# **Introduction to service-oriented analysis**

Def: The process of determining how business automation requirements can be represented through service-orientation

**Objectives of service-oriented analysis**

The primary questions addressed during this phase are:

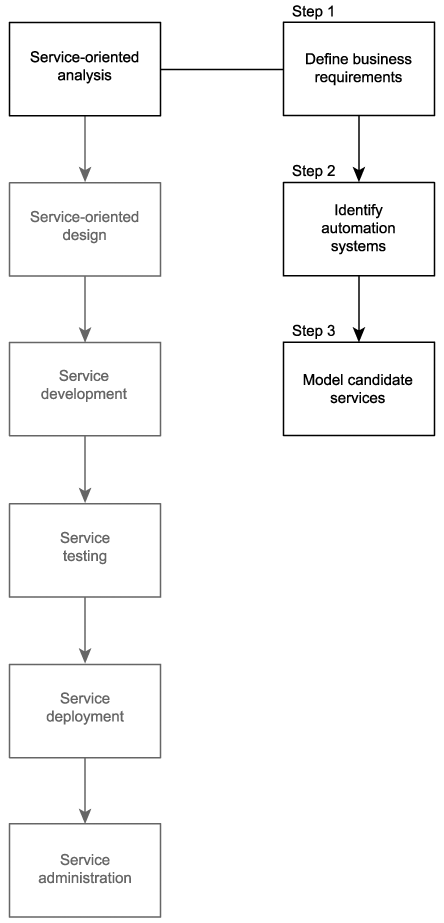
1. What services need to be built?
2. What logic should be encapsulated by each service?

**Goals of performing a service-oriented analysis**

1. Define a preliminary set of service operation candidates.
2. Group service operation candidates into logical contexts. These contexts represent service candidates.
3. Define preliminary service boundaries so that they do not overlap with any existing or planned services.
4. Identify encapsulated logic with reuse potential.
5. Ensure that the context of encapsulated logic is appropriate for its intended use.
6. Define any known preliminary composition models.

**A high-level service-oriented analysis process**

The service-oriented analysis process is a sub-process of the overall SOA delivery lifecycle.



# **Business-centric SOA**

The advent of Web services has publicized the importance of an open communications framework. As a result, most IT professionals are at a stage where they acknowledge the relevance of Web services technologies.

As we've established previously, the majority of Web services currently being built are, more or less, a mixture of application and business services. These types of hybrid services are attractive because, with minimal effort, they fulfill immediate requirements with clear ROI benefits. The proliferation of hybrid services is the result of the bottom-up approach having become so commonplace. They provide an immediate means for all past forms of application architecture to take part in the open Web services communications framework.

Business services, on the other hand, often need some justification. Many still resistor are even unaware ofthe benefits of introducing service-oriented principles into the domain of business analysis. It is easy to ignore service-oriented business modeling and simply focus on service-orientation as it applies to technology and technical architecture. The common rationale for this approach is that whatever business processes need to be automated can be divided up into physical Web services as required.

Many of the characteristics of contemporary SOA we listed in Chapter 3 can still be attained without the use of business services. Though it may appear that you are saving yourself a load of work by taking this path, the apparent benefits are superficial. There are very good reasons for taking the time to model and build business services. In this section we list a series of benefits for incorporating service-orientation into the business process level.

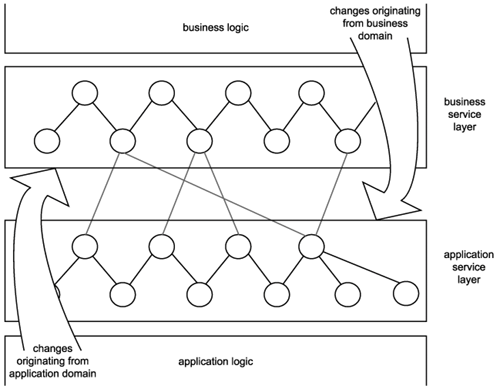
11.2.1. Business services build agility into business models

Service-orientation brings to business process models a structure that can significantly improve the flexibility and agility with which processes can be remodeled in response to changes. When properly designed, business services can establish a highly responsive information technology environment; responsive in that changes in an organization's business areas can be efficiently accommodated through re-composition of both a business process and its supporting technology architecture (as expressed by the application services layer).

