
How Do I Manage Multiple Versions of my BI Implementation?

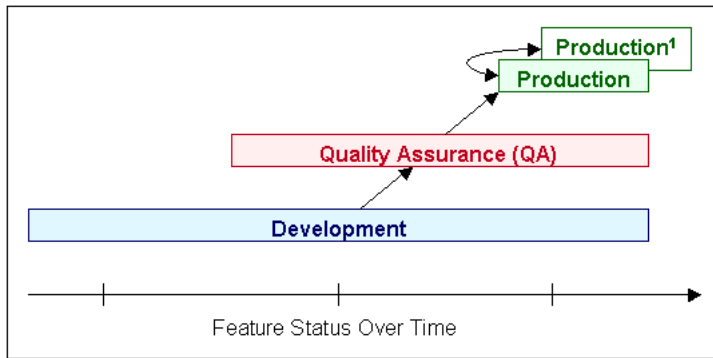
This case study focuses on the life cycle of a business intelligence system. This case study covers two approaches for managing individually changing versions of your BI system once you have implemented in production. The approach you select depends on the phase of your BI system development life cycle.

Scenario

After a period of development and testing, one company implements its BI system in production. The Production version of the system typically changes as new features are incrementally implemented from Development, and as Production bugs are discovered and fixed. At the same time, the Development version of the system continues to evolve with new functionality. This company now has several individually changing versions of the system and faces a challenge familiar to all companies, regardless of how many BI environments they maintain: how to best manage changes in different versions of the system.

One version of this common scenario is depicted in [Figure 9-1](#), where the Development environment is consistently more advanced than the functionality in Production, and QA is somewhere between the two extremes. Development changes are incrementally propagated to QA and subsequently to Production. At the same time, Production has its own cycle of changes, denoted in [Figure 9-1](#) as the shadow environment labeled 'Production¹', and used for controlled problem solving. 'Production' and 'Production¹' are at the same stage of development, and serve to illustrate the errors that occur in Production, which are fixed and implemented directly in Production, but that must somehow be merged with Development. Other companies may have fewer or more differing environments for their BI systems, but the same maintenance challenges still apply.

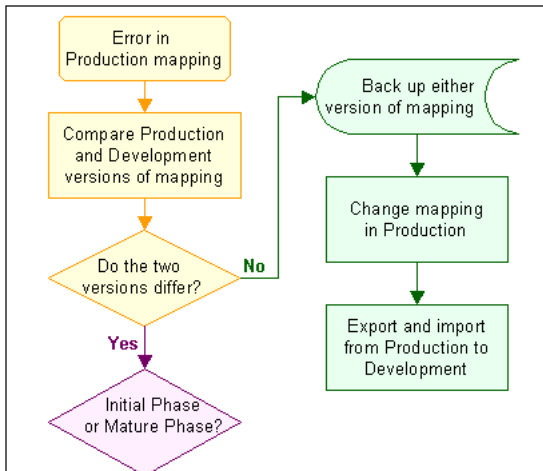
Figure 9–1 Typical Life Cycle of a Business Intelligence System



Companies may need multiple environments for their BI systems, as illustrated in [Figure 9–1](#), because they typically implement incremental changes to the system. However, some companies implement only whole projects in Production. [Figure 9–1](#) does not apply to these companies.

In this case study, a company finds a problem with a mapping in Production. The first step is to compare the Production version of the mapping with the Development version of the mapping, as illustrated in [Figure 9–2](#). If the mapping is identical in both environments, the solution is simple: make the changes in either environment and copy the mapping to override the older version. If the mapping in Production differs from its Development version, then the approach depends on whether the BI system is in its initial or mature phase.

