponding sections in "Service-Oriented Architecture: Concepts, Technology, and Design" (by Thomas Erl, ISBN: 0131858580, Prentice Hall).

For more information, visit: <http://www.soabooks.com>

11.1 Introduction to service-oriented analysis

The process of determining how business automation requirements can be represented through service-orientation is the domain of the service-oriented analysis.

11.1.1 Objectives of service-oriented analysis

Different organizations have adopted different approaches to analyzing business automation problems and developing corresponding solutions. Years of effort and documentation will often have been invested into well-established processes and modeling deliverables.

The process described in this section is not intended to supplant existing procedures. Instead, it proposes a set of supplementary steps that help shape the organization of automation logic in preparation for the service-oriented design processes. These steps raise issues that tie the use of service models, service-orientation principles, and other key considerations into a preliminary definition of services.

It is important to note that the process provided here is generic and high-level. It establishes a starting point from which an organization is expected to customize a variation of service-oriented analysis that is most suitable to its goals in relation to an SOA transition and most compatible with its existing approaches to business analysis.

The primary questions addressed during this phase are:

- What services need to be built?
- What logic should be encapsulated by each service?

The extent to which these questions are answered is directly related to the amount of effort invested in the analysis. Many



SOA Methodology (r. 2.0)

of the issues we discussed in the past two chapters can be part of this stage.

The overall goals of performing a service-oriented analysis are as follows:

- Define a preliminary set of service operation candidates.
- Group service operation candidates into logical contexts. These contexts represent service candidates.
- Define preliminary service boundaries.
- Identify encapsulated logic with reuse potential.
- Ensure that the context of encapsulated logic is appropriate for its intended use.
- Identify preliminary issues that may challenge required service autonomy.
- Define any known preliminary composition models.

11.1.2 Service-oriented analysis and the enterprise service model

Service-oriented analysis can be applied at different levels, depending on which of the SOA delivery strategies are used to produce services. As explained in the previous chapter, the chosen strategy will determine whether some form of enterprise service model has been established and to what extent its detail has been defined.

It is technically possible to follow a strict top-down approach during which service-oriented analysis is iteratively applied to the enterprise service model to such an extent that all possible services have been defined with the required detail to allow service contracts for any service to be derived. However, top-down SOA efforts frequently succeed in only producing a high-level enterprise service model that establishes standardized service models, service layers, and a set of proposed service contexts, without a significantly detailed definition of what functionality is actually encapsulated by each service.

It is with this perspective of an enterprise SOA that the service-oriented analysis process described here is commonly applied. Using whatever has been established within the enterprise service model as its guide, this process addresses the definition of preliminary services within a predefined scope.

11.1.3 The service-oriented analysis process

The service-oriented analysis process is a sub-process of the overall SOA delivery lifecycle. The steps shown in Figure 11.1 represent common tasks associated with this phase and are described further in the following sections.

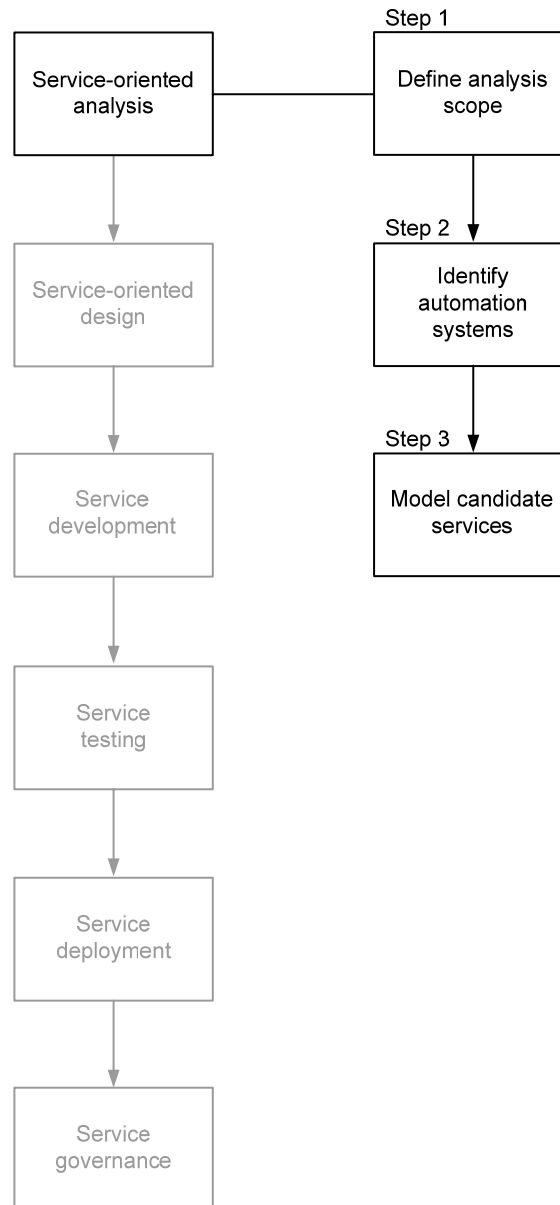


Figure 11.1 – A high-level service-oriented analysis process.

