Oracle® Enterprise Repository

Metrics Configuration Guide

10*g* Release 3 (10.3)

July 2009

ORACLE®

Oracle Enterprise Repository Metrics Configuration Guide, 10*g* Release 3 (10.3)

Primary Author: Vimmika Dinesh

Contributing Author: Jeff Schieli, Sharon Fay

Oracle Enterprise Repository

Metrics Configuration Guide

Table of Contents

Introduction

This guide addresses metrics enhancements available in Oracle Enterprise Repository. Information in this guide will assist those who wish to add metrics fields to asset types, which do not already have them. The metrics fields provided by this guide are, **Total Development Hours (TDH**), **Production Investment (Pinv)**, **Consumption Factor (Cfac)**, **Predicted Number of Annual Reuse Opportunities (n)**, and **Hourly Burden Rate (B)**. These fields most commonly appear on the **Metrics** tab in the Asset Editor.

The upgrade process described in this document requires the participation of your site's DBA, system administrator, and a registrar.

Metrics Elements

Overview

This document explains how to add metrics elements to a Type. These metrics are used to estimate the value of each asset and to determine the ROI of the asset portfolio.

A total of five metrics elements must be added, and the **Development Hours** element must already exist in the Type. If it does not, or if it has been deleted, it, too, must be added. The necessary changes are made in the **Type Manager** and the database; instructions for both are included here. The new metrics elements are listed below:

- **Total Development Hours (TDH)** - the total development effort expended to create this asset.
- **Production Investment (Pinv)** - the extra time or effort involved in building or harvesting the asset, and in packaging and documenting the asset to make it reusable. This value is represented as a percentage over the time initially required to build the asset for one-time use.
- **Consumption Factor (Cfac)** - the time an asset consumer spends locating, evaluating, and using an asset, represented as a percentage of the time necessary to build the asset from scratch.
- **Predicted Number of Annual Reuse Opportunities (n)** - indicates the number of times the asset is expected to be used over the course of a year
- **Hourly Burden Rate (B)** - the hourly overhead cost of the individual that produced the asset.

The metrics should be included in the ten base Asset Types, in addition to the other asset types that have been added to your system:

- Application
- Business Process
- Communication Adapter
- Component

- Environment
- Framework
- Pattern
- Process
- Service
- XML Schema

Adding Metrics

This procedure is performed in the **Asset Editor**.

Use this text to cut and paste element names:

- Total development hours (TDH)
- Production investment (Pinv)
- Consumption Factor (Cfac)
- Predicted number of annual reuse opportunities (n)
- Hourly burden rate (B)

- **Note**
  - The `ValidateMetrics.sql` file must be run against the database.

1. Select **Manage Types** in the **Actions** menu.

   The **Type Manager** launches.

2. Select the Type to which the metrics are to be added from the **Type Manager** sidebar.

3. Select the **Editor** view.

4. Add a tab called **Metrics**. It should appear above the **Miscellaneous** tab.

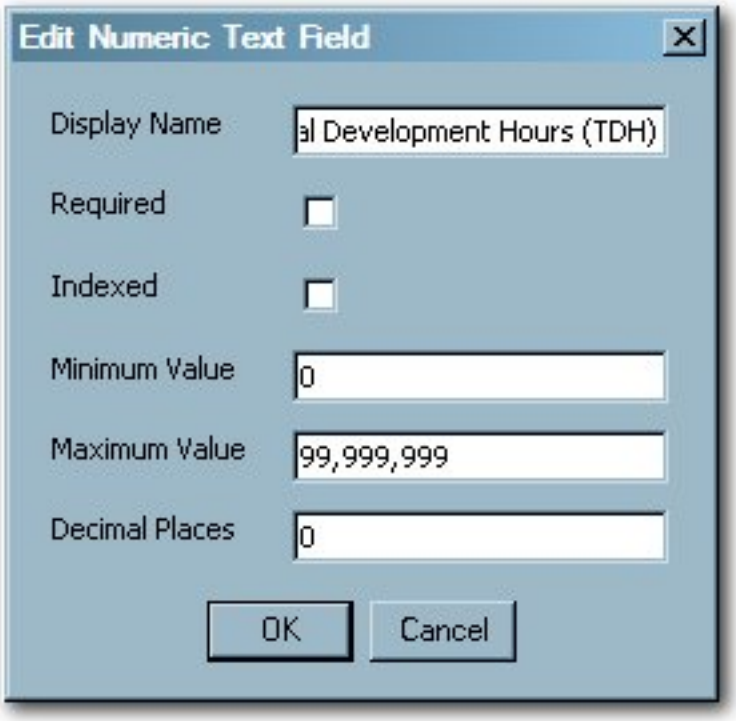
5. Click **Add** in the **Elements** section.