Figure 9–2 Comparing the Production Mapping to Development



Approach

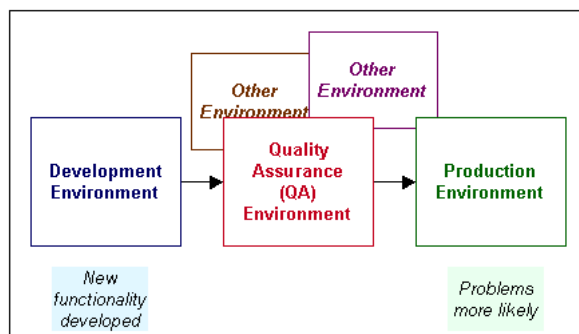
Typically, there are two phases that mark the BI system life cycle: **Initial Phase** and **Mature Phase**. The two phases present different needs and call for two different version management methodologies, each of which has benefits and drawbacks.

Initial Phase

After implementation of a business intelligence system in Production, the system is generally in its initial phase, depicted in [Figure 9–3](#). The initial phase is marked by

aggressive changes in the Development environment, coupled with errors sometimes found in Production. Because Production bugs are more likely in this mode, consider a management methodology that facilitates quick updates to each environment.

Figure 9–3 Initial Phase: Changes in Production More Likely



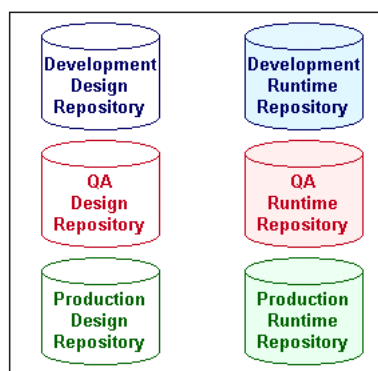
Companies often have two to five different environments. For the initial phase, this company keeps a separate definition of the metadata in each different environment (in this case, Development, QA, and Production). To propagate a change from Production, they export only the portions of the system that have changed and import them into the Development definition.

Case Study

The company has recently implemented its BI system in production, and the system is still in its initial phase, where many additional features are yet to be tested and rolled out. The production system is fairly new, and therefore the occurrence of problems is higher in this phase.

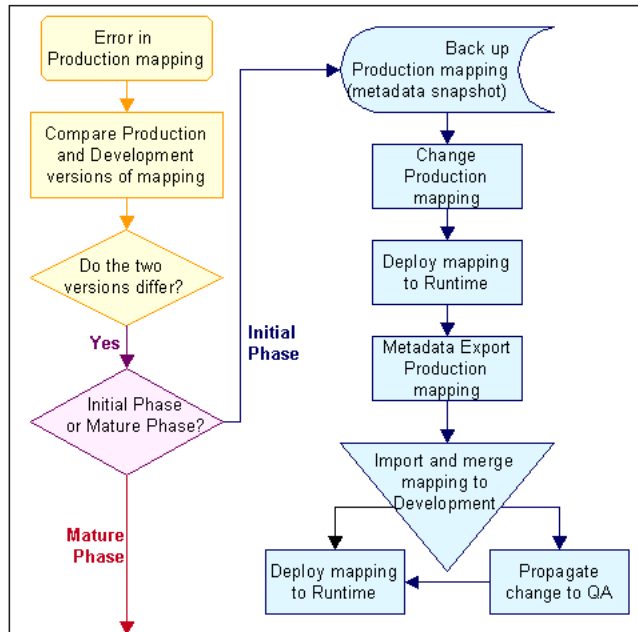
The company decides to keep a separate design repository—or definition of the system design—for each environment, as depicted in [Figure 9–4](#). In addition, they implement their processes into a separate runtime repository for each environment.

Figure 9–4 Initial Phase: Separate Design Repositories



In this example, an error occurs in a Production mapping. The company changes the mapping in Production, then exports its definition, and merges it into Development, as illustrated in [Figure 9–5](#).

