Extension methods

Extension methods are static methods that can be invoked using instance method syntax. In effect, extension methods make it possible to extend existing types and constructed types with additional methods.

Extension methods are declared by specifying the keyword **this** as a modifier on the first parameter of the methods. Extension methods can only be declared in non-generic, non-nested static classes.

The following is an example of a static class that declares two extension methods:

```
public static class Extensions
{
        public static int ToInt32(this string s) {
                return Int32.Parse(s);
        }
        public static T[ ] Slice<T>(this T[ ] source, int index, int count) {
                if (index < 0 || count < 0 || source.Length – index < count)
                        throw new ArgumentException();
                T[] result =newT[count];
                Array.Copy(source, index, result, 0, count);
                return result;
        }
}
```

It becomes possible to invoke the extension methods in the static class **Extensions** using instance method syntax:

Instance method syntax

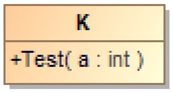
```
string s = "1234";

int i = s.ToInt32();// Same as Extensions.ToInt32(s)

int[] digits={0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

int[] a = digits.Slice(4, 3);// Same as Extensions.Slice(digits, 4, 3)
```

Example

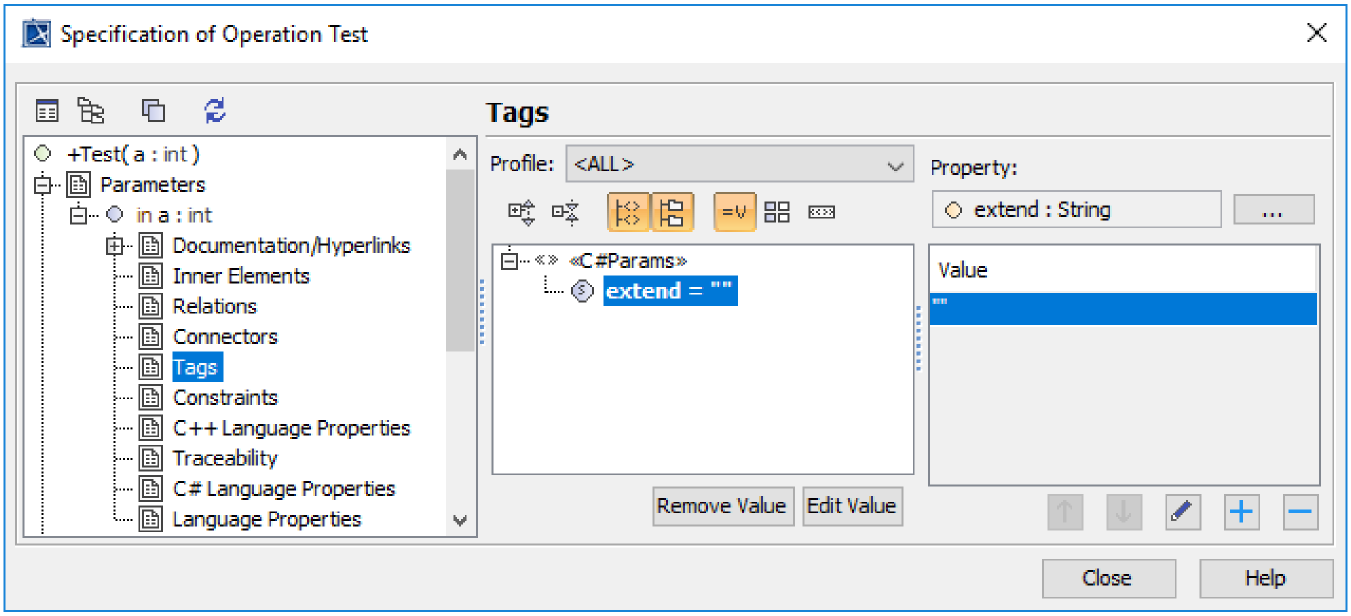
Code:

```
class K
{
                Test (this int a)
                {}
}
```

Reversed UML model:



The value is created to Tag *extend* in «C#Params»:



Specification of Operation Test

```
+Test( a : int )
Parameters
  in a : int
    Documentation/Hyperlinks
    Inner Elements
    Relations
    Connectors
    Tags
    Constraints
    C++ Language Properties
    Traceability
    C# Language Properties
    Language Properties
```

Tags

Profile: <ALL>

Property: extend : String

```
«C#Params»
  extend = ""
```

Value
""