Warranty Disclaimer

THIS DOCUMENT AND ALL INFORMATION PROVIDED HEREIN (THE "INFORMATION") IS PROVIDED ON AN "AS IS" BASIS AND FOR GENERAL INFORMATION PURPOSES ONLY. ORACLE EXPRESSLY DISCLAIMS ALL WARRANTIES OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. ORACLE MAKES NO WARRANTY THAT THE INFORMATION IS ERROR-FREE, ACCURATE OR RELIABLE. ORACLE RESERVES THE RIGHT TO MAKE CHANGES OR UPDATES AT ANY TIME WITHOUT NOTICE.

As individual requirements are dependent upon a number of factors and may vary significantly, you should perform your own tests and evaluations when making technology infrastructure decisions. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle Corporation or its affiliates. If you find any errors, please report them to us in writing.

Third Party Content, Products, and Services Disclaimer

This document may provide information on content, products, and services from third parties. Oracle is not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Limitation of Liability

IN NO EVENT SHALL ORACLE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES, OR DAMAGES FOR LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY YOU OR ANY THIRD PARTY, WHETHER IN AN ACTION IN CONTRACT OR TORT, ARISING FROM YOUR ACCESS TO, OR USE OF, THIS DOCUMENT OR THE INFORMATION.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Process Classification FrameworkSM (PCF) was developed by APQC and member companies as an open standard to facilitate improvement through process management and benchmarking regardless of industry, size, or geography. The PCF organizes operating and management processes into 12 enterprise-level categories, including process groups and over 1,500 processes and associated activities. The PCF and associated measures and benchmarking surveys are available for download and completion at no charge at www.apqc.org/OSBCdatabase.

ABOUT APQC

APQC is the leading resource for performance analytics, best practices, process improvement, and knowledge management worldwide. APQC’s research studies, benchmarking databases and renowned Knowledge Base provide managers with intelligence that they can use to transform their organizations for better results. A member-based nonprofit founded in 1977, APQC currently serves over 750 of the Fortune Global 1000 companies as well as hundred of education and government organizations. Learn more by visiting www.apqc.org or calling 800-776-9676 or 1-713-681-4020.
# Contents

Send Us Your Comments ........................................................................................................................................................................... ix

Preface ........................................................................................................................................................................................................... xi
  Document Purpose ........................................................................................................................................................................... xii
  Audience ....................................................................................................................................................................................... xii
  Document Structure ...................................................................................................................................................................... xii
  How to Use This Document ........................................................................................................................................................ xiii
  Conventions .................................................................................................................................................................................... xiii

## 1 Introduction

1.1 Service Portfolio Management ......................................................................................................................................................... 1-1
1.2 Identifying Services and Service Reuse Opportunities ........................................................................................................... 1-3

## 2 SOA Requirements

2.1 Traditional Approach to Requirements ........................................................................................................................................ 2-1
2.2 Why a New Method is Needed for SOA ....................................................................................................................................... 2-2
2.3 Applying SOA-based Requirements ........................................................................................................................................... 2-3
  2.3.1 Gather & Review Project Requirements .................................................................................................................................. 2-4
  2.3.2 Refine Project Requirements into Enterprise Requirements ......................................................................................................... 2-4
  2.3.3 Create / Expand Business Function Model .................................................................................................................................. 2-5
    2.3.3.1 Functional Requirement Decomposition .................................................................................................................................... 2-7
    2.3.3.2 Application Decomposition ......................................................................................................................................................... 2-8
    2.3.3.3 Business Process Decomposition ............................................................................................................................................... 2-8
    2.3.3.4 Expand Business Function Model ........................................................................................................................................... 2-9
  2.3.4 Classify Enterprise Requirements Against Function Model ................................................................................................... 2-10
  2.3.4.1 Check for Duplicate Requirements ........................................................................................................................................ 2-12
  2.3.4.2 Define Data Consumption Model ........................................................................................................................................... 2-14
  2.4 The Importance of an Enterprise Repository ............................................................................................................................. 2-15

## 3 Service Identification & Discovery

3.1 Identify Service Candidate ............................................................................................................................................................. 3-2
  3.1.1 Functional Activity Analysis ................................................................................................................................................... 3-2
  3.1.1.1 Examine Business Function Model Node(s) ................................................................................................................................. 3-3
  3.1.2 Business Entity Analysis .......................................................................................................................................................... 3-6
3.2 Create Service Candidates & Define Reuse............................................................................ 3-7
3.2.1 Candidate Justification........................................................................................................ 3-7
3.2.2 Reuse Validation............................................................................................................... 3-9

4 Service Boundary Analysis
4.1 Influencing Factors ........................................................................................................... 4-1

5 Summary

A Function Modeling
5.1 Functional Decomposition............................................................................................... A-3

B Data Modeling
5.2 Domain Data Hierarchy....................................................................................................... B-1
5.2.1 Semantic Community...................................................................................................... B-2
5.2.2 Data Authority................................................................................................................ B-2
5.2.3 Sources........................................................................................................................ B-2
5.2.4 Entities........................................................................................................................ B-3

C Service Naming Guidelines
5.3 Service Names .................................................................................................................. C-1
5.3.1 Nouns vs. Verbs ............................................................................................................. C-1
5.3.1.1 Service Naming Guidelines................................................................................... C-2
5.3.1.2 Service Operation Naming.................................................................................... C-3

D Further Reading
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Service Portfolio Management</td>
<td>1-2</td>
</tr>
<tr>
<td>1-2</td>
<td>Key Processes &amp; Techniques for Identifying &amp; Discovering Services</td>
<td>1-3</td>
</tr>
<tr>
<td>2-1</td>
<td>Traditional Approach to Requirements</td>
<td>2-2</td>
</tr>
<tr>
<td>2-2</td>
<td>SOA Based Requirements</td>
<td>2-3</td>
</tr>
<tr>
<td>2-3</td>
<td>SOA Requirements Process</td>
<td>2-4</td>
</tr>
<tr>
<td>2-4</td>
<td>Example Requirements Lifecycle</td>
<td>2-5</td>
</tr>
<tr>
<td>2-5</td>
<td>Create / Expand Business Function Model Process</td>
<td>2-6</td>
</tr>
<tr>
<td>2-6</td>
<td>Example Partial Business Function Model</td>
<td>2-7</td>
</tr>
<tr>
<td>2-7</td>
<td>Functional Requirements Example</td>
<td>2-8</td>
</tr>
<tr>
<td>2-8</td>
<td>Business Process Activities</td>
<td>2-9</td>
</tr>
<tr>
<td>2-9</td>
<td>Expanded Business Function Model</td>
<td>2-10</td>
</tr>
<tr>
<td>2-10</td>
<td>Classified Requirements</td>
<td>2-11</td>
</tr>
<tr>
<td>2-11</td>
<td>Classify Requirements Against Function Model Process</td>
<td>2-12</td>
</tr>
<tr>
<td>2-12</td>
<td>Function Requirement Examination</td>
<td>2-13</td>
</tr>
<tr>
<td>2-13</td>
<td>Data Decomposition Example</td>
<td>2-14</td>
</tr>
<tr>
<td>2-14</td>
<td>Data Consumption Model</td>
<td>2-15</td>
</tr>
<tr>
<td>2-15</td>
<td>SOA Requirements Management - Repository</td>
<td>2-16</td>
</tr>
<tr>
<td>3-1</td>
<td>Service Candidate Identification and Discovery Process</td>
<td>3-2</td>
</tr>
<tr>
<td>3-2</td>
<td>Functional Activity Analysis Process</td>
<td>3-3</td>
</tr>
<tr>
<td>3-3</td>
<td>Complete Service Overlap Example</td>
<td>3-4</td>
</tr>
<tr>
<td>3-4</td>
<td>Partial Service Overlap Example</td>
<td>3-5</td>
</tr>
<tr>
<td>3-5</td>
<td>No Service Overlap Example</td>
<td>3-6</td>
</tr>
<tr>
<td>3-6</td>
<td>Service Candidate Justification Process</td>
<td>3-7</td>
</tr>
<tr>
<td>3-7</td>
<td>Service Candidate Selection Framework</td>
<td>3-8</td>
</tr>
<tr>
<td>3-8</td>
<td>Reuse Validation Process</td>
<td>3-9</td>
</tr>
<tr>
<td>5-1</td>
<td>Example Function Model Hierarchy Structure</td>
<td>A-1</td>
</tr>
<tr>
<td>5-2</td>
<td>Example Function Model</td>
<td>A-2</td>
</tr>
<tr>
<td>5-3</td>
<td>Major Business Functions</td>
<td>A-3</td>
</tr>
<tr>
<td>5-4</td>
<td>Business Process Groups</td>
<td>A-4</td>
</tr>
<tr>
<td>5-5</td>
<td>Example Domain Data Hierarchy Structure</td>
<td>B-1</td>
</tr>
<tr>
<td>5-6</td>
<td>Example Domain Data Model</td>
<td>B-2</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2–1</td>
<td>Refined Harvest Names</td>
<td>2-7</td>
</tr>
<tr>
<td>2–2</td>
<td>Requirements Classification Example</td>
<td>2-13</td>
</tr>
<tr>
<td>3–1</td>
<td>Example Criteria for Realization Benefits</td>
<td>3-8</td>
</tr>
<tr>
<td>3–2</td>
<td>Example Criteria for Realization Inhibitors</td>
<td>3-9</td>
</tr>
<tr>
<td>4–1</td>
<td>Example Service Boundary Analysis Influencing Factors</td>
<td>4-1</td>
</tr>
<tr>
<td>5–1</td>
<td>Service Naming Influences</td>
<td>C-1</td>
</tr>
<tr>
<td>5–2</td>
<td>Business Service Nouns vs. Verbs</td>
<td>C-1</td>
</tr>
<tr>
<td>5–3</td>
<td>Example Service Naming Guidelines</td>
<td>C-2</td>
</tr>
</tbody>
</table>
Identifying and Discovering Services, Release 3.2
E14487-03

