Understanding change concept

Change is a difference, found between the ancestor and a contributor. Every change can be accepted or rejected, as well as have dependent changes. Changes can conflict with each other or be equivalent.

Merging begins with building a composite change tree, which consists of a model, diagramming, and non-model changes.

Change types

Read the following definitions to get familiar with different change types.

Addition change
If an element has been added to a contributor, an addition change occurs.

Deletion change
If an element has been removed from a contributor, a deletion change occurs.

Modification change
If an element property in a contributor has been modified, a modification change occurs.

If the **IsAbstract** property value of a class in the ancestor had the default value *false* and the same property value in a contributor has been changed to *true*, a modification change occurs.

There are three types of modification changes:

- **Addition modification change** that occurs when a value is added to a property.
- **Deletion modification change** that occurs when a value is removed from a property.
- **Replacement modification change** that occurs when one value is replaced with another. This type of modification change occurs only for properties that have multiplicity less or equal to 1.

Movement change

If an element owner has been changed in a contributor, a movement change occurs.

Let's say package A contains some class in the ancestor and package B contains the same class in a contributor. This means that the class has been moved from package A to package B in the contributor. This case is recognized as a movement change.

Another case of the movement change is when an attribute or an operation that has been owned by class A in the ancestor, becomes the attribute or an operation of class B in a contributor.

Order change
If the order of elements has been changed in a contributor, an order change occurs. Order changes can occur on elements such as attributes, operations, and other ordered elements. Even if a single element in a collection has changed its place, the order change is applied to the entire collection.

Since an element can have several ordered collections, several order changes can occur on a single element.

Let's say class A has attributed a, b, and c in the ancestor. The attribute c has been moved up and placed above attribute a in a contributor. This means that the order of attribute collection in class A has changed in the contributor. This is a case of the order change.

Turn off order change detection

This is the optional phase of the merge procedure. You may not need to perform it.

Order changes can be skipped while merging. For this, you need to specify names of properties wherein order changes should not be detected.

To turn off the order changes detection in specificproperties

---

1. Fromthe **Options** menu select **Environment**. The **Environment Options** dialog opens.
2. Find the Do Not Detect Order Changes for the option under the **Merge** category in the **General** options group.
3. In the option value cell, specify names of properties wherein order changes should not be detected while merging.

Property names must be written in camel case, for example, ownedAttribute, ownedElement, and so on.

Change states

Every change, whether it is addition, modification, deletion, movement, or order change, can be either accepted or rejected. It is a change state.

All accepted changes will be incorporated into the target. Alternatively, they can be rejected and excluded from the target.

Dependent changes

In some cases, other changes have to be accepted or rejected before accepting or rejecting a selected change. In other words, a selected change sometimes depends on other changes and is called a dependent change.

For a better understanding of the concept of dependent changes, study the following examples.

Let's say a class attribute type has been changed to a type that had been created by another change. In consequence, the attribute type change depends on the change that has created the type. This means that type creation change must be accepted before accepting type modification change.

Let's suppose there is an attribute type change in a contributor. An old type has been deleted and a new type has been added to the contributor. In this case, three changes occur:

- deletion change (for the old type)
- addition change (for the new type)
- modification change (for the property type)

The modification change depends on the addition change, and the deletion change depends on the modification change. Thus accepting the deletion change means also accepting the addition change, the modification change and the deletion change itself.

Conflicting changes

Conflict occurs when two changes are incompatible, i.e., the changes cannot be accepted together. Conflicts can only occur when using 3-way merge.

The following situations will result in a conflict:

- Each contributor has changed a class name or any other element property value.
- One contributor has added an operation to a class and the other contributor deleted the class.
- One contributor has moved a class into one package while the other contributor moved it to another package.

Equivalent changes

Equivalent change is a pair of identical changes that are detected in both source and target when performing a 3-way merge.

Let's say package A contains some class in the ancestor and both contributors contain the same class in package B. This means that the class has been moved from package A to package B in both source and target. This is a case of the equivalent change.

Another case of equivalent changes is when the same element A becomes renamed to B in both contributors.

Related pages

- Understanding merge types
- Preparing for merge
- Starting Model Merge
- Analyzing and managing merge results
- Finishing and canceling merge