Purpose

This section introduces the syntax used message locations when collection element is used in a validation rule

Desription

Message location supports added logic, which is especially great when dealing with collections; we are able to give the error message only for the cases where invalid XML content was found and correction has to be made.

Usage in short is: when collection is used, query will have to be a negation of the validation rule. In other words, if a validation rule limits the values of elements, we want to give the query for every element which does not fill the validation rule.

We shall be doing the following here:

1. Adding a query to a rule which uses collection
2. Testing

Instructions

1: Adding a query to a rule which uses collection

The rule we made in the previous phase should look like this:

self.Unstructured->forAll(q | q.matches("Ustrd1") or q.matches("Ustrd2"))

The message location where we want to tell about the invalid content given in XML should be everything which does not fill the OCL rule above, i.e. the query should be a negation of the above.

There are three steps involved here:

1. Replacing forAll() with select()
2. include everything in right-hand-side with operation not()
3. Replacing or with and

Reasons for these changes are:

1. In a query, we do not want to limit the content anymore; it is already limited with an OCL rule. Instead, we want to find content. Therefore select() is the method we want to use
2. not() is an OCL operation which returns the reverse of the content within it. if true is returned otherwise, false will be returned. And vice versa
3. in a rule we want to use or as both of the values are valid to be used. In a query, we know that validation rule has been triggered and there is at least one occurrence within XML which does not pass the rule. We want to point to every case which does not fill the rule. Every case which does not return true will contain something other than "Ustrd1" and "Ustrd2" Therefore operand "and" fills the purpose and operand or would lead to incorrect behaviour.

Doing the above will give us a guery:

```
self.Unstructured->select(q | not(q.matches("Ustrd1") and q.matches("Ustrd2")))
```

2: Testing

In the testing phase, it's a good idea to include every possible scenario within the same XML file to save time. We should cover the cases where valid values are given along with invalid values.