Note that Steps 1 and 2 are essentially information gathering tasks that are carried out in preparation for the modeling process described in Step 3.



SOA Methodology (r. 2.0)

Step 1: Define the scope of the analysis

It's always a good idea to clearly determine the scope of automation logic before you actually get into the delivery lifecycle. With traditional development projects, scope was often tied to the business process requiring automation or to some extension or enhancement to an existing solution (which usually also tied back to a change in the business process).

Service delivery projects can also have a scope that is defined by a unit of business process logic. However, it is not uncommon to only deliver a single process-agnostic service that will establish itself as a part of the overall service portfolio for use by future automated business processes. In this case, the scope is not as much a specific business task as it is the delivery of a series of tasks (operations) associated with a generic service context.

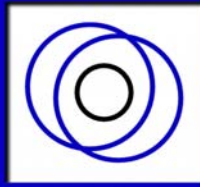
For example, a larger business process that will involve the composition of multiple services will require a scope that involves the definition of several new services and/or the involvement of several existing services. The parent process logic itself may or may not end up residing in a service (depending on the service layers in use).

The delivery of an entity-centric business service, though, will introduce a series of processes, each tied to a generic task that represents a piece of functionality the service is required to encapsulate. Even though only a single service is being delivered, the scope may still be substantial, as each operation of the service can ultimately contain process logic that requires the composition of further services.

To properly establish the scope of the analysis requires that the business requirements we are intending to fulfill be fully defined and mature so that one or more high-level automation processes can be clearly documented with sufficient detail. This business process documentation will be used as the starting point of the service modeling process described in Step 3.

Step 2: Identify existing automation systems

Existing application logic that is already, to whatever extent, automating any of the requirements identified in Step 1 needs to be identified. While a service-oriented analysis will not determine how exactly services will encapsulate or replace legacy application logic, it does assist us in scoping the potential systems affected.



SOA Methodology (r. 2.0)

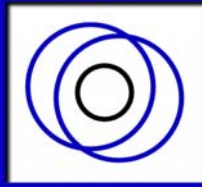
The details of how services relate to existing systems are ironed out in the service-oriented design phase. For now, this information will be used to help identify some of the more immediately recognizable constraints that can affect the definition of service candidates during the service modeling process described in Step 3.

Note that this step is more geared to supporting the modeling efforts of larger scaled service-oriented solutions. An understanding of affected legacy environments is still useful when modeling a smaller amount of services, but a great deal of research effort would generally not be required in this case.

Step 3: Model candidate services

A service-oriented analysis introduces the concept of service modeling—a process by which service operation candidates are identified and then grouped into a logical context. These groups eventually take shape as service candidates are then further assembled into a tentative composite model representing the combined logic of the planned service(s).

This process is explained in detail in the *Service modeling (a step-by-step process)* section of Chapter 12. Its purpose is to produce an ideal representation of business automation logic through service-orientation. The service candidates defined as a result of service modeling eventually become input for the service-oriented design processes described in Chapter 15.



SOA
Methodology
(r. 2.0)

12.1 Service modeling (a step-by-step process)

A service modeling process is essentially an exercise in organizing the information we gathered in Steps 1 and 2 of the parent service-oriented analysis process. Sources of the information required can be diverse, ranging from various existing business model documents to verbally interviewing key personnel that may have the required knowledge of a relevant business area.

This process can therefore be structured in many different ways. As with the parent service-oriented analysis process, the steps described in this section are best considered a starting point from which you can design your own process to fit within your organization's existing business analysis conventions and procedures.

12.1.1 “Services” versus “Service Candidates”

Before we begin, let's first introduce an important modeling term: *candidate*. The primary goal of the service-oriented analysis stage is to figure out how services are ideally defined on a logical level, so that we know what to design and build in subsequent project phases.