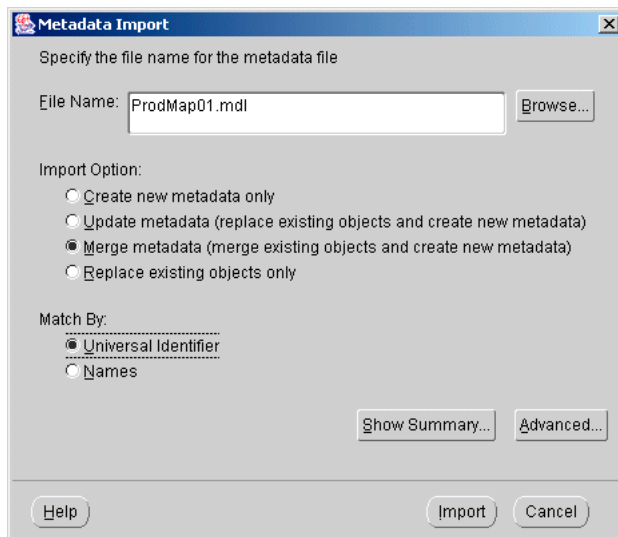
Figure 9–5 Initial Phase: Propagate Changes from Production to Development



To correct an error found in a Production mapping during the initial phase:

1. For backup, capture the definition of any mapping before modifying it.
Create a full metadata snapshot of the mapping in the Production Design Repository. Do the same with the Development and QA versions of the mapping. Because you can only restore objects from full snapshots, a full snapshot is essential when you create a backup.
2. Correct the mapping in the Production Design Repository and deploy it to the Production Runtime Repository.
This results in a changed version of the mapping that must be propagated to other environments.
3. Use Metadata Export to export only the changed mapping from Production.
Consult the *Oracle Warehouse Builder User's Guide* for instructions on using Metadata Export.
4. Use Metadata Import to import and merge the change to Development and QA, as shown in [Figure 9–6](#).
 - From the Metadata Import dialog Import Options, select **Merge metadata**.
 - From the Metadata Import dialog Match By options, select the **Universal Identifier** option.
 Matching objects by Universal Identifier is important when maintaining multiple individually changing environments.

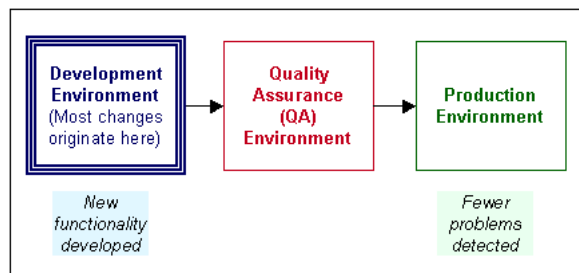
Consult the *Oracle Warehouse Builder User's Guide* for more information on using Metadata Import.

Figure 9–6 Metadata Import Options

Merging the change into Development and QA can vary in complexity depending on the changed object. If the change in the mapping in this example consists of increasing the column width of a table, the merge is simple. A merge can be more complicated and time-consuming if, for example, join criteria are changed, and other dependencies exist.

Mature Phase

The second is the mature phase, depicted in [Figure 9–7](#). The mature phase is marked by continued changes in the Development environment, but a decrease in changes required in Production.

Figure 9–7 Mature Phase: Fewer Changes in Production

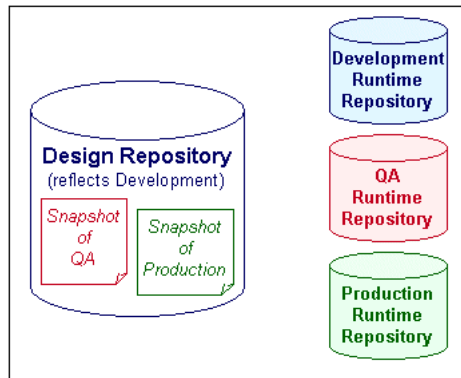
For this mode, the company chooses a methodology that saves space and administration costs: they maintain only one active definition of the BI system design, and this definition reflects the development state of the system. The company stores the design definitions of the QA and Production environments in backup, and extracts and restores changed portions of these systems when required.

Case Study

At this stage, the company's BI system has stabilized and is now in its mature phase. Some additional functionality is still being developed in the Development environment, but fixes originating in Production are rare.

