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.

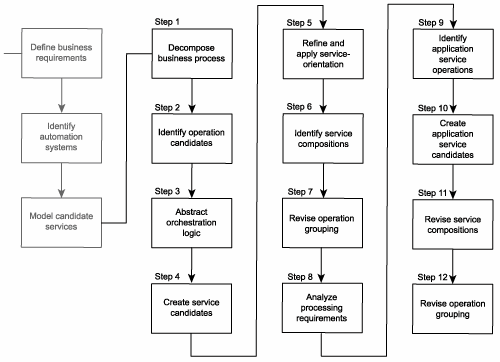
The primary goal of the service-oriented analysis stage is to figure out what it is we need to later design and build in subsequent project phases. It is therefore helpful to continually remind ourselves that we are not actually implementing a design at this stage. We are only performing an analysis 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.

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 12 steps that comprise a proposed service modeling process (Figure 12.1). Specifically, this particular process provides steps for the modeling of an SOA consisting of application, business, and orchestration service layers.

Figure 12.1. A sample service modeling process.

Step 1: Decompose the business process

Take the documented business process and break it down into a series of granular process steps. It is important that a process's 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: Identify business service operation candidates

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.
* Process steps performed by existing legacy logic for which service candidate encapsulation is not an option.

By filtering out these parts we are left with the processing steps most relevant to our service modeling process.

Step 3: Abstract orchestration logic

If you have decided to build an orchestration layer as part of your SOA, then you should identify the parts of the processing logic that this layer would potentially abstract. (If you are not incorporating an orchestration service layer, then skip this step.)

Potential types of logic suitable for this layer include:

* business rules
* conditional logic
* exception logic
* sequence logic

Note that these forms of orchestration 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, only remove the condition and leave the action.

Also note that some of the identified workflow logic likely will be dropped eventually. This is because not all processing steps necessarily become service operations. Remember that at this point, we are only creating candidates. When we enter the service-oriented design phase, practical considerations come into play. This may result in the need to remove some of the candidate operations, which would also require that corresponding workflow logic be removed from the orchestration layer as well.

Step 4: Create business service candidates

Review the processing steps that remain and determine one or more logical contexts with which these steps can be grouped. Each context represents a service candidate. The contexts you end up with will depend on the types of business services you have chosen to build.

For example, task-centric business services will require a context specific to the process, while entity-centric business services will introduce the need to group processing steps according to their relation to previously defined entities. (As stated in the Deriving business services section in Chapter 11, an SOA also can consist of a combination of these business service types.)

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 the required set of contexts.

Also it is encouraged that 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 entity services with a complete set of reusable operations.

Step 5: Refine and apply principles of service-orientation

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. This is where the study we performed in the Native Web service support for service-orientation principles section of Chapter 8 becomes useful. As you may recall, we identified the following four key principles as those not intrinsically provided through the use of Web services:

* reusability
* autonomy
* statelessness
* discoverability

Of these four, only the first two are important to us at the service modeling stage. The latter two on this list are addressed in the service-oriented design process. Therefore, our focus in this step is also to ensure that each service operation candidate we identify is potentially reusable and as autonomous as possible.

Step 6: Identify candidate service compositions

Identify 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.

This exercise accomplishes the following:

* It gives you a good idea as to how appropriate the grouping of your process steps is.
* It demonstrates the potential relationship between orchestration and business service layers.
* It identifies potential service compositions.
* It highlights any missing workflow logic or processing steps.

Ensure that as part of your chosen scenarios you include failure conditions that involve exception handling logic. Note also that any service layers you establish at this point are still preliminary and still subject to revisions during the design process

Step 7: Revise business service operation grouping

Based on the results of the composition exercise in Step 6, revisit the grouping of your business process steps and revise the organization of service operation candidates as necessary. It is not unusual to consolidate or create new groups (service candidates) at this point.

Step 8: Analyze application processing requirements

By the end of Step 6, you will have created a business-centric view of your services layer. This view could very well consist of both application and business service candidates, but the focus so far has been on representing business process logic.

**This next series of steps is optional and more suited for complex business processes and larger service-oriented environments.** It requires that you more closely study the underlying processing requirements of all service candidates to abstract any further technology-centric service candidates from this view that will complete a preliminary application services layer. To accomplish this, each processing step identified so far is required to undergo a mini-analysis.

Specifically, what you need to determine 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 application boundaries. In other words, is more than one system required to complete this action?

Step 9: Identify application service operation candidates

Break down each application logic processing requirement into a series of steps. Be explicit about how you label these steps so that they reference the function they are performing. Ideally, you would not reference the business process step for which this function is being identified.

Step 10: Create application service candidates

Group these processing steps according to a predefined context. With application service candidates, the primary context is a logical relationship between operation candidates. This relationship can be based on any number of factors, including:

* association with a specific legacy system
* association with one or more solution components
* logical grouping according to type of function

Various other issues are factored in once service candidates are subjected to the service-oriented design process. For now, this grouping establishes a preliminary application service layer.

Step 11: Revise candidate service compositions

Revisit the original scenarios you identified in Step 5 and run through them again. Only, this time, incorporate the new application service candidates as well. This will result in the mapping of elaborate activities that bring to life expanded service compositions. Be sure to keep track of how business service candidates map to underlying application service candidates during this exercise.

Step 12: Revise application service operation grouping

Going through the motions of mapping the activity scenarios from Step 11 usually will result in changes to the grouping and definition of application service operation candidates. It will also likely point out any omissions in application-level processing steps, resulting in the addition of new service operation candidates and perhaps even new service candidates.