6. Select **Numeric Text Field** from the drop-down.

7. Click **OK**.

8. In the **Edit Text Field** box, copy and paste the element name in the **Display Name** box.

9. Change the **Decimal Places** value to **0**. Only the **Consumption Factor** element has two decimal places.



10. Click **OK**.

11. Repeat steps 4 - 9 for each metric.

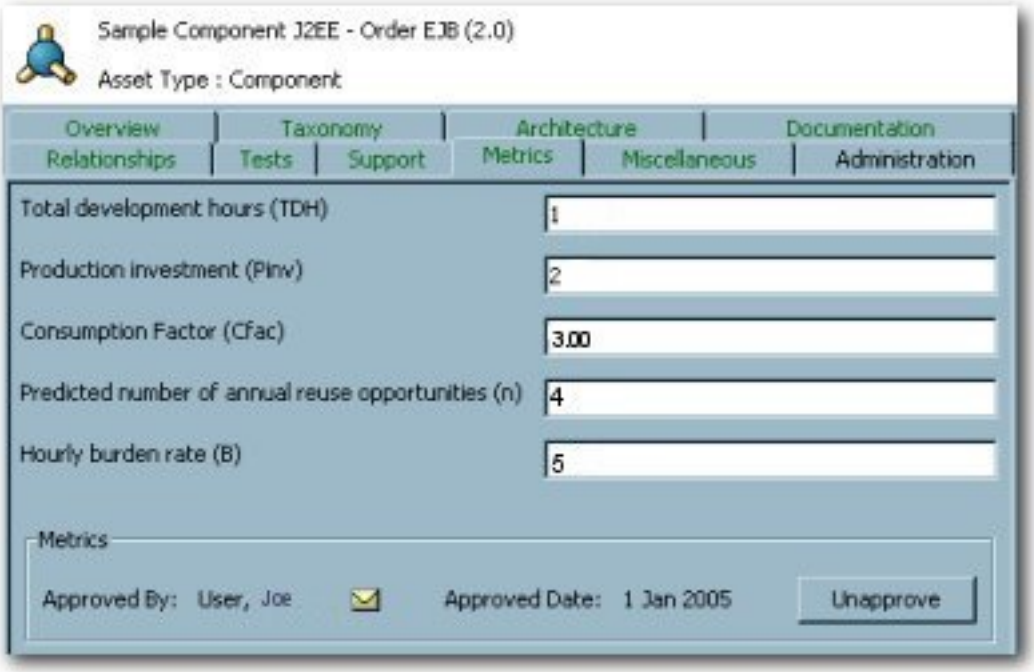
12. When all metrics have been added to the **Type**, click **Save**.
    - Do not add the elements to the **Viewer** tab at this time.

13. Repeat the process for each of the base asset types.

Validate the Elements on the Asset Type

1. Select an asset belonging to one of asset types just modified. The easiest way to find such an asset is to right-click on the **Registered** folder in the **Asset Editor** and sort by asset type.