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this document?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us at its_feedback_ww@oracle.com.
Preface

**IT Strategies from Oracle (ITSO)** is a series of documentation and supporting collateral designed to enable organizations to develop an architecture-centric approach to enterprise-class IT initiatives. ITSO presents successful technology strategies and solution designs by defining universally adopted architecture concepts, principles, guidelines, standards, and patterns.

ITSO is made up of three primary elements:

- **Oracle Reference Architecture (ORA)** defines a detailed and consistent architecture for developing and integrating solutions based on Oracle technologies. The reference architecture offers architecture principles and guidance based on recommendations from technical experts across Oracle. It covers a broad spectrum of concerns pertaining to technology architecture, including middleware, database, hardware, processes, and services.

- **Enterprise Technology Strategies (ETS)** offer valuable guidance on the adoption of horizontal technologies for the enterprise. They explain how to successfully execute on a strategy by addressing concerns pertaining to architecture, technology, engineering, strategy, and governance. An organization can use this material to measure their maturity, develop their strategy, and achieve greater levels of success and adoption. In addition, each ETS extends the Oracle Reference Architecture by adding the unique capabilities and components provided by that particular technology. It offers a horizontal technology-based perspective of ORA.

- **Enterprise Solution Designs (ESD)** are industry specific solution perspectives based on ORA. They define the high level business processes and functions, and the software capabilities in an underlying technology infrastructure that are
required to build enterprise-wide industry solutions. ESDs also map the relevant application and technology products against solutions to illustrate how capabilities in Oracle’s complete integrated stack can best meet the business, technical, and quality of service requirements within a particular industry.

This document is part of a series of documents that comprise the SOA Enterprise Technology Strategy, which is included in the IT Strategies from Oracle collection. Please consult the ITSO web site for a complete listing of SOA and ORA documents as well as other materials in the ITSO series.

**Document Purpose**

A Practitioner Guide provides insight and guidance when working with a particular type of technology and address the common concerns faced by enterprises and practitioners.

![Diagram of Enterprise Technology Strategies]

This Practitioner Guide provides an approach to identifying and discovery Services within an SOA environment. It describes guidelines and best practices that enterprises and practitioners can utilize to develop a consistent and repeatable approach to identifying Services.

**Audience**

The primary audience for this guide is those who are responsible and accountable for identifying Services when defining Service portfolios and delivering projects in an SOA environment. A secondary audience are those who are responsible for designing and implementing services and those responsible for defining business requirements.

**Document Structure**

This document is organized into the following sections.

*Chapter 1* - an overview of Oracle’s approach to identifying Services as well as its position within Service Portfolio Management.

*Chapter 2* - provides an approach to defining and classifying enterprise project requirements against a business function model.

*Chapter 3* - provides an approach to identifying Service Candidates from the business function model and associated business entities.
Chapter 4 - provides a description of Service boundary considerations. in realizing a Service Candidate.

Chapter 5 - provides a summary of this document.

Appendix A - provides a high-level description of function modeling and its benefits when identifying Services.

Appendix B - provides a high-level description of key terminology when using data modeling to identify Services.

Appendix C - provides an example Service naming guidelines.

Appendix D - provides a list documents for further reading.

How to Use This Document

This document should be read by everyone that is interested in learning about or leveraging an approach to identifying Services.

Readers should refer to the Software Engineering in an SOA Environment document for background material and how a Service identification approach fits within an overall Service Engineering framework.

Conventions

The following typeface conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>boldface text</td>
<td>Boldface type in text indicates a term defined in the text, or in the separate glossary document, or in both locations.</td>
</tr>
<tr>
<td>italic text</td>
<td>Italics type in text indicates the name of a document or external reference.</td>
</tr>
<tr>
<td>underline text</td>
<td>Underline text indicates a hypertext link.</td>
</tr>
</tbody>
</table>

In addition, the following conventions are used throughout the SOA ETS documentation:

“Service” v. “service” - In order to distinguish the “Service” of Service Oriented Architecture, referred to throughout the SOA ETS document series, the word appears with its initial letter capitalized (“Service”), while all other uses of the word appear in all lower-case (e.g. “telephone service”); exceptions to this rule arise only when the word “service” is part of a name, such as, “Java Message Service” (“JMS”), “Web Service”, etc.
Leading enterprises are gaining operational efficiencies and business agility through adaptable, re-usable business processes and services built on a truly flexible Service-Oriented Architecture (SOA) foundation. Some enterprises are still attempting to use traditional delivery approaches for SOA. However, these approaches are designed to deliver solutions that do not consider the requirements of the business outside of the scope of the project in question. These methods are too narrowly focused and need to be adjusted to enable SOA.

Utilizing a traditional approach becomes an inhibitor to reuse and agility and this requires enterprises to address these inhibitors by adopting a sound, enterprise-class service engineering and modeling framework. The Oracle Service Engineering Framework (see Software Engineering in an SOA Environment document for more detail) is an engineering approach for delivering projects within an SOA environment. It identifies the unique software engineering challenges faced by enterprises adopting SOA and provides a framework to remove the hurdles and improve the efficiency of the SOA initiative.

1.1 Service Portfolio Management

One such challenge is identifying Services that enable the flexibility and agility that SOA extols, while at the same time delivering value to projects. A successful approach to Service Identification must bring together two major disciplines, Service Engineering and SOA Governance. One key area where these two disciplines intersect and enable Service Identification is within Service Portfolio Management (SPM). See Figure 1–1 for the interaction of Service Portfolio Management and traditional Portfolio Management disciplines.