As business-driven as SOAs are, there are often real world restrictions (infrastructure, security constraints, budget constraints) that require the technology side to push back. This can shift the burden of adaptation over to the business process models. This type of agility requirement can be met by the business service layer, as it allows business services to adjust to requirement changes originating from an organization's technical environments.

In other words, applying service layer abstraction to both business and technology ends establishes the potential for an enterprise to achieve a form of two-way agility (Figure 11.2).

Figure 11.2. Changes originating in the business logic domain are accommodated by the application logic domain and vice versa.



11.2.2. Business services prepare a process for orchestration

Whether or not you will be moving to an orchestration-based service-oriented environment anytime soon, it is becoming increasingly important to be ready for this transition. Orchestration brings with it concepts that, when implemented, lie at the core of SOA. Therefore, modeling current processes so that they eventually can be more easily migrated to an orchestration-driven environment is recommended.

11.2.3. Business services enable reuse

The creation of a business service layer promotes reuse in both business and application services, as follows:

* By modeling business logic as distinct services with explicit boundaries, business process-level reuse can be achieved. Atomic sub-process logic or even entire processes can be reused as part of other process logic or as part of a compound process (which translates into a service composition in its own right).
* By taking the time to properly align business models with business service representation, the resulting business service layer ends up freeing the entire application service layer from assuming task or activity-specific processing functions. This allows application services to be positioned as and to evolve into pure, reusable utility services that facilitate business services across solution boundaries.

11.2.4. Only business services can realize the service-oriented enterprise

Business service modeling marries the principles of service-orientation with an organization's business model. The resulting perspective can clearly express how services relate to and embody the fulfillment of business requirements.

Applying business services forces an organization to view and reinterpret business knowledge in a service-oriented manner. Altering the perspective of how business processes can be structured, partitioned, and modeled is an essential step to achieving an environment in which service-orientation is standardized, successfully consistent, and naturally commonplace.

Though the business services layer may accurately represent a corporate business model upon implementation, it will become outdated once new and revised business requirements emerge. As long as it is kept in relative alignment with the current state of business models, it will continue to serve as a valuable view of the enterprisevaluable because it does not exist in abstract but in an implemented and operational form.

|  |
| --- |
| Case Study  As mentioned earlier, RailCo has decided to structure its SOA around a services layer consisting of separate business and application sub-layers. Given that it does not have extensive business requirements, these layers will not contain many services. However, the important aspect of this initial project is that it will define an architecture that accommodates additional services and solutions.  As part of its analysis, RailCo has identified the following benefits of investing in extensible business services:   * Business services can be modeled in such a manner that they perform common functions for non-specific service requestors. This will allow RailCo to fulfill its first requirement of making its services available for online transactions with multiple customers. * A separation of business and application logic will allow RailCo to wrap its legacy accounting system in one or more services. This will allow it to establish an application services layer initially consisting of an application service that will be reusable to task-specific business services. * RailCo's second requirement also can be fulfilled by adding a service to represent the contact management system. Integration can then be achieved by again reusing the service that encapsulates the accounting service.   At this stage, RailCo simply is listing potential benefits that can be met by introducing services it has not yet even modeled. |

| **SUMMARY OF KEY POINTS** |
| --- |
| * Investing in a separate business service layer builds agility into an SOA by abstracting business and application logic domains, allowing them to evolve independently, and adapting to each other's changes. * Business services prepare an SOA for orchestration and promote reuse of both business process and application logic through careful encapsulation and abstraction. * Business services are key to fulfilling an enterprise-wide standardization of service-orientation. |

**Deriving business services**

As much as no industry-standard definition of SOA exists and as much as service-orientation principles have not been globally standardized, there is also no standardized means of modeling business services. As with all other aspects of SOA, there are plenty of opinions, and though many have ideas, few concrete methodologies have emerged. Instead, there are a select set of approaches, some of which have been more accepted than others.