2. Add a unique value to each metric element on the asset. Recommended values are 1, 2, 3, 4, and 5.



3. Determine the asset name, version and asset type name from the status bar on the edited asset.

4. Save the asset.

Running Queries Against the Database

The following metrics queries may be run against the database by following the instructions provided **below**.

```
-- Metrics SQL (for Oracle, UDB and MsSQL Server)
-- Total Development Hours
select a.name as AssetName, a.version as Version, at.name as
AssetTypeName, axi.stringvalue as tdh
from assetxmlindex axi, assets a, assettypes at
where axi.assetid = a.id and axi.assettypeid = at.id
and axi.fieldid in
        (select id from assetxmlindexmappings where fieldname =
'/custom-data/total-development-hours--tdh-')
order by a.name, a.version;
```

```
-- Production Investment
select a.name as AssetName, a.version as Version, at.name as
AssetTypeName, axi.stringvalue as Pinv
from assetxmlindex axi, assets a, assettypes at
where axi.assetid = a.id and axi.assettypeid = at.id
and axi.fieldid in
	(select id from assetxmlindexmappings where fieldname =
'/custom-data/production-investment--pinv-')
order by a.name, a.version;

-- Consumption Factor
select a.name as AssetName, a.version as Version, at.name as
AssetTypeName, axi.stringvalue as Cfac
from assetxmlindex axi, assets a, assettypes at
where axi.assetid = a.id and axi.assettypeid = at.id
and axi.fieldid in
	(select id from assetxmlindexmappings where fieldname =
'/custom-data/consumption-factor--cfac-')
order by a.name, a.version;

-- Predicted Number of Annual Reuse Opportunities
select a.name as AssetName, a.version as Version, at.name as
AssetTypeName, axi.stringvalue as n
from assetxmlindex axi, assets a, assettypes at
where axi.assetid = a.id and axi.assettypeid = at.id
and axi.fieldid in
	(select id from assetxmlindexmappings where fieldname =
'/custom-data/predicted-number-of-annual-reuse-opportunities--n-')
order by a.name, a.version;

-- Hourly Burden Rate
select a.name as AssetName, a.version as Version, at.name as
AssetTypeName, axi.stringvalue as B
from assetxmlindex axi, assets a, assettypes at where axi.assetid =
a.id and axi.assettypeid = at.id
and axi.fieldid in
	(select id from assetxmlindexmappings where fieldname =
'/custom-data/hourly-burden-rate--b-')
order by a.name, a.version;

-- Predicted Net Hours Saved
select a.name as AssetName, a.version as Version, at.name as
```

```
AssetTypeName, axi.stringvalue as PNHS
from assetxmlindex axi, assets a, assettypes at
where axi.assetid = a.id and axi.assettypeid = at.id
and axi.fieldid in
        (select id from assetxmlindexmappings where fieldname =
'/custom-data/development-hours')
order by a.name, a.version;

-- End Metrics SQL
```

dbqueries

1. Open a connection to the database and run the six queries listed above. The asset name, version, asset type, and metric should be returned as the result. If a metric is not returned, it is likely that the metric name was not copied and pasted accurately. Each metric has its own query; it is recommended that these queries be run separately. Any metric that returns the expected results is validated.

2. If any metric query fails, remove the element and follow the steps to create the metric element, validate it, and run it against the database. If the PNHS element is missing from the selected asset type, it must be added, as described below:
    1. Add the string **Development Hours**
    2. Click **OK**.
    3. Edit the **Development Hours** element and change the name to **Predicted Net Hours Saved (PNHS)**. (The element name in the metrics is based on the **Development Hours** programmatic name.) The **Predicted Net Hours Saved** label must appear in the **Asset Editor**.

3. After all metrics are validated, open the asset type in the Oracle Enterprise Repository **Type Manager**.

4. Select the **Viewer** tab.
    - **Note:**
        - All added elements should appear in the **Hidden Elements** area.
        - Only the **Predicted Net Hours Saved (PNHS)** metric will be meaningful to asset consumers.
        - It is recommended that **Predicted Net Hours Saved** (PNHS) appears in the first column in the **Overview** group.

5. Save the asset type.

6. Repeat steps 1 - 5 to add metrics elements to other asset types.

7. Remove any sample data that may have been added to test the queries, and save your data.

Overview

IT organizations have at their disposal software assets that, if systematically reused in the transformation to and evolution of a Service-Oriented Architecture (SOA), can potentially save hundreds of thousands—even millions—of dollars each year by increasing development productivity and decreasing software maintenance costs. The resulting performance improvements can have a significant impact on business agility and the ability to respond quickly to competitive challenges.