IT Project Portfolio Management allows enterprises to collaboratively propose, fund, prioritize, plan, align, and control their IT initiatives to achieve their objectives, while understanding the impact of changing or adding initiatives to their portfolios. IT Application Portfolio Management utilizes operational metrics to measure the benefits and continual investment into each application and how it aligns and supports the business.
Service Portfolio Management enables an enterprise to manage a long-term Service portfolio, which defines the right set of Services, and enables when, where, and how they are used. Service Portfolio Management allows enterprises to:

- Identify and classify a coherent portfolio of Services rather than an ad-hoc approach to identifying Services that leads to Service sprawl (i.e. duplicate Services, no Service reuse, etc.)
- Define and utilize a Service Candidate selection framework to analyze whether a Service Candidate is realized as a shared Service or not. (See *Creating an SOA Roadmap* document for more details)
- Define the schedule in which Service Candidates and Service modifications are realized. The prioritization approach should cater for effort, complexity, resource availability, and the delivery date driven by the needs of the projects requiring the Services.
- Define a Service sourcing and ownership strategy based on build vs. buy vs. borrow (SaaS) strategies.
- Define a set of processes and techniques that continuously measures SOA investments against business goals (e.g., cost reduction, IT simplification, process agility). The intent is to maximize the return and alignment of SOA assets and investments to the enterprise, while making timely investments in SOA and effectively managing change.
This document primarily focuses on the first two points detailing an approach to identifying candidate Services and justifying their use via a selection framework. There are additional approaches to Service identification that have not been included within this document, but this document covers an approach that is pragmatic and lays down a foundation which other Service identification techniques can build upon.

1.2 Identifying Services and Service Reuse Opportunities

Identifying appropriate Services and Service reuse opportunities is a fundamental tenet of SOA. Therefore, a repeatable and pragmatic approach is required for the successful adoption and execution of an SOA. The key enabler of a successful approach is the availability of information and an understanding of where to look to get the information. Human instinct is always going to be a fundamental part of any analysis exercise when not all of the required information is available to make a decision. However, the more information made available to the process the more repeatable, consistent, and effective it will be.

Figure 1–2  Key Processes & Techniques for Identifying & Discovering Services

To address the need around managing and making this information available, it is key that an enterprise metadata repository is used to enable the identification of Services
and Service reuse opportunities. See Figure 1-2 for the key processes and techniques and their relationships with an enterprise repository.

- **SOA Requirements** - Project requirements are iteratively refined into enterprise requirements that are used to build/extend the enterprise's business function model. The enterprise requirements are then classified against the enterprise's business function model.

- **Service Identification & Discovery** - Enterprise requirements are analyzed to identify Service Candidates, Service modifications, and Service reuse opportunities.

- **Reuse Validation** - Identified Service reuse opportunities are analyzed to determine if they should be reused by the project or not.

- **Candidate Justification** - Proposed Service Candidates are analyzed to determine if they should be realized as shared Services or not.

- **Service Boundary Analysis** - Proposed Service Candidates are analyzed to identify the Service boundaries by analyzing various influencing conditions.

The remainder of this document details each of the above highlighted techniques and processes.
SOA requirements represents an extension to, rather than a replacement for, traditional requirements gathering techniques. The requirements gathering procedures established by an enterprise are reused and extended by the approach outlined in this document. However, more interaction is required between business and IT in order for the approach to be successful. The business will no longer be able to simply draft requirements and throw them over the wall, since an iterative procedure is required.

A few of the key differentiators of SOA based requirements are:

- Requirements are harvested with an eye towards enterprise service-orientation, and then the requirements are iteratively refined into enterprise requirements.
- Requirements are used to construct an enterprise scoped business function model that allows an enterprise to define an appropriate taxonomy that assists with Service Identification, Service Discovery, Service Naming, and requirement tracking.
- Requirements are classified against the business function model rather than within a single project’s scope.

### 2.1 Traditional Approach to Requirements

The traditional approach to requirements gathering is fundamentally focused on the requirements for a single project. This method works very effectively for constructing applications but has always resulted in problems with enterprise planning, integration, and enterprise scaled reuse, not to mention agility. Figure 2–1 below illustrates a traditional approach.
In the traditional approach, the business constructs their objectives and goals for the project and work with analysts to develop detailed requirements. These requirements are then thrown over the wall for IT to deliver in the form of an application. From then on, the business involvement is typically minimal. The business sometimes has very little interaction with the project until the onset of user acceptance testing. There are obvious variations to the approach used between enterprises, but the result is typically the same: an application built as a silo with little consideration for existing or future business functional reuse. The application is then deployed in production and managed in isolation.

The traditional approach tends to break down when applications need to be developed as part of a larger SOA because these applications need to be planned at the enterprise level, rather than in a self-contained silo. From the enterprise perspective, some semblance of integration and reuse could be achieved by establishing an n-tiered application architecture and prescribing this architecture to the delivery teams for individual projects. This technique simply drove the enterprise-level integration between applications and data into the applications themselves, rather than addressing it at the enterprise level. The reuse achieved was typically compile-time reuse through shared libraries with any runtime reuse achieved only through integration. This resulted in applications being chained together through tight coupling.

2.2 Why a New Method is Needed for SOA

In order to achieve effective SOA, applications simply cannot be developed independently of one another. SOA applications utilize shared services that are not owned by any single application, have their own lifecycle separate from any application, and are managed independently. In order to be effective in managing requirements within an SOA, projects have to be aware of the requirements of existing applications, in-flight projects, and proposed projects.

This approach is one of the fundamental enablers for Service identification and discovery, as well as the centralized information required to achieve SOA release planning. (See Software Engineering in an SOA Environment document for a more
detailed description.) As previous highlighted, the traditional method of requirements management is not designed to handle these demands.

### 2.3 Applying SOA-based Requirements

As with any requirements gathering process, or analysis process for that matter, a certain amount of facilitation and coordination skills are required. Requirement gathering requires an attention to detail as well as the foresight required to ask the questions necessary to fill in the gaps and gain an understanding of functional and non-functional requirements for a system.

SOA based requirements also requires these same skills to be applied at the enterprise level with the addition of extra analytical skills required to create a business function model, classify requirements and models, and analyze the project requirements against existing enterprise requirements. These skills are essential to navigate through the SOA based requirements gathering exercises.

**Figure 2–2  SOA Based Requirements**

Figure 2–2 above illustrates the high-level aspects of enterprise SOA-based requirements.

- Traditional requirements management method is used to draft the project based requirements for an application.
- The project requirements are iteratively refined into enterprise requirements that are used to build and extend the enterprise’s business function model.
- The enterprise requirements are classified against the enterprise’s business function model.
- An enterprise repository is used to enable and publish the overall SOA-based requirements approach.
Applying SOA-based Requirements

The result is a taxonomy of requirements based on the enterprise’s business function model, rather than requirements grouped by specific areas of application functionality. This allows duplicate or overlapping requirements to be identified in addition to providing a basis for looking at requirements with potential reuse in mind. This approach, therefore, provides a natural avenue for enabling a Service identification and discovery approach.

Figure 2–3  SOA Requirements Process

Figure 2–3 highlights the 4 primary steps of taking a SOA based approached to requirements management. The rest of this section will go into more detailed steps and tasks.

2.3.1 Gather & Review Project Requirements

As highlighted earlier in this section, a SOA-based requirements process represents an extension to, rather than a replacement for, traditional requirements gathering techniques. The requirements gathering procedures established by an enterprise are reused and extended by the approach outlined in this document.

2.3.2 Refine Project Requirements into Enterprise Requirements

A key technique to assist with identifying Service Candidates is functional decomposition. In order to support SOA-based functional decomposition, requirements must be presented in a format suitable for the scope of the SOA rather than just for the project’s individual needs.