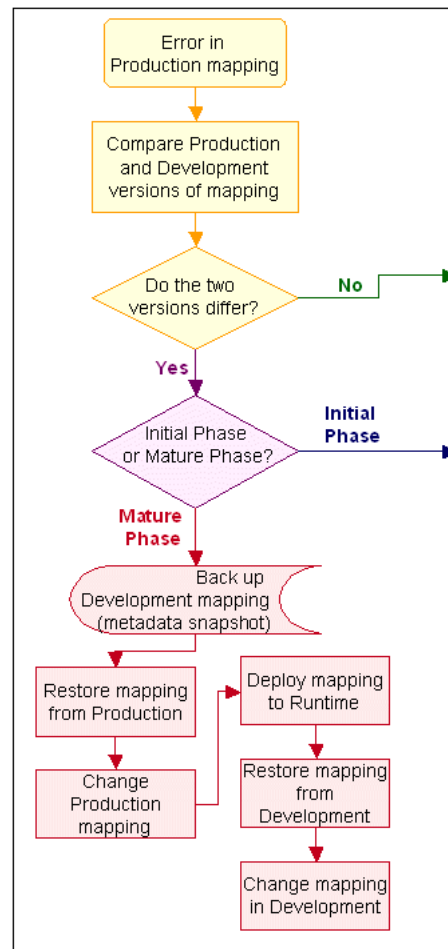
Although they continue to implement their processes into a separate runtime repository for each environment, the company decides to keep only one design repository, as depicted in [Figure 9-8](#).

Figure 9-8 Mature Phase: One Design Repository Reflecting Development



The one design repository reflects the Development environment, because it is the one active environment that regularly originates design changes. The design repositories from the QA and Production environments are stored as metadata snapshots inside the Development Design Repository. Snapshots are a backup mechanism that consumes minimal space, and still provides access to any objects that you need to restore. Because design changes rarely originate in Production or QA, storing those definitions in snapshots makes sense.

Although it is more rare during the mature phase, errors still occur in the Production environment. In this example, an error occurs in a Production mapping. The company changes the mapping in Production, then restores its definition from a snapshot in Development and makes the same change there, as illustrated in [Figure 9-9](#).

Figure 9–9 Mature Phase: Propagate Changes from Production to Development**To correct an error found in a Production mapping during the mature phase:**

1. Compare the Production version of the mapping in your Production snapshot to the Development version of the same mapping in your Design Repository.
 - If the two differ, the company follows the rest of the steps in this procedure.
 - If the two are identical, they correct the mapping as in Step 8, then deploy it to their Design and Production Runtime Repositories, and then update their Production snapshot with the changed mapping.

Consult the *Oracle Warehouse Builder User's Guide* for instructions on comparing snapshots to objects, deploying, and on updating snapshots.

2. Back up the Development version of the mapping by creating a full metadata snapshot of it.

The Development version of the mapping may differ from the Production version if developers have been working on a new iteration of that mapping. This step preserves their work. Creating a full snapshot is essential, because you can only restore from a full snapshot.

3. Restore the mapping in question from the Production snapshot.

This mapping should be identical to the one running in Production.

Consult the *Oracle Warehouse Builder User's Guide* for instructions on restoring objects from metadata snapshots.

4. Correct the mapping that you have restored from the Production snapshot.
5. Deploy the corrected mapping to the Production Runtime Repository.
6. Remove the existing definition of the mapping from the snapshot of the Production Design Repository and update the snapshot with the new version of the mapping.
7. Restore the mapping from the full snapshot you took as a backup in Step 2.

This is the mapping from the Development Design Repository. Typically, this mapping has had other work done to it as part of development of new features.

Optionally repeat this same step for QA.

8. Make the same correction to this Development version of the mapping that you made in Step 4 to the Production version of the mapping.

The cost of this methodology is that every change has to be made at least twice, in the Production and Development versions of the object. The company uses this methodology only because the mature phase does not require frequent changes originating in Production. The benefits of this approach are the minimal administration costs and reduced space requirements on the database.