The transformation to SOA involves deconstructing monolithic applications into a matrix of discreet, loosely coupled, highly reusable services. The governance measures in place to manage that transformation and ensure the maximum reuse of services and related artifacts must include the means to assess, track, and communicate the value of that reuse. This provides the feedback critical to determining whether SOA is on track to meet organizational goals. That information then guides investment and other decisions about the evolution of SOA.

An organization’s portfolio of software assets can include services, business processes, and other supporting artifacts that comprise SOA. In order to gauge the return on investment for the reuse of services and other assets, an organization must assess the potential value of key assets within its portfolio, continuously measure the impact of reuse on development productivity, and determine the ultimate impact on its bottom line.

This guide focuses on the stages of the SOA lifecycle during which services are designed and created. It provides an approach for estimating the value of the various software assets contained in a typical portfolio. This approach can be used by IT managers, enterprise architects, and other stakeholders in building the business case for, and determining the business impact of, Service-Oriented Architecture. It should be noted, however, that any calculation of an SOA’s overall value must also include metrics that cover the other stages of the SOA lifecycle, including individual service performance in the operational environment. Such measurement is beyond the scope of this paper. The asset valuation process should expand incrementally, beginning with the simpler tasks: those assets that are of the most obvious relevance and value to the SOA. For example, common security components or customer information services may already be in use

across multiple applications and projects. The asset valuation process should start with these services and then expand to encompass other assets, providing an increasingly comprehensive and accurate picture of the asset portfolio's overall value.

The process and calculations described in this document distill years of experience with customers as well as recognized industry best practices. They were designed to estimate the potential reuse value of each asset and to project the expected return on investment (ROI) on an asset portfolio. On an individual basis, some organizations using the asset valuation process described in this document have recorded millions of dollars in reuse savings.

The Measuring_SOA_Success_workbook.xls will assist in establishing the value of an asset portfolio, and the following steps walk through the worksheet entries and explain the logic behind the calculations. At each step, the variables used in the calculations can be modified to reflect specific situations at any organization.

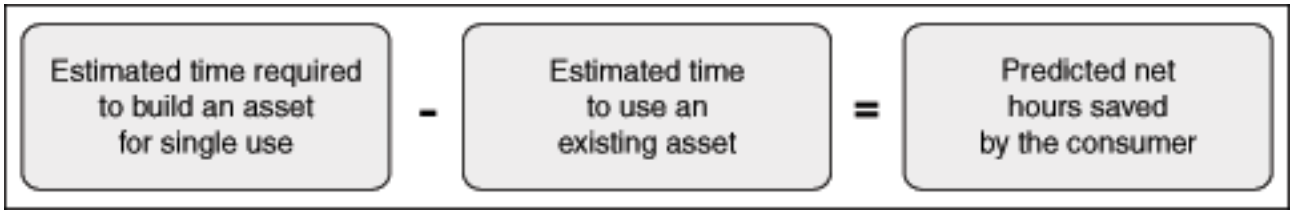
What is a reusable asset?

A reusable asset can be defined as any artifact of the software development or SOA lifecycle, including but not limited to: services (Web services, data services, etc.), business processes, architecture models, applications, frameworks, components, WSDL files, XML schemas, DTDs—quite literally any code, metadata, or supporting artifact that is relevant to the SOA. If it has value, that value can be extended through reuse—and that reuse value should be tracked and measured.

Estimating the Value of Assets in the Portfolio

An asset's estimated value is based on the development costs avoided by reusing rather than recreating it. That measurement is referred to here as **Predicted Net Hours Saved** (See **Predicted Net Hours Saved**). By reusing existing assets, an organization avoids both the costs of repeatedly developing the same functionality and the costs of maintaining different implementations of the same functionality. Reuse consolidates functionality and reduces redundancy; the more that reuse occurs, the greater the savings.