Raw project requirements (or use cases, storyboards - depending on requirements management approach used by the enterprise) that have been gathered need to be assessed to ensure that the appropriate level of detail has been captured and verified. In order for the requirements to be valuable in defining a business function model, they must be formatted in a project neutral format. Requirements that have project specific terminology should be cleansed then reviewed with the project team and business representatives to ensure the requirements are still valid. This iterative process should continue until agreement has been reached on the details of the requirements.

Scoping requirements at the enterprise level, rather than at the application or project level, enables project architecture and planning exercises to remain connected to the overall enterprise SOA. This connection also provides a natural enabler for Service discovery, identification, and Service release planning.
Apart from the standard information gathered for a requirement (e.g., description, priority), additional attributes are required to classify and enrich requirements to be a viable asset at the enterprise level. Examples of these attributes include lifecycle status, type, and owner. The lifecycle status represents the current state of the requirement. See Figure 2–4 for an example.

**Figure 2–4  Example Requirements Lifecycle**

A requirement is initially proposed and a decision is made by the appropriate governance authority to accept or decline the requirement. If the requirement has been accepted, then it has been validated in terms of understanding the rationale behind the requirement, and it has been described precisely enough to be realized. At some point, the requirement will be assigned to a delivery project (application or Service project) to be realized. At this time the requirement owner is identified as a specific application or it may be identified as being “Shared”. The requirement is then fulfilled by being implemented, and when and if it is no longer valid, the requirement can be retired.

**2.3.3 Create / Expand Business Function Model**

The project requirements that have been iteratively refined into enterprise requirements should be classified against the enterprise’s business function model. Refer to Figure 2–5 for the expanded process.
Rather than build a fully developed business function model for an enterprise, since this can take an extended period of time, it is best practice to build out the relevant parts of the business function model as and when it is required. Figure 2–6 highlights a hierarchal representation of a partial business function model based on the APQC Process Classification Framework. (See Appendix A for more detail around the structure of a business function model.)
Any part of the business function model that has not been developed yet can be extended by using a number of decomposition techniques. There are many entry points into using a decomposition technique to build a business function model. Figure 2–5 highlights a few of the most commonly used decomposition techniques, and as more and more project requirements go through these decomposition techniques, the business function model will become more complete and less expansion will be required and encountered.

### 2.3.3.1 Functional Requirement Decomposition

A functional requirement decomposition approach takes raw requirements such as text document, use cases, storyboards, etc. (depending on methodology used by the enterprise) and refines them to elaborate and extend the existing business function model. It is common for this approach to be called the “Top-Down” approach.

#### Figure 2–7 Functional Requirements Example

Below are the high-level steps that assist with this approach. (Over time, experienced practitioners collapse these steps into one streamlined activity.)

- Requirements should be reviewed noting the key verbs and associated nouns (see Figure 2–7).
- Specifics should be removed as a business function model is concerned with the “What” and is not concerned with the “Who”, “How”, or “When”.
- Appropriate business terminology should replace any project specific or unnecessary minor words.
- The identified function name should be based around a present tense “verb + singular noun” (best effort). See Table 2–1 for example refined names.

#### Table 2–1 Refined Harvest Names

<table>
<thead>
<tr>
<th>Harvest</th>
<th>Refined</th>
</tr>
</thead>
<tbody>
<tr>
<td>An expense request must be completed by a consultant at the end of each engagement</td>
<td>Complete Expense Request</td>
</tr>
</tbody>
</table>
2.3.3.2 Application Decomposition

An application decomposition approach takes existing applications and identifies sources of functionality to service enable. It is common for this approach to be called the “Bottom-Up” approach and is commonly used to modernize an enterprise's integration architecture. While this approach does have some value, this approach can lead to a number of challenges.

- Not all functionality that should be service enabled is identified.
- It is easier to define the wrong service granularity.
- SOA is not solely about integration so higher valued IT and business benefits will be neglected.

To overcome these challenges it is best to use a functional requirement decomposition approach coupled with the application decomposition approach. This allows a practitioner to analyze applications and identify the appropriate functional boundaries. This can then be used to elaborate and extend the existing business function model. It is common for this approach to be called the “Hybrid” or “Meet in the Middle” approach.

2.3.3.3 Business Process Decomposition

An approach that uses business process decomposition takes requirements defined in various sources, such as text documents, business process models, and business context models and refines them to elaborate and extend the existing business function model.

During the elaboration process of documenting a business process, individual process activities at all business function levels (see Figure 2–8) should be considered when expanding the existing business function model. A common error is to include technology specific activities within a business process model. It is important that these are not considered as part of the business function model.

**Table 2–1 (Cont.) Refined Harvest Names**

<table>
<thead>
<tr>
<th>Harvest</th>
<th>Refined</th>
</tr>
</thead>
<tbody>
<tr>
<td>All expense requests are reviewed by the office coordinator before approval by the consultant’s manager.</td>
<td>Review Expense Request</td>
</tr>
<tr>
<td></td>
<td>Approve Expense Request</td>
</tr>
<tr>
<td>The finance department has the responsibility to verify the expense request against the expense policies</td>
<td>Verify Expense Request</td>
</tr>
<tr>
<td>Funds are sent weekly to consultants by the finance department</td>
<td>Transfer Funds</td>
</tr>
</tbody>
</table>
2.3.3.4 Expand Business Function Model

The identified functions are then used to expand the existing business function model. It may be necessary to add additional levels to group related activities. Figure 2–9 shows an expanded business function model where the identified functions (highlighted by the ticks) have been added to the partial business function model that is based on the APQC framework.
2.3.4 Classify Enterprise Requirements Against Function Model

The classification of functional models and requirements is achieved iteratively between the business and IT divisions. The business provides input regarding the overall business process, while the delivery team and SOA leadership team develop the models and classifications that are then crosschecked with the business. The requirement is attached to the appropriate node within the business function model based on granularity and scope (see Figure 2–10).
As the requirements are classified against the nodes within a business function model, it is important that the origination of the requirement be documented. Once this is done it is possible to check whether there are overlapping or duplicate requirements (see Figure 2–11).
2.3.4.1 Check for Duplicate Requirements

The first step is to determine if there are any existing enterprise requirements which partially or fully cover the proposed requirement.

If an existing requirement fully covers the proposed requirement, then simply earmark the existing requirement as being relevant to the project. If an existing requirement partially covers a proposed requirement, propose a new version of the existing requirement and earmark both as being relevant to the project. Another potential scenario that would cause a new version to be split is when an existing requirement fully covers the proposed requirement but is partially implemented, with the unimplemented portion being required for this project. If there are no matches for the proposed requirement, register the new requirement with the repository and earmark it for the project. Refer to Figure 2–12 for an example.
Figure 2–12 Function Requirement Examination

Figure 2–12 shows a scenario where the requirements from three projects are classified and attached to a business function model. Refer to Table 2–2 for an overview of the steps.

Table 2–2 Requirements Classification Example

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1    | - Project ABC attaches a requirement against 8.6.1.2.1 Review Expense Request.  
      - There are no other requirements at this function node; so this requirement will be classified as a New Requirement. |
| 2    | - Project XYZ attaches 2 requirements which are attached to 8.6.1.3.2 Transfer Funds and 8.6.1.2.1 Review Expense Request.  
      - There are no other requirements at the 8.6.1.3.2 Transfer Funds function node; so this requirement will be classified as a New Requirement.  
      - The requirement encountered at 8.6.1.2.1 Review Expense Request is a sub-set of the requirements classified by Project XYZ; therefore a new version of the existing requirement is proposed that is versioned and extended. |
2.3.4.2 Define Data Consumption Model

The objective of a Data Consumption Model is to understand the key business entities within the scope of the project(s), determine where they fit with respect to semantic communities, and organize this information into a form that can be communicated and classified. (See Appendix B for more detail on these definitions.)