Perhaps there should be no single approach to deriving services. It is not unusual for the business model behind a typical enterprise to have undergone thousands of revisions, shaped through years of adapting to the organization's surrounding business climate. Organizations employ different methodologies, business entity relationships, and vocabularies, resulting in vastly diverging business model structures. Further, there are cultural preferences and vendor platform influences that result in expression of the business models through different sets of modeling tools and languages.

**The bottom line is that every business model is unique. Therefore, up-front analysis cannot be avoided to properly derive business services that best represent an organization as a cohesive business entity.**

11.3.1. Sources from which business services can be derived

The inner workings of any organization, regardless of structure or size, can be decomposed into a collection of business services. This is because a business service simply represents a logical unit of work, and pretty much anything any organization does consists of units of work.

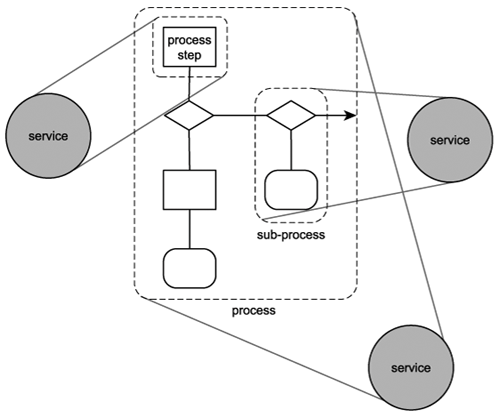
What differs, though, is how organizations structure and document the work they perform. At the beginning of this section we stressed the fact that every corporate environment is unique in the shape and size of its business models and in how it implements and maintains them. It is therefore up to the service-orientation-aware analyst to determine how to best map existing logic to services.

Below are some examples of common business analysis approaches used by many organizations. For each we briefly discuss how services can be derived.

Business Process Management (BPM) models

The advent of BPM has resulted in an industry-wide flurry of process modeling and remodeling activity. Process models have therefore become a central form of business analysis documentation in many organizations. Business services can be derived from process workflow logic.

Figure 11.3. Parts of a process that can be encapsulated by a business service.



*Deriving a business service from a business process requires a thorough knowledge of the underlying workflow logic. This is because defining the scope of the business logic to be represented is a judgment call that can have significant implications when the business service is implemented as part of solution environments; hence, the better the judgment, the better the quality of the service. And, of course, the better quality services you end up with, the better quality your service-oriented environment will be.*

Note

It is worth recalling that even though business logic encapsulation typically is illustrated using services, it is actually the service operations that represent and execute the logic in the service layer. Therefore, it is critical to ensure that when identifying logic suitable for service encapsulation, it be broken down into individual operations. Services are then determined by the proposed grouping of these operations.

|  |
| --- |
| Case Study  Following are descriptions of two existing RailCo processes that are currently being partially automated by its services. In past chapters, we focused solely on service interaction and therefore only described the involvement of RailCo's Invoice Submission and Order Fulfillment Services as part of interaction scenarios with the TLS B2B solution. The descriptions provided here document the steps that comprise internal RailCo processes because these are the processes that eventually will be affected by the introduction of SOA.  First, we'll explore the existing Invoice Submission Process. It consists of a series of steps that describe how invoices are generated specifically for TLS, as follows:   1. Accounting clerk creates and issues an electronic invoice using the legacy accounting system. 2. The save event triggers a custom script that exports an electronic copy of the invoice to a network folder. 3. A custom developed component, which polls this folder at ten-minute intervals, picks up the document and transforms it into an XML document. 4. The invoice XML document is then validated. If it is deemed valid, it is forwarded to the Invoice Submission Service. If validation fails, the document is rejected, and the process ends. 5. Depending on when the last metadata check was performed, the service may issue a Get Metadata request to the TLS B2B solution. 6. If the Get Metadata request is issued and if it determines that no changes were made to the relevant TLS service descriptions, the Invoice Submission Service transmits the invoice document to the TLS B2B solution using the ExactlyOnce delivery assurance. If the Get Metadata request identifies a change to the TLS service descriptions, the invoice is not submitted, and the process ends.   The internal Order Fulfillment Process is similar in that it establishes the same type of relationship between the accounting system and a Web serviceonly this time, the data flow is reversed, as described here:   1. The RailCo Order Fulfillment Service receives a SOAP message from TLS, containing a payload consisting of a TLS purchase order document. 2. The service validates the incoming document. If valid, the document is passed to a custom component. If the TLS PO fails validation, a rejection notification message is sent to TLS, and the process ends. 3. The component has the XML document transformed into a purchase order that conforms to the accounting system's native document format. 4. The PO then is submitted to the accounting system using its import extension. 5. The PO ends up in the work queue of an accounting clerk who then processes the document.   Note that both of these processes are intentionally incomplete. The Invoice Submission Process contains additional steps related to an acknowledgement it receives from TLS upon the successful delivery of the invoice submission. The complete Order Fulfillment Process consists of several additional steps that are required to formulate a response to TLS containing back-order information, if required. The process steps that are provided are those most relevant to our study of how implementing SOA leads to improvements for RailCo.  Note that these two process descriptions form the basis of the examples used to demonstrate the service modeling process in Chapter 12. |