The formula used to calculate avoided costs is simple and straightforward. The first part of the equation estimates the amount of time a developer would spend creating the asset for a single use. Next, the formula deducts the time spent to reuse an existing asset on a project. The result is an estimate of the net hours saved by reusing it.



The formula then considers the level of annual usage for each asset and converts the total predicted net hours saved to a dollar value. The result is an estimate of the potential value

of existing assets in the organization’s portfolio.

Valuation Process

Following are the steps to the valuation process:

Step 1: Inventory available assets

The first step in the valuation process is to inventory available assets and organize them into categories. The assets and supporting artifacts addressed in this document fall into three basic categories:

- Services (and other assets that are exposed as interfaces)
- Black-box code assets
- White-box code assets

Services and other assets exposed through interfaces typically have higher production and overhead costs. For example, it takes time to gather and generalize requirements, parametrize the interface, and test under a broad spectrum of use cases. However, services and similar assets also offer the most significant overall reuse savings. Developers can easily access and understand a services's interface and test its

functionality, avoiding much of the design, development, and testing necessary for other types of assets. In addition, because there is a single implementation of the service's functionality, ongoing maintenance costs are greatly reduced.

The term black-box indicates that an asset's code implementation is hidden from the end-user. Unlike services, which typically have one deployed instance that is called by multiple users, black-box assets are incorporated into the developer's code base, resulting in multiple deployed instances.

Rate calculators and security frameworks are typical examples of black-box assets, which are distributed as binaries or as obfuscated objects. Clearly defined and documented interfaces serve as the communication channel between the asset and its surrounding environment. While the production and overhead costs for developing black-box assets are similar to those for services, any reuse savings are reduced by testing and maintenance costs incurred when the black-box asset is incorporated into the consuming developer's code base.

The term white-box is used to indicate that the code implementation of an asset is visible to end-users and that the source code can be modified when it is incorporated into the developer's code base. Open-source assets, such as Struts, are white-box by definition. White-box assets offer the lowest level of return in a reuse program. While they can improve development productivity, their value in terms of maintenance savings is limited because their use can result in the deployment of multiple variations of an asset, each of which must be individually supported. If a mandatory change occurs to the core functionality of a white-box asset, each deployed variation will have to be examined, modified, and retested.

Start small

Given the sheer volume of assets that may be available for reuse, the inventory process should proceed incrementally, identifying and categorizing key assets—those that are in the greatest demand and are the most frequently used. The process can then expand to encompass other assets, as resources allow.

Step 2: Enter the assets into the appropriate worksheet

Download the Measuring_SOA_Success_workbook.xls to your local machine.

In Column A of the "Services" worksheet, enter all services (or other assets that are invoked through an exposed interface).

In Column A of the "Black Box Code" worksheet, enter all black-box assets.

In Column A of the "White Box Code" worksheet, enter all white-box assets.

Step 3: Estimate the total time spent generating the asset

In Columns B through G on each of the worksheets, enter the time invested in producing each reusable asset and any supporting artifacts. This number should reflect the total development effort for each asset. For convenience, the time estimates are broken down into individual lifecycle stages.

Table 1 provides an example of the time invested in producing a reusable security service that authenticates and validates user IDs.

Table 1. Example: **Time invested in producing a reusable security service**

Business requirements and business modeling	7 weeks (280 hours)
Analysis and design	6 weeks (240 hours)
Coding and implementation	8 weeks (320 hours)
Integration and testing	6 weeks (240 hours)
Total	1,080 hours

Following the example in the table, conduct the same exercise for assets in your portfolio. The cumulative total of this measurement will be automatically calculated and will appear in Column H. Note that the numbers entered will be estimates. These estimates should be correct to an order of magnitude (measured in hours, days, weeks, months, or years) and should be conservative. For example, if the estimate ranges between three and six weeks, use the three-week figure. If an estimated value for each phase of the lifecycle is unavailable, enter an estimate of the total time invested in Column H.