Figure 2–13 Data Decomposition Example

Table 2–2 (Cont.) Requirements Classification Example

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>• Project DEF attaches a requirement against 8.6.1.3.2 Transfer funds.</td>
</tr>
<tr>
<td></td>
<td>• An identical existing requirement at 8.6.1.3.2 Transfer Funds is encountered. This requirement is then earmarked as a Duplicate Requirement, which gives a practitioner an indication that there is a potential reuse when identifying Services.</td>
</tr>
</tbody>
</table>

Below are the high-level steps that assist with this approach. (Over time, experienced practitioners collapse these steps into one streamlined activity.)

- Requirements attached to each node within the Business Function Model are reviewed noting nouns that infer business entities (See Figure 2–13).
- Understand synonyms and relationships between business entities.
- Convert pronouns (e.g. they, he, it) into inferred business entities.
- Identify possible actions on entities by noting verbs.
- Construct a Data Consumption model by detailing the business entity, its semantic community, and the identified CRUD operations applied against each business entity. (See Figure 2–14)
2.4 The Importance of an Enterprise Repository

A key tool during the SOA-based Requirements Management activity is an enterprise repository to capture the SOA assets and their dependencies to other SOA assets and requirements. Some of the dependencies can be seen in Figure 2–15.

- Projects and application requirements are linked (classified) against the business function model.
- Business processes, Services, and applications are linked to the implemented requirements.
- Data consumption and provisioning requirements linked to the relevant business functions.
Figure 2–15  SOA Requirements Management - Repository
Service Identification & Discovery

Service Identification deals with the procedures and guidelines that an enterprise adopts to identify new shared Service Candidates. Discovery is the mechanism whereby existing shared Services can be easily identified as the need arises within a new project or application. Identification and discovery are equally important. These two topic areas share an obviously natural relationship to one another. In the process of analyzing requirements, new Service Candidates need to be considered on two angles. Either they have already been realized in the form of a shared Service, or they are a potential candidate for a shared Service.

- Service Identification begins with the analysis of requirements and ends with the identified set of Service Candidates. Service Candidates represent the set of business function that have been recommended for reuse. A Service Candidate does not necessarily map one-to-one to a Service because Service Candidates could be realized by the creation of one or more Services. This determination, however, is outside the scope of Service Identification but is part of Service Boundary Analysis.

- Service Discovery is a natural extension of the Service Identification technique whereby a practitioner analyzes requirements against existing Services to check that there may be viable reuse to fulfill some or all of the requirements. This is a more natural approach to Service Discovery compared to a complicated search paradigm that typically tends to be inefficient and inaccurate in identifying reuse candidates.

In order to achieve effective Service Identification & Discovery (SID&D), these processes cannot be simply worked through in a standalone manner. If this is attempted, there will simply not be enough information available to ensure consistent results. Refer to Figure 3–1.
The processes and guidelines around SID&D are a part of a larger Enterprise Service Engineering Framework that provides a connected methodology and best practice for achieving effective delivery of SOA based projects. (See *Software Engineering in an SOA Environment* document.)

### 3.1 Identify Service Candidate

The Identify Service Candidate process focuses on analyzing the requirements and the business function model found within an enterprise repository that was populated and expanded as part of an SOA Requirements approach (as discussed in the previous chapter).

This does not actually change the business function model; rather it identifies new Service Candidates, Service Candidates covering the extension of an existing Service, and Service reuse prescriptions.

#### 3.1.1 Functional Activity Analysis

During Functional Activity Analysis the functional activities and their attached requirements are examined to discovery Service Candidates and Service reuse opportunities (refer to Figure 3–2).
3.1.1.1 Examine Business Function Model Node(s)

The business functions that have attached project requirements are examined and analyzed. By examining the requirements of each functional activity, reuse candidates, Business Process Service Candidates, and Business Activity Service Candidates are identified. (See ORA SOA Foundation document for details around Service Categories.)

When examining a business function node, there are three potential scenarios:

- Complete Service overlap,
- Partial Service overlap,
- No Service overlap.

3.1.1.1.1 Complete Service Overlap If all requirements within the business function node being examined are linked to the current project and are also linked to one or more shared Services, then this is known as a “Complete Service Overlap”.

---

**Figure 3-2  Functional Activity Analysis Process**
In Figure 3–3 the node “8.6.1.3.2 Transfer Funds” is being examined. In this node, all of the requirements are both linked to the current project as well as the TransferFunds Service. In this scenario, the TransferFunds Service should be validated for reuse by the current project.

3.1.1.1.2 Partial Service Overlap If some, but not all, of the requirements in the examined business function node (that are linked to the project) are also linked to a shared Service, then this is known as a “Partial Service Overlap”.

**Figure 3–3 Complete Service Overlap Example**
In Figure 3–4 the node “8.6.1.3.2 Transfer Funds” is being examined. In this node, some of the requirements are linked to the project and some are linked to both the project and the TransferFunds Service. In this scenario, there are a number of options:

- Simply reuse the existing Service and realize the remaining requirements within the project.
- Propose a new Service Candidate from the remaining requirements.
- Propose an extension to the existing TransferFunds Service to include the new requirements.

3.1.1.3 No Service Overlap The requirements located within the business function node are not linked to any Service. They may be linked to other projects but none of the requirements has been realized within a shared Service: this is known as “No Service Overlap”.

Figure 3–4 Partial Service Overlap Example
In Figure 3–5 the business function node “8.6.1.3 Approve Reimbursements and Advances” is being examined. Because none of the requirements in lower nodes have been put forward and approved as Service Candidates, the entire set of requirements is being put forward as a potential coarse-grained Service Candidate.

This technique, which can lead to coarse-grained Services, is critical to understand and implement. Otherwise a practitioner may be inclined to propose a Service for each individual node in the business function model. This can lead to too many fine-grained Services that may have an impact on performance once the Services are realized.

In addition, this technique allows a practitioner the ability to later refactor a coarse-grained Service into finer-grained Services. For example, the “8.6.1.3.2 Transfer Funds” business function node may later be justified as an autonomous Service, and the appropriate business functionality would be extracted from the current “Approve Reimbursements and Advances” Service. The current Service then uses a Service composition technique to replace the extracted functionality.

### 3.1.2 Business Entity Analysis

Analyzing the data needs by the identified candidate business Services and the data currently being used by identified reuse Service candidates in conjunction with the Data Consumption Model enables an architect to identify business entities and build a Canonical Message Model. A Canonical Message Model applies a Canonical Message Model to message formats by constructing messages structures, elements, and attributes using canonical data. The Canonical Message Model describes the structure
of the data that will be used to transported across the Business Function Model and, therefore, may later be used to standardize interfaces.

The architect can search the Service catalog for existing data Services that could support the needs of the identified candidate business Services. The architect may also need to search the Service catalog for existing business Services that have encapsulated data Service functionality that could support the needs of the identified candidate business Services.

The architect will then have enough information to identify:

- Candidate Data Services,
- Candidate reuse Data Services,
- Candidate Business Service modifications where encapsulated Data Service functionality could be refactored.

### 3.2 Create Service Candidates & Define Reuse

Service Candidates are Services that have not yet been justified for realization. This determination, however, is outside the scope of Identify Service Candidates process, and is part of the Service Candidate Justification process.

In addition, identified Service reuse prescriptions need to be validated to determine whether a proposed reuse candidate will be authorized for reuse by the application. This determination is also outside of the scope of Identify Service Candidates process but is part of the Service Reuse Validation process.