It is therefore helpful to continually remind ourselves that we are not actually implementing a design at this stage. We are only performing an analysis that results in a proposed separation of logic used as input for consideration during the service-oriented design phase. In other words, we are producing abstract candidates that may or may not be realized as part of the eventual concrete design.

The reason this distinction is so relevant is because once our candidates are submitted to the design process, they are subjected to the realities of the technical architecture in which they are expected to reside. Once constraints, requirements, and limitations specific to the implementation environment are factored in, the end design of a service may be a significant departure from the corresponding original candidate.

So, at this stage, we do not produce services; we create *service candidates*. Similarly, we do not define service operations; we propose *service operation candidates*. Finally, service candidates and service operation candidates are the end-result of a process called *service modeling*.

12.1.2 Process description

Up next is a series of twelve steps that comprise a proposed service modeling process, as illustrated in Figure 12.1.

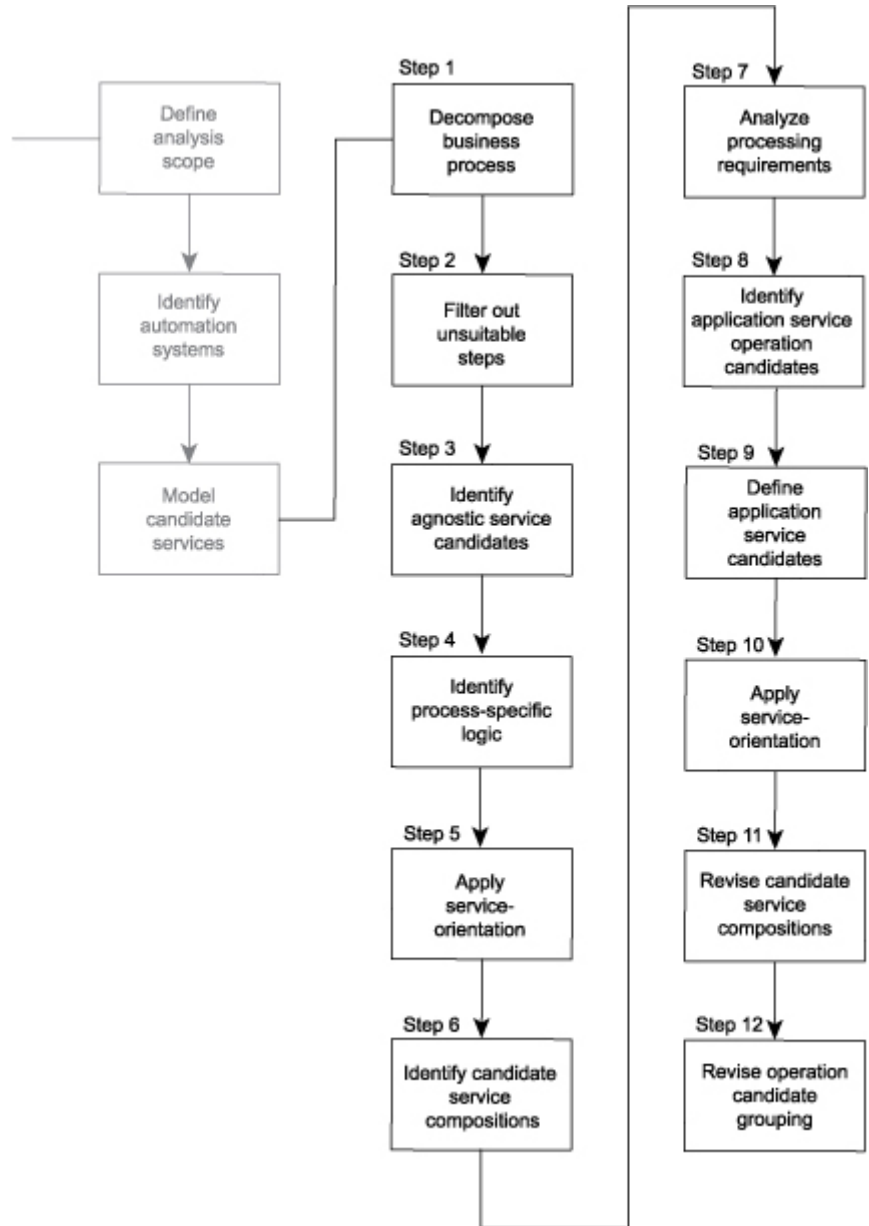


Figure 12.1- The service modeling process.