Bear in mind that reuse value can accumulate quickly, especially in a well-managed reuse environment. So in keeping with the idea that the asset valuation process should start small, with the "low-hanging fruit", even a conservative, “good enough” estimate of the time invested in producing a reusable asset can result in compelling evidence of the value of reuse.

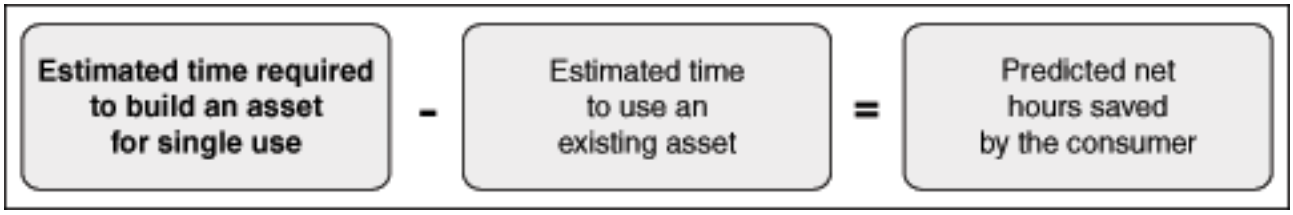
Step 4: Estimate the time required to build an asset for single use

Table 2. Production investment for reusable assets

Asset	Production Investment (P Inv)
Services	200%
Black-Box Code Assets	200%
White-Box Code Assets	100-200%

Building and/or harvesting a service or other reusable asset typically takes more time and effort than building an asset for single use. This extra time and effort is called the Production Investment (see Table 21) and it includes the time and effort involved in the development, packaging, and documentation necessary to enhance an asset’s reusability and availability across the widest possible audience. For example, a developer might invest 540 hours creating a security function for use on a project. An organization will spend approximately twice the amount of time—200%—building a reusable security service.

The reuse savings estimate for each asset is based on the time that a developer would have spent developing equivalent functionality for a single use. The Measuring_SOA_Success_workbook.xls uses the total development hours and the Production Investment values to estimate that time and effort.



For example, Table 1 indicates a total investment of 1,080 hours to produce the reusable security service. If the average production investment for a reusable service is 200 percent (that is, twice the time it takes to build the equivalent functionality in a single-use asset), a developer would have spent 540 hours building a single-use version of the security service. That figure (540 hours) provides the basis for calculating the estimated reuse savings for the security service.

Table 2 lists the Production Investment for various categories of reusable assets. Notice that in most cases the Production Investment exceeds 100 percent, reflecting the additional effort required beyond what is needed to build a single-use asset.

Given that the reuse of a white-box asset assumes some level of modification, the ease with which the consuming developer can make the necessary modifications to the asset is a significant factor in that asset's reusability. Additional documentation and other resources provided to the developer to facilitate white-box asset modification are part of the Production Investment. For white-box assets, this investment ranges from 100 percent (no additional effort is invested in improving the asset's reusability) to 200 percent. Obviously, there is a direct correlation between the effort invested in making an asset reusable and the effort required to reuse it. That reuse effort is discussed in the next section.

The values above are provided as a baseline. Few organizations actually measure the Production Investment associated with the reusable assets in their portfolios until they have established a managed reuse program. For the purposes of the accompanying workbook, these values may be used as is, or may be adjusted to reflect organizational experience. Enter the appropriate Production Investment value in cell D4 of the appropriate worksheet.

The next step is to calculate and subtract the Consumption Factor, the time that a developer will spend reusing the asset.

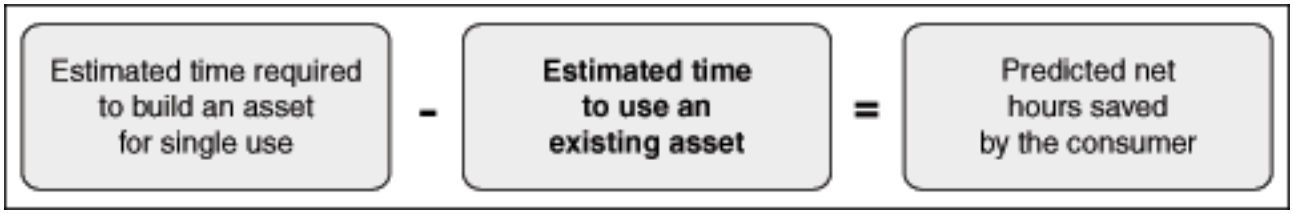
Step 5: Estimate the time to use an existing asset