### 3.2.1 Candidate Justification

Service Justification determines if a proposed Service Candidate should be realized as a shared Service or not. A Service Candidate that is not justified is pushed back to the project for realization. The project team may opt to realize the Service Candidate as a Service, but it will not be hosted and managed within the realm of shared Services infrastructure.

Service Candidates must be proposed in a standardized and consistent structure. Therefore enterprises must have an agreed upon definition of a Service and its associated lifecycle states.

Apart from the standard information traditionally expected to be attached to a Service Candidate, (i.e. requirements) there are a number of other key attributes required that enrich the Service Candidate. Example key attributes include Service name, intended scope (i.e. Public, Multi-enterprise, Enterprise, Line of Business, and Application), and intended Service Classification (i.e. Presentation, Business Process, Business Activity, Data, Connectivity, Utility). See *ORA SOA Foundation* document for details regarding Service Scope and Service Categories.

Much has been written about the appropriate approach to naming a Service Candidate, whether a Service should be based around a noun or a verb. While both have merits, it is more important that an enterprise defines a Service-naming standard and enforces a consistent approach via establishing SOA governance policies. Without a consistent approach, enterprises will encounter challenges around Service Definition and Service Discovery that leads to Service duplication. (See Appendix C for a synopsis of a Service naming strategy.)

Once the requirements and key attributes have been determined, the Service Candidate is put forward for Service Justification.
Proposed Service Candidates are evaluated against a balanced scoring system to indicate whether a Service Candidate should be realized. Oracle has developed a Service Candidate Selection Framework that assists in selecting and justifying whether Service Candidates should be realized. (See *Creating an SOA Roadmap* document for more details).

A Service Candidate selection process should contain a number of criteria used to score the Service Candidates (refer to Figure 3–6).

**Figure 3–6  Service Candidate Justification Process**

The criteria covers both the benefits and inhibitors if the Service Candidate is realized. See Table 3–1 and Table 3–2 for example criteria.

**Table 3–1  Example Criteria for Realization Benefits**

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope</td>
<td>Weighs the benefit that the intended scope of the Service Candidate will have on the enterprise if realized.</td>
</tr>
<tr>
<td>Reuse</td>
<td>Weighs the potential reuse levels for the Service Candidate if realized.</td>
</tr>
<tr>
<td>Agility</td>
<td>Weighs the potential for enterprise agility resulting from the Service Candidate if realized.</td>
</tr>
<tr>
<td>Compliance</td>
<td>Weighs the potential for enterprise compliance resulting from the Service Candidate if realized.</td>
</tr>
<tr>
<td>Enablement</td>
<td>Weighs the potential to leverage existing enterprise assets to realize the functions of the proposed Service Candidate.</td>
</tr>
</tbody>
</table>
Any criteria used by an enterprise should be open to customization as and when the needs and priorities change. This can be achieved by adding/removing as well as weighting each criterion based on the relative importance of each. A balanced scoring system simply yields a 'Decision Basis Score' that is the sum of 'Realization Benefits' less the sum of the 'Realization Inhibitors'.

**Table 3–2   Example Criteria for Realization Inhibitors**

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skill-set Impact</td>
<td>Weighs the additional skill-set necessary to realize the functions proposed by the Service Candidate in a shared manner.</td>
</tr>
<tr>
<td>Project Impact</td>
<td>Weighs the additional impact placed on the proposed, in-flight, and operational projects that would be incurred by realizing the Service Candidate in a shared environment.</td>
</tr>
<tr>
<td>Technology Capability</td>
<td>Weighs the additional tools or technology necessary to realize the functions proposed by the Service Candidate in a shared manner.</td>
</tr>
<tr>
<td>QOS Feasibility</td>
<td>Weighs the level of difficulty and risk that will be encountered when trying to realize the nonfunctional requirements of the Service Candidate within a shared environment.</td>
</tr>
</tbody>
</table>

Figure 3–7 shows five Service Candidates and a Service modification. A positive 'Decision Basis Score' indicates a Service Candidate should be realized into a shared Service. Thus, Service Candidates 2, 5, and the Service Modification A should be realized while Service Candidates 1, 3, and 4 should not be realized (at this time).

A balanced scoring system can only ever be an indicator because justification cannot be fully prescribed. Therefore, it is critical that an enterprise defines an appropriate SOA governance structure to make the final decision. (See *A Framework for SOA Governance* document for more details.)

### 3.2.2 Reuse Validation

Service Reuse Validation is used to determine if a proposed Service reuse candidate should be reused by the project or not. This covers the instance where a Service...
Candidate has already been realized or planned to be realized and a new project would like to reuse the asset.

Before a Service can be validated for reuse by a project, a Service consumption request needs to be defined.

A Service consumption request provides details regarding the expected usage requirements along with the corresponding time periods, purpose, etc. Example details include:

- Expected (and maximum) load that consumer will place upon the Service.
- Maximum amount that the Service will consume - invocations per time (e.g., invocations/hour).
- Expected usage scenarios including days of the week, hours, and average consumption during the indicated time periods; and the peak usage scenarios including days of week, hours, and average consumption during the indicated time periods.

Once the Service reuse consumption requirements have been determined, the Service Candidate is put forward for Service Reuse Validation.

There are a number of validation criteria that should be used when determining if a proposed Service should be reused. Refer to Figure 3–8 for an illustration of how example validation criteria is utilized as part of a Service Reuse Validation process.

*Figure 3–8 Reuse Validation Process*

It is imperative that further analysis takes place to ensure that the existing Service being considered for reuse really satisfies the appropriate requirements of the project. If not all requirements will be covered then the enterprise should consider identifying an alternative solution such as proposing a Service extension.

In addition to reviewing the Service from a functional perspective, it should also be reviewed from a non-functional perspective. This ensures that the Service has been designed in a manner that is appropriate for the requesting consumer. This catches cases where the requirements match the consumer’s needs, however, other aspects of
the Service make the Service inappropriate for reuse as is. This may highlight the need to identify an alternative solution such as extending an existing Service, or even the creation of a new Service.

In particular a Reuse Validation Process should cater for validation against security policies, service design suitability, and expected capacity.

- **Security Policies** - Even though the Service may be a functional fit for the consumer, it still needs to be determined whether the consumer will be authorized to reuse the proposed Service. For example, the Service security policies should be reviewed to see if they prevent the project from reusing the Service.

- **Service Design** - The consumer must make sure that the Service reuse candidate has been designed in a manner that is appropriate. In particular the design regarding **QoS (Quality of Service)** as well as the interface may invalidate the Service for reuse by the specific consumer.

- **Capacity** - A key criterion to review is whether the required capacity or demand that the new project will require is available within the current Service-hosting environment. If the capacity requirements are not currently available, a determination needs to be made as to whether to expand the Service-hosting environment or reject the reuse candidate. If there is capacity, then the candidate can be approved for reuse and prescribed for the project. If the necessary capacity is not currently available in the Service hosting environment, but the determination has been made to extend that environment, then the appropriate extension plans need to be arranged to have the environment scaled. The Service is then approved for reuse and must be prescribed for the project.

Apart from the above example criteria, there are still additional considerations that should be reviewed. These considerations, such as change management, scheduling, access to resources, etc., requires a SOA governance framework to enable informed decisions. (See *A Framework for SOA Governance* document for some more details.)
A key activity in Service Identification is Service Boundary Analysis that determines if one or more Service boundaries are necessary in order to make the best use of hardware resources and satisfy the intended Service consumers who require the functions of the Service Candidate.

Typically, a Service Candidate will result in a single contract being defined leading to a single Service being realized after Service Delivery is completed. However, a Service Candidate may result in the definition of multiple Service Contracts since there are additional factors to consider when determining the Service boundaries based on a Service Candidate’s attached requirements.