SOA Methodology (r. 2.0)

Step 1: Decompose the business process

Take a documented business process and break it down into a series of steps. It is important that the process workflow logic be decomposed into the most granular representation of processing steps, which may differ from the level of granularity at which the process steps were originally documented. (The *Classifying service model logic* section at the end of this chapter introduces new terms that help distinguish the scope of individual process steps.)

Step 2: Filter out steps not suitable for service encapsulation

Some steps within a business process can be easily identified as not belonging to the potential logic that should be encapsulated by a service candidate.

Examples include manual process steps that cannot or should not be automated and process steps performed by existing legacy logic for which service candidate encapsulation is not an option. (The latter type of logic will often be recognizable with the information gathered as part of *Step 2: Identify existing automation systems* from the parent service-oriented analysis process.)

By filtering out these parts we are left with the processing steps most relevant to our service modeling process. The steps we remove at this stage are simply placed on a separate list and revisited when we define how the parent business process logic will need to be implemented during the service-oriented design stage.

NOTE

While going looking through the individual process steps you are encouraged to annotate them with initial thoughts about how they could potentially be grouped and utilized.

Step 3: Identify agnostic service candidates

Using any available enterprise models from the top-down analysis (see Chapter 10, *The top-down strategy* section) as a starting point, identify or create a set of logical contexts that are agnostic to the business process but still relevant to the business process steps defined in Steps 1 and 2. Each context represents an agnostic service candidate. Group the steps for potential encapsulation within these service candidates. Each step represents a potential service operation candidate.



SOA Methodology (r. 2.0)

If the scope of the analysis is the delivery of a single agnostic service, then simply group together all actions that appear to belong to this service's context. Next, look to the enterprise service model (or corresponding documentation of the service inventory) for agnostic services (existing or new) that the remaining actions may belong to.

It is important that you do not concern yourself with how many steps belong to each group. The primary purpose of this exercise is to establish an initial set of contexts.

Step 4: Identify process-specific logic

The actions that remain after completing Step 4 should represent logic that is specific to the task or business process. Common examples include process-specific business rules, conditional logic, exception logic, and logic that dictates the sequence in which other actions should be carried out.

Note that these forms of process logic may or may not be represented accurately by a step description. For example, some processing step descriptions consist of a condition and an action (if condition x occurs, then perform action y). In this case, focus on the action.

Depending on which service layers have been standardized, this type of logic will likely comprise a context for a business process definition as part of an orchestration environment or as part of one or more task-centric business services. In the latter case, be sure to clearly define the context of the task-centric business services by labeling them according to the scope of their tasks. More often than not, only one task-centric business service will be defined at this stage. This service will then be positioned as the parent controller for the service composition.

If you are only delivering a single agnostic service, then this step should not be required. Parent processes within the scope of a single service are typically implemented within one or more operations of the service. If actions should remain after completing Step 3, then revisit the wording of each action and see how it could or should be associated with the context of the service. If, for some reason, it can't, then that may indicate the need for a new service or perhaps a reduction in the original analysis scope.

Also note that regardless of where the actions will ultimately reside, it is likely that some of the identified workflow logic will eventually be dropped. This is because



SOA Methodology (r. 2.0)

not all processing steps necessarily become service operations.

Step 5: Apply service-orientation principles

So far we have just grouped processing steps derived from an existing business process. To make our service candidates truly worthy of an SOA, we must take a closer look at the underlying logic of each proposed service operation candidate. This step gives us a chance to make adjustments and apply key service-orientation principles.

Two principles that have special significance during the service modeling stage are reusability and autonomy. (The other principles are addressed in the service-oriented design process, at which point reuse and autonomy are also revisited.)

For example, it is encouraged that application and entity-centric business service candidates be equipped with additional operation candidates that facilitate future reuse. Therefore, the scope of this step can be expanded to include an analysis of additional service operation candidates not required by the current business process, but added to round out agnostic services with a more complete set of reusable operations. For business services especially, we can take advantage of the involvement of business analysts who are expected to participate in the service modeling process (but not in the subsequent service-oriented design process).

NOTE

It is recommended that newly added operation candidates be somehow tagged or labeled differently so that should the scope of the service need to be scaled down in the design phase, those operations that are required for the automation of the business process are easily distinguished from those that aren't.

With regards to autonomy, it may be possible to get a sense of the underlying processing environments for the identified service operation candidates. This is where technology architects can provide a foreknowledge of any potential legacy system functionality the operations may be required to encapsulate (again relating back to Step 2 of the parent service-oriented analysis process). Even though we are focused more on a logical definition of services at this stage, it can be beneficial to take legacy logic encapsulation requirements into account.