Table 3. Consumption factor for different types of assets

Asset	Consumption Factor (Cfac)
Services	5%
Black-Box Code Assets	20%
Domain-specific black-box code assets	5%
White-Box Code Assets	40% (with less than 25% code modification)
White-Box Code Assets	90% (with greater than 25% code modification)

Reuse is not free. It takes time to evaluate and then reuse an asset in a project. This time is estimated using the Consumption Factor.

Table 3 displays the Consumption Factors associated with various asset categories. As previously discussed, a developer would have spent 540 hours building a single-use version of the security service in our example. The consumption factor for services (and other assets that are exposed as interfaces) is 5 percent. (In this context, consumption refers to the design-time incorporation of a service call in the development of an application.) Applying this consumption factor to the previously determined 540 hours figure produces an estimate of 27 hours to reuse the security service.



Oracle's experience confirms the findings of a number of studies with regard to the Production Investment and Consumption Factors associated with reuse. In Table 3, note that there are two entries for white-box assets. The first reflects the Consumption Factor that applies if less than 25 percent of an asset's code is modified when the asset is reused. The second entry reflects the Consumption Factor that applies when more than 25 percent of the asset's code is modified. As a rule of thumb, the Production Investment and Consumption Factor for white-box assets are inversely proportional. That is, if no effort is made during production to make an asset reusable (Production Investment 100 percent), the developer will spend more time evaluating and modifying the asset in order to reuse it, resulting in a higher Consumption Factor (Consumption Factor 90 percent).

Table 3 also includes two entries for black-box code assets and their associated artifacts. The first entry reflects the general Consumption Factor for a black-box code asset, while

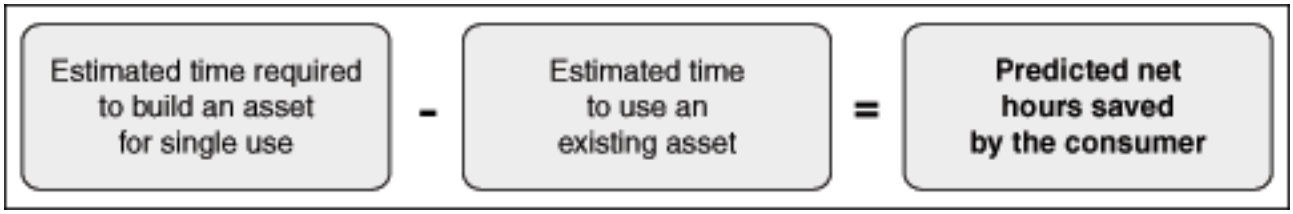
the second reflects the Consumption Factor that applies if the asset is produced and consumed within the same domain. The advantage developers have when using domain-specific reusable assets is reflected in these Consumption Factors. It takes less time and effort to understand and reuse domain-specific assets if the developer is familiar with the domain.

As with the Production Investment values described in Step 4, Consumption Factor values are provided only as a baseline and may be adjusted to reflect organizational experience. Enter the appropriate Consumption Factor in cell D5 of the appropriate worksheet.

Predicted Net Hours Saved

The difference between the time a developer would spend building an asset for single use and the time needed to use a reusable version of that asset is referred to as Predicted Net Hours Saved. This represents the development cost avoided through reuse of the asset.

The security service in our example would have taken about 540 hours to develop for single use. Based on the calculations described above, a developer will take an estimated 27 hours to reuse the security service. The difference (513 hours) is the Predicted Net Hours Saved value for the security service.



Prioritizing Investments

Steps 1–5 illustrate how to calculate an asset's estimated reuse value. Those responsible for building a business case for reuse, or for prioritizing investments in reusable services or assets, should use Steps 6–8 below to translate the reuse value into the expected ROI.