### 4.1 Influencing Factors

The practitioner identifies the Service boundaries by analyzing various influencing factors against the Service Candidate requirements. Example influencing factors that affect Service boundaries can be found in Table 4–1.

<table>
<thead>
<tr>
<th>Influencing Factors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clarification of Service Candidates</td>
<td>If requirements fall into separate Service classifications, then there is a good chance that a Service boundary should be established, and that separate Services should be created for each scope that has different requirements.</td>
</tr>
<tr>
<td>Scope of Requirements</td>
<td>If requirements are different for differently scoped consumers, then there is a good chance that a Service boundary should be established, and that separate Services should be created for each scope that has different requirements.</td>
</tr>
<tr>
<td>Security Policies</td>
<td>If a Service candidate has a variety of different security policies spread across differing requirements across the Service Candidate, then it is likely that separate Services should be created along these different security requirements. This pattern usually provides a key indicator that significantly different types of consumers will be using the functions identified by the requirements.</td>
</tr>
</tbody>
</table>
Identifying and Discovering Services

Service Boundary Analysis is not an exact science, and its primary goal is to draw the practitioner’s attention to potential areas where a Service boundary may be required. There will certainly be situations where there are differences in the factors identified above, but there is not enough benefit to justify the creation of a separate Service.

Once the Service boundaries have been identified, a Service Contract for each identified Service should be defined. (See Software Engineering in an SOA Environment document for more details.)

Practitioners should expect and anticipate the need to change a Service. Services will evolve and go through a documented Service versioning process. In some circumstances, this will require the need to change the Service boundary and potentially promote a Service operation into a Service. Referring to Figure 3–5, "No Service Overlap Example", a coarse-grained Service was proposed with three Service operations (Verify Expense Request, Review Change Rate, and Transfer Funds). Over the course of time, it may be deemed that the Service operation 'Transfer Funds' should be promoted to a fully-fledged Service. This scenario occurs when a number of articulation points are considered against an individual Service operation. Example articulation points include:

- Factors described in Table 4–1.
- Increased probability of change due to either business or technical issues.
- Service composition requirements.
- Failure conditions and risk of failure.

Using the Business Service naming strategy highlighted in Appendix C, it is a matter of proposing a new Service Candidate named 'TransferFunds'. This allows new consumers to access the new Service but also allows the existing Service to delegate any calls to the existing Service operation to the new service without affecting the original Service consumers.

<table>
<thead>
<tr>
<th>Influencing Factors</th>
<th>Description</th>
</tr>
</thead>
</table>
| Message Exchange Patterns | Variance in Message Exchange Patterns (MEP) are another indicator of a potential Service boundary. If certain requirements require a publish/subscribe pattern while others require request response, there is a good chance that separate Services should be defined along these lines. Reasons for identifying Service boundaries by Message Exchange Patterns include:
  - MEPs typically represent significantly different functional requirements.
  - QoS is usually significantly different across differing MEPs.
  - Different MEPs typically require separate interfaces. |
| Quality of Service Requirements | The QoS requirements of the Service candidate need to be examined since they may represent significant differences in infrastructure requirements. When these conditions exist, defining separate Services along QoS boundaries leads to greater flexibility especially when managing scalability within the shared environments. |
Many enterprises have discovered that traditional delivery methodologies tend not to consider the needs and requirements of projects outside of the project in focus. These methods are much too narrowly focused and need to be adjusted to enable SOA; otherwise this leads to major inhibitors and loss of agility.

These enterprises require an update to their existing delivery methodologies with key Service engineering and modeling activities that will allow for consistency across deliveries for projects that need to coexist.

One of the key challenges to adopting SOA is having a repeatable and pragmatic approach to identifying and discovering Services. It is important to have an approach that builds incrementally since it is imperative that existing projects still deliver value while adopting an SOA approach.

The method described in this document is not the only approach to Service Identification, but it demonstrates an approach that can be applied to an enterprise’s existing requirements gathering techniques.
A business function model is used to capture business functions and the relationships between them. In essence, these business functions summarize what an enterprise does.

Enterprise SOA business function modeling requires a structure whereby a list of business functions can be cataloged. There are many function model structures available, including industry standards such as ETOM (Enhanced Telecom Operations Map), SCOR (Supply-Chain Operations Reference model), and internally defined structures. See Figure 5–1 for an example.

**Figure 5–1   Example Function Model Hierarchy Structure**

The function model hierarchy forms the basis to develop a business function model that provides a way to define and locate business functions within and across the enterprise quickly. See Figure 5–2 for an example. One advantage of a business function model is that it remains much the same independent of the current organizational structure of the enterprise.
The following rules apply to function models:

- Each level in the function model has a specific designation with respect to functional granularity.
- Each decreasing level is finer grained with respect to functional representation when compared to the level(s) above.
- Each node in the graph may have exactly one parent (excluding the root node that has no parent).
- Cycles are not allowed in the model.
- A function model is navigated by narrowing functional granularity, and not by organization structure.

A business function model that prevents duplication of enterprise functionality across the model is one of the aspects that make it useful for application within SOA deployments. One of the major issues that the SOA strategy attempts to overcome is the duplication of critical business functions across systems. In many cases, this happens simply because there is a lack of visibility with respect to requirements and existing business and IT functions. A business function model assists in eliminating functional duplication and is an ideal fit for supporting the Service Identification & Discovery process.
5.1 Functional Decomposition

As enterprises develop projects through the SOA discipline, business functions are decomposed and broken down into finer-grained functions and Services rather than bulky applications. See Figure 5–3 for an example list of major business functions that an enterprise performs. These major business functions were adopted from the APQC framework. The APQC organization offer a process classification framework that is a taxonomy of cross-functional business processes intended to allow objective comparison of organizational performance within and among enterprises.

**Figure 5–3  Major Business Functions**

Each major business function can then be decomposed into process groups. (See Figure 5–4). Practitioners can continue to keep decomposing in an incremental and pragmatic manner.
Figure 5–4  Business Process Groups

7.0 Manage IT
- 7.1 Manage the business of IT
- 7.2 Develop and manage IT customer relationships
- 7.3 Manage business resiliency and risk
- 7.4 Manage enterprise information
- 7.5 Develop and maintain IT solutions
- 7.6 Deploy IT solutions
- 7.7 Deliver and support IT services
- 7.8 Manage IT knowledge

8.0 Manage Financial Resources
- 8.1 Perform planning and management accounting
- 8.2 Perform revenue accounting
- 8.3 Perform general accounting and reporting
- 8.4 Manage fixed assets
- 8.5 Process payroll

10.0 Manage Environmental Health & Safety
- 10.1 Determine health, safety, and environment impacts
- 10.2 Develop and execute health, safety, and environmental program
- 10.3 Train and educate employees
- 10.4 Monitor and manage health, safety, and environmental management program
- 10.5 Ensure compliance with regulations
- 10.6 Manage remediation efforts

11.0 Manage External Relationships
- 11.1 Build investor relationships
- 11.2 Manage government and industry relationships
- 11.3 Manage relations with board of directors
- 11.4 Manage legal and ethical issues
- 11.5 Manage public relations program
5.2 Domain Data Hierarchy

Enterprise SOA data modeling requires a structure whereby enterprise and project level modeling activities can be done in an organized manner. A domain data hierarchy provides the ability to visualize how data is organized and dispersed within the enterprise. While there are several structures that can be utilized, the structure in Figure 5–5 will be used as the basis for this document.

*Figure 5–5  Example Domain Data Hierarchy Structure*

The domain data hierarchy is used to capture key entities, data sources, authorities, semantic communities, and the relationships between them. The domain data hierarchy forms the basis to develop a domain data model (see Figure 5–6) that provides a way to locate data within and across the enterprise and quickly understand the guardianship of that data as well as the potential for semantic conflicts between groups of people.
As Services are built, data provisioning will require such knowledge to support Service engineering efforts.