Entity models

Primary entities represent the main business documents and transaction areas of an enterprise. For example, Invoice, Purchase Order, Customer, and Claim are all milestone entities within different types of businesses. Further, organizations model entities according to proprietary rules and business policies. This results in entities having relationships with each other.

**Entity-centric services** (explained shortly) mirror the entity model by containing a set of generic operations that facilitate the different types of functions associated to the processing of the entity. Communication between different entity-centric services also can be governed by constraints relating to the inherent relationship between entities.

Note

These types of services often follow the OOP naming convention where a noun is used to label an object and verbs are used to name methods. Because entity-centric services represent data entities, the services can be named with nouns and the operations with verbs.

|  |
| --- |
| Case Study  RailCo has decided to build services centered around specific tasks and is therefore not using entity models as a source for deriving business services. However, to demonstrate the concept of entities, let's briefly look at what entities exist within the RailCo business areas we've been exploring so far.  Based on the business processes we described earlier, RailCo entities of relevance are:   * Invoice * Purchase Order   Additional entities involved with these processes could include:   * Employee * Order * Back Order * Customer   Here are some examples of how the entities described above could relate to each other:   * A Purchase Order can be related to zero or more Invoices. * A Purchase Order can only be related to one Customer. * An Order can be related to one or more Purchase Orders. * A Back Order can be related to one or more Purchase Orders. * An Invoice can only be related to one Customer. * An Employee can be related to zero or more Invoices, Purchase Orders, Orders, Back Orders, or Customers.   Figure 11.4 illustrates this series of one-to-one and one-to-many relationships.  Figure 11.4. Relationships between RailCo entities (normally these entity symbols would contain attributes as well). |

11.3.2. **Types of derived business services**

Deriving services from the two sources we just identified results in the creation of distinct types of business services.

* 1. Task-centric business services

These are Web services that have been modeled to accommodate a specific business process. Operations are grouped according to their relevance to the execution of a task in support of a process.

Typical examples of task-centric business services are:

* VerifyInvoice
* GetHistoryReport

Each of these services contains operations that relate to a particular task within the context of a process. Task-centric services usually result from modeling exercises that are focused on meeting immediate business requirements. Typical sources include use-case models and BPM process definitions.

While they require less analysis effort to produce, these types of business services have limited reusability potential. Modeling reusable task-centric business services often requires that multiple use-cases and business process models first be analyzed to identify commonality, prior to the actual modeling of the services.

|  |
| --- |
| Case Study  Although they are hybrid (application + business) in design, the following RailCo services follow a task-centric model:   * Invoice Submission Service * Order Fulfillment Service * TLS Subscription Service   Note that the TLS Subscription Service displayed in the preceding list does not actually belong to TLS. It is just a poorly named service produced by RailCo for communication with TLS. |

* 1. Entity-centric business services

Entity-centric business services generally are produced as part of a long-term or on-going analysis effort to align business services with existing corporate business models. Their inherent generic nature makes them highly reusable by numerous business processes. Even though entity-centric business services often are built as part of application development projects centered around a particular business process, they differ from task-centric services in that they do not provide an interface specific to that process. Instead, the source of inspiration for these types of services is entity models.