Establishing the expected ROI of assets within the portfolio requires consideration of

annual usage of each asset, and the conversion of the total Predicted Net Hours Saved to a dollar value. That conversion will be covered in Step 7.

Step 6: Estimate the level of annual usage for each asset

In any given year, most organizations have a number of development projects on the docket. It is likely, however, that only a subset of these projects will actually have an opportunity to reuse assets in the organization's portfolio. For example, of the 100 projects an organization may have funded for next year, 25 percent may deal with facilities—building and property updates. Reuse of software assets in projects of this nature may be impractical or inapplicable.

In this example, the potential for asset reuse is possible and appropriate in only 75 projects (100 minus 25 percent). This number (75) should be entered into cell D7 of the appropriate worksheet. The next step is to estimate how many times each asset will be reused. It is unlikely that every project with access to the asset portfolio will leverage every available asset. For example, perhaps only one-third of the projects that represent opportunities for reuse will be able to reuse the example security service. This translates to 25 annual reuse opportunities (one-third of 75).

Enter the usage estimate for each asset in the column labeled "# Annual Reuse Opportunities."

Step 7: Converting the total Predicted Net Hours Saved to a dollar value

The term *Burden Rate* refers to the hourly overhead cost associated with each of the various asset-consumer roles. The average Burden Rate for software developers may be different than the average Burden Rate for business analysts. Burden Rates should be standard for each organization. Enter your organization's Burden Rate in Cell D6 of the appropriate worksheet. The worksheet uses the Burden Rate to calculate the *Potential Reuse Value* for each asset.

Step 8: Calculate potential reuse value

The Projected Annual Saving Summary worksheet totals the savings from all of the worksheets. This total should be used as the basis for justifying your reuse program and as a target goal for that program.

Step 9: Tracking progress

An enterprise registry repository with the appropriate capabilities is essential in tracking progress toward organizational SOA and reuse goals. A registry repository should track each asset's value and then, as assets are used, the registry repository should gather the necessary data to generate reports that detail assets' values as used in the SOA initiative. This provides a way for organizations to compare estimated ROI to actual returns.

Summary

While reusable assets may vary in their size, scope, and purpose, their estimated valuation —and the accurate reporting of their actual value—are essential in guiding SOA governance efforts to ensure sustainable alignment with business goals.

Just as asset valuation information is vital in guiding SOA governance efforts, those efforts are essential for increasing the value of the assets by insuring their alignment with and support of policies and standards established at the architectural, IT, and corporate levels. Ultimately, an asset's value is determined not solely by its reusability or reuse, but also by the contribution it makes in moving the organization toward its business goals. To that end, the importance of effective SOA lifecycle governance in guiding the design, development, and use of assets cannot be overstated.2

While the enterprise portfolio of services and other software assets plays a vital role in an SOA, it represents one aspect of the overall value of a Service-Oriented Architecture. Given the SOA environment's unique nature, a variety of factors contribute to the ultimate ROI of SOA, including the sharing of services in the operational environment and individual service performance. A variety of methods are available to calculate these other factors' ROI.

The approach outlined in this paper, when used in conjunction with a robust enterprise registry repository, will provide an organization with the information necessary to make

informed decisions regarding the management and consolidation of the services and other software assets in the enterprise portfolio, the assignment of those assets for use/reuse in projects, and the ongoing, proactive governance of the SOA.

Oracle Registry Repository provides essential visibility, traceability, and governance for services and other assets in the enterprise portfolio. Its tracking, measurement, and reporting capabilities provide the means to assess and leverage the value of services and assets, helping organizations to ensure business and architectural alignment and measurable ROI in the transformation to and evolution of SOA.

Oracle Registry Repository is one component of the Oracle Service-Oriented Architecture governance solution. SOA success is achieved through the proper combination of people, process, and technology. Oracle Systems brings these elements together, providing the expertise, experience, and technologies to help organizations achieve their SOA goals.

For information on Oracle Registry Repository and the Oracle SOA governance solution, please visit: www.oracle.com/soa