5.2.1 Semantic Community

A semantic community is a set of stakeholders that share a common understanding of data semantics, rules, and constraints, i.e., follow the same glossary. Semantic communities that are not clearly defined often follow division or departmental boundaries.

In an evolving Services-Oriented Architecture, it is common to have several communities for political, economic, or historical reasons. These communities typically have divergent definitions for similar or overlapping concepts. Services exposed across semantic communities may suffer from semantic incompatibilities. Services that attempt to bridge communities (e.g., single view of X) need to reconcile semantic differences.

5.2.2 Data Authority

A data authority is a set of stakeholders who are responsible for the stewardship of data. Authorities ensure data integrity, consistency, confidentiality, freshness, cleanliness, and enforce business rules pertaining to data entities such as semantics, relationships, and constraints. Authorities maintain the data glossary and set rules regarding the usage of data.

A technical authority is concerned with the physical aspects of stewardship (e.g., database management). A business authority is concerned with the business/conceptual rules and semantics. Data authorities set the terms of data usage and the types of access permitted when developing data services and accessing foreign sources.

5.2.3 Sources

Data sources are systems, applications, and Services that provide access to data. Sources generally provide access to persisted data, such as entities stored in a database, files in a file system, content in a Content Management (CM) system, etc.
Sources may also offer access to transient data such as system events, streaming data feeds, and messages from a subscription in a store-and-forward queue.

### 5.2.4 Entities

Key business entities are defined as those that can exist stand-alone and are significant to the business to support a business function. For instance, employee, customer, and expense report are examples of key entities, while dependent entities such as address, contact, and expense item are not. The reason for gathering key entities is to understand where they are maintained across the enterprise. Often, key entities are duplicated in multiple systems and databases. This presents challenges for an enterprise in terms of data rationalization and ubiquitous reuse via shared Services.
5.3 Service Names

One of the key elements of a Service Contract is the Service name. Defining an appropriate and consistent Service naming strategy is a key activity. Without a consistent approach, enterprises will encounter challenges around Service Definition and Service Discovery that leads to Service duplication.

5.3.1 Nouns vs. Verbs

Much has been written about an appropriate approach to naming a Service Candidate. One area of contention tends to focus around whether a Service name should be based around a noun or a verb. One reason for such debate can be highlighted by investigating a practitioner's background. See Table 5–1 for some common naming tendencies.

<table>
<thead>
<tr>
<th>Background</th>
<th>Naming Tendency</th>
</tr>
</thead>
<tbody>
<tr>
<td>OO Architect / Engineer</td>
<td>Noun</td>
</tr>
<tr>
<td>Data Analyst / DBA</td>
<td>Noun</td>
</tr>
<tr>
<td>Business Analyst / Process Engineer</td>
<td>Verb</td>
</tr>
<tr>
<td>Procedural Architect / Engineer</td>
<td>Verb</td>
</tr>
<tr>
<td>REST Engineer</td>
<td>Verb</td>
</tr>
</tbody>
</table>

See Table 5–2 for some of the pros and cons with respect to the noun versus verb debate when focusing on Business Activity Services.

<table>
<thead>
<tr>
<th>PROS</th>
<th>VERBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical grouping of related operations may enable code sharing. (Technical Cohesion)</td>
<td>Easier to discover and define Service granularity via use of a business function model.</td>
</tr>
<tr>
<td></td>
<td>Business Functionality Cohesion</td>
</tr>
</tbody>
</table>
In reality, both noun and verb approaches are required since Service naming guidelines should cover all Service classifications and not just Business Activity Services.

5.3.1.1 Service Naming Guidelines

While it is important to define guidelines around Service naming, it is also important that an enterprise enforces a consistent approach by establishing SOA governance policies to enforce the agreed upon standards.

Some example guidelines around Service naming conventions are:

■ A Service name should convey the purpose of the capabilities belonging to the Service.

■ A Service name should clearly establish a meaning and a context in the business.

■ A Service name should not use redundant words like ‘service’.

■ A Service name should not use the name of the Service classification that it belongs to such as “Business Process”, “Business Activity”, or “Data”.

■ The first letter of a Service name should be capitalized. If the Service name includes two words, then the second word’s first letter should also be capitalized.

■ A Service name should not contain any spaces or special characters.

■ Each Service classification has specific guidelines. See Table 5–3 for some example guidelines.

### Table 5–3 Example Service Naming Guidelines

<table>
<thead>
<tr>
<th>Service Classification</th>
<th>Guideline</th>
<th>Description</th>
<th>Example(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presentation</td>
<td>Verb+Noun</td>
<td>Use the purpose of the portlet and the information/business entity being displayed.</td>
<td>LookupCustomer</td>
</tr>
<tr>
<td>Business Process</td>
<td>Verb+Noun</td>
<td>Use the business function model to dictate the name of a Business Process Service.</td>
<td>ManageCustomerSale</td>
</tr>
<tr>
<td>Business Activity</td>
<td>Verb+Noun</td>
<td>Use the business function model to dictate the name of a Business Activity Service.</td>
<td>InvoiceCustomer</td>
</tr>
</tbody>
</table>
5.3.1.2 Service Operation Naming

As well as having guidelines for naming Services, it is important to include guidelines around naming Service operations.

- A Service name establishes a meaning and a context in the business. Therefore the Service Operation name should not duplicate words from their Service name.
- A Service operation name consists of a Verb + Noun combination. Using the business function model can guide the Service operation name.
- Standardize on operation verbs across all Service classifications.

### Table 5–3 (Cont.) Example Service Naming Guidelines

<table>
<thead>
<tr>
<th>Service Classification</th>
<th>Guideline</th>
<th>Description</th>
<th>Example(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Noun</td>
<td>Represents a business entity and should be predetermined by the name that the business would understand. Noun based preferable but when necessary can be prefixed with a verb or adjective but only if it makes sense to the business.</td>
<td>(Authorized)Customer</td>
</tr>
<tr>
<td>Connectivity</td>
<td>(Verb+)Noun</td>
<td>Use the name of the foreign source that the business entity represents.</td>
<td>Finance</td>
</tr>
<tr>
<td>Utility</td>
<td>Verb(+Noun)</td>
<td>Verb based, but in the event that clarification or differentiation is needed between two or more utility services that perform the same type of action, and therefore need to use the same verb name, adding a noun after the verb can be used.</td>
<td>Notify</td>
</tr>
</tbody>
</table>
Further Reading

The *IT Strategies From Oracle* series contains a number of documents that offer insight and guidance on many aspects of technology. In particular, the following documents pertaining to *Identifying and Discovering Services* may be of interest:

*Software Engineering in an SOA Environment* - Provides an approach to delivering projects within an SOA environment. It identifies the unique software engineering challenges faced by enterprises adopting SOA and provides a framework to remove the hurdles and improve the efficiency of the SOA initiative.

*Creating an SOA Roadmap* - Provides a repeatable process for constructing an SOA Roadmap. The process described follows the standard four steps used within the industry to create roadmaps i.e. establish the current state, define the future vision, analyze the gap, and define the phases and schedule of the roadmap. It is the particulars within each phase of the overall process that provide the uniqueness and value of the approach described in this document.

*A Framework for SOA Governance* - Provides a description of an SOA governance framework which eases the transition and on-going execution of an enterprise’s Service-Oriented Architecture (SOA) by providing a means to reduce risk, maintain business alignment, and show the business value of SOA investments.

*ORA SOA Foundation* - Provides a description of the foundational aspects of SOA in support of the broader Oracle Reference Architecture. This document is intended to provide historical, as well as current, context for SOA so the reader will understand SOA fundamentals underpinning the ORA.

Refer to the *ORA Glossary* document for descriptions of key terms.