When compared to task-centric services, entity-centric services significantly increase the agility with which service-oriented processes can be remodeled. This is because task-centric services often are built to help automate one business process and can therefore get tied to that process. When the process logic changes, the context under which the services are used and composed may change as well. This may invalidate the original grouping of service operations and could result in the requirement for a redesign and redevelopment effort.

Entity-centric services do require more up-front analysis, increasing both the cost of each service and the time required to produce it. Additionally, they can be so generic in nature that they are delivered with no concept of business process logic. Their use may therefore become dependent on parent business controllers, such as process services or task-centric controller services. As we established in Chapter 9, building a business service layer consisting of a series of entity-centric services composed by a parent orchestration service layer establishes a desirable SOA, promoting a high degree of agility and accurate business model representation.

|  |
| --- |
| Case Study  Although we have identified logical entities within RailCo, no entity-centric business services currently exist. Having followed the top-down SOA delivery strategy, though, TLS already has a set of entity-centric business services in place (Figure 11.5).  Figure 11.5. Different organizations, different approaches to building services.    These are:   * Accounts Payable Service * Purchase Order Service * Ledger Service * Vendor Profile Service   When utilized within the B2B solution, these services actually do not need to be composed by a separate task-centric or process service. The reason this parent controller layer is not required is because the Accounts Payable and Purchase Order Services are not part of a solution-specific business process. They are simply fulfilling their respective document processing roles for the accounting system, receiving valid invoices and issuing valid purchase orders. Therefore, no further workflow logic is required.  TLS is planning to continue building entity-centric services. But, as decided in Chapter 10, they will deliver future business services according to the agile delivery strategy. RailCo does not have the time or budget to outfit each new business service with additional functionality for future reuse opportunities. No entity-centric services are therefore planned for RailCo. |

11.3.3. Business services and orchestration

The process service, an implementation of orchestration, also can be classified as a form of business service. It is very much "business-centric," as it resides at the top of the service layer hierarchy and is responsible for composing business services according to the rules specified in the orchestration workflow logic.

Details regarding the process service are described in the Orchestration section of Chapter 6, the Orchestration services layer section of Chapter 9, as well as the service-oriented business process design portions of Chapter 16. An orchestration can compose a combination of task-centric and entity-centric business services. **The core business model is represented by the entity-centric services, while business logic-related tasks can be implemented in task-centric services that are designed specifically to supplement the process service.**

Essentially, the use of orchestration establishes the following structure in the services layer:

1. Workflow logic and process-specific business rules are embedded in a process definition. Orchestration composes business services (and possibly application services) according to this definition.
2. Business services compose application services to execute business logic.
3. Application services interact with underlying systems to process the required functions.

**Orchestration abstracts workflow logic, positioning it outside of service boundaries. This increases agility by allowing changes to business rules to be made without affecting business or application services. This is a critical aspect of orchestration, as business process logic is subject to many factors that can result in change. These include human intervention, changes to corporate policies and business rules, and unforeseeable exception conditions.**

|  |
| --- |
| Case Study  TLS has decided that as part of its plans to expand its existing SOA, it will introduce an orchestration service layer. It begins by introducing a process service that abstracts business rules and workflow logic from the business services already in use.  Although this requires a significant modeling effort, it is anticipated that the existing service structures and service interfaces will remain unaffected. After the workflow logic represented by the process service is finalized in the service-oriented design phase, the true impact of the orchestration service layer on the existing business services will be determined.  This example is continued in the Contrasting service modeling approaches (an example) section at the end of Chapter 12. |

| **SUMMARY OF KEY POINTS** |
| --- |
| * Business services are derived from common business model documents, including use-case models, business process definitions, and entity models. * There are two distinct types of business services that are commonly derived: task-centric and entity-centric. Each has pros and cons, and each approaches encapsulation in a different manner. |