NOTE

Service discoverability can also be addressed at this stage. Although it may not directly influence the structure of a service candidate, it can be beneficial to take advantage of the involvement of business analysts and other subject matter experts. These professionals can contribute clear and concise descriptions that express the purpose and capabilities of a service. These statements can then be incorporated into the physical service contract during the service-oriented design phase (at which point business analysts may no longer be actively involved).

Step 6: Identify candidate service compositions

Here we need to document a set of the most common scenarios that can take place within the boundaries of the business process. For each scenario, follow the required processing steps as they exist now. Don't be too concerned with the specific type of data that is being processed and passed between service candidates. Focus on the flow of action across service candidates and through service operation candidates. Also, ensure that as part of your chosen scenarios you include failure conditions that involve exception handling logic.

By identifying potential service compositions, this exercise can help determine how appropriate the grouping of service operation candidates is, while also demonstrating the potential relationships between service layers. Additionally, it will often indicate where missing workflow logic or processing steps exists. This may result in the need to define new service candidate operations and perhaps even new service candidates.

It is advisable to revisit the original grouping of service operation candidates and make any necessary refinements as part of completing this step.

Step 7: Analyze application processing requirements

Because our focus so far has been on business process logic, there is a natural tendency to emphasize the definition of business service candidates over application service candidates.

This step requires that we more closely study the underlying processing requirements of each service operation candidate to abstract any further processing logic that could potentially be added to the preliminary application services layer. To accomplish this, each operation candidate is required to undergo a mini-analysis.



SOA Methodology (r. 2.0)

Specifically, what needs to be determined is:

- What underlying application logic needs to be executed to process the action described by the operation candidate.
- Whether the required application logic already exists or whether it needs to be newly developed.
- Whether the required application logic spans existing solution boundaries. In other words, is more than one system required to complete this action?

The result of this step will be a list of processing requirements that act as input for Step 8.

This and the next series of steps are especially relevant when the scope of the overall analysis is a larger, complex business process. Note that the information we gathered during Step 2 of the parent service-oriented analysis process will be referenced again during these steps.

Step 8: Identify application service operation candidates

What Step 7 essentially leads us to do is look for an opportunity to identify more steps within our overall business process that qualify as service operation candidates for application services. This can be accomplished by breaking down each application logic processing requirement into one or more processing steps.

It is important to be explicit about how these steps are labeled so that they reference the function they are performing. Because our focus at this stage is the definition of application services, we want to avoid business-specific references. For example, when labeling the service operation candidates, you would ideally not make reference to the business process step for which this function is being identified.

Step 9: Define application service candidates

The processing steps now need to be grouped according to a logical context. Unlike with business services, application services don't have the benefit of a context predefined by existing business models and processes. The primary context therefore is *processing-centric*. This means that we are looking for a logical relationship derived from commonality between the processing functionality represented by the service operation candidates.

For example, this relationship can be based on an association with a specific legacy system, an association



SOA Methodology (r. 2.0)

with one or more solution components, or (most commonly) a logical grouping according to the nature of functionality to be encapsulated by each operation candidate.

As part of this step, the current service inventory needs to be reviewed to ensure that no existing services have been defined that may already relate to any newly proposed candidate contexts.

Step 10: Apply service-orientation principles

This is essentially a repeat of Step 5 but applied specifically to the new application service candidates we have defined as a result of completing steps 7 to 9. The emphasis on reuse and autonomy is just as important when delivering application services as they are for business services.

Autonomy plays a large role in determining the functional scope of each application service operation candidate (which, in turn, influences the potential granularity of the operation candidate).

Once a suitable processing context is established for the operation candidates identified so far, it may lead to opportunities to add new operation candidates or make existing ones more generic, all in support of increasing the reuse potential of the overall service candidate.

Various other issues are factored in once service candidates are subjected to the service-oriented design process. For now, the identified service and operation candidates establish a preliminary application service layer.

Step 11: Revise candidate service compositions

Revisit the original scenarios you identified in Step 6 and run through them again. Only, this time, incorporate the new application service and operation candidates as well. This will result in the mapping of elaborate activities that bring to life the expanded service compositions. Be sure to keep track of how business service candidates can map to underlying application service candidates during this exercise.

Step 12: Revise service operation candidate grouping

Going through the motions of mapping the activity scenarios from Step 11 will usually result in refinements and changes to the grouping and definition of the service operation candidates that have been defined so far.