Langauge Definitions Demo

arcanefoam

October 26, 2017

Chapter 1

Eclipse OCL

1.1 Invariant - Examples

```
context Meeting inv: self.end > self.start

-- "self" always refers to the object identifier from which the constraint is evaluated.
context Meeting inv: end > : end > start

-- Names can be given to the constraint
context Meeting inv startEndConstraint:
self.end > self.start
```

1.2 Precondition - Examples

```
context Meeting::shift(d:Integer)
pre: self.isConfirmed = false

context Meeting::shift(d:Integer)
pre: d>0 pre: d>0

context Meeting::shift(d:Integer)
pre: self.isConfirmed = false and d>0
```

1.3 Postcondition - Examples

```
context Meeting::duration():Integer
post: result = self.end     self.start
-- keyword result refers to the result of the operation

context Meeting::confirm()
post: self.isConfirmed = true
```

1.4 Examples for Collection Operations

```
context Teammeeting
inv: participants->forAll(team=self.for)

context Meeting inv: oclIsTypeOf(Teammeeting)
implies participants->forAll(team=self.for)

context Teammember::numMeeting():Integer
post: result=meetings->size()
context Teammember::numConfMeeting():Integer context
    Teammember::numConfMeeting():Integer
post: result=meetings->select(isConfirmed)->size()
```

Chapter 2

OClinECore

Taken from the OCLinECore example page

```
import ecore : 'http://www.eclipse.org/emf/2002/Ecore#/';

package tutorial : tut = 'http://www.eclipse.org/mdt/ocl/oclinecore/tutorial'
{
    class Library
    {
        attribute name : String;
        property books#_'library' : Book[*] { composes };
        property loans : Loan[*] { composes };
        property members#_'library' : Member[*] { composes };
    }

    class Book
    {
        invariant SufficientCopies:
            library.loans->select(book=self)->size() <= copies;
        attribute name : String;
        attribute copies : Integer;
        property _'library'#books : Library;
        property loans : Loan[*] { derived, volatile }
        {
            derivation: library.loans->select(book=self);
        }
        property _'library'#books : Library[?];
        operation isAvailable() : Boolean[?]
        {
            body: loans->size() < copies;
        }
    }

    class Member
    {
        attribute name : String;
        property _'library'#members : Library;
    }

    class Loan
    {
        property book : Book;
        property member : Member;
        attribute date : ecore::EDate;
    }
}
```

Chapter 3

QVT-R

Taken from the QVT Specification

Chapter 4

QVT-C

Taken from the QVT Specification

Chapter 5

EOL

Taken from the Epsilon Book

```
1.add1().add2().println();

operation Integer add1() : Integer {
    return self + 1;
}

operation Integer add2() : Integer {
    return self + 2;
}
```

```
"1".test();
1.test();

operation String test() {
    (self + " is a string").println();
}

operation Integer test() {
    (self + "is an integer").println();
}
```

Chapter 6

EVL

Taken from the Epsilon Book

```
context Singleton {

    guard : self .stereotype->exists(s | s.name = "singleton")

    constraint DefinesGetInstance {
        check : self .getGetInstanceOperation().isDefined()

        message : "Singleton " + self .name + " must define a getInstance() operation"
        fix {
            title : "Add a getInstance() operation to " + self .name
            do {
                // Create the getInstance operation
                var op : new Operation;
                op.name = "getInstance";
                op.owner = self;
                op.ownerScope = ScopeKind#sk_classifier;
                // Create the return parameter
                var returnParameter : new Parameter;
                returnParameter.type = self;
                op.parameter = Sequence{returnParameter};
                returnParameter.kind = ParameterDirectionKind#pdk_return;
            }
        }
    }
}
```

Chapter 7

ETL

Taken from the Epsilon Book

```
rule Tree2Node
    transform t : Tree!Tree
    to n : Graph!Node {
    n.label = t.label;
    if (t.parent.isDefined()) {
        var edge = new Graph!Edge;
        edge.source = n;
        edge.target = t.parent.equivalent();
        edge.target ::= t.parent;
    }
}
```

```
rule Tree2Node
    transform t : Tree!Tree
    to n : Graph!Node {

        guard : UserInput.confirm ("Transform tree " + t.label + "?", true)

        n.label = t.label;
        var target : Graph!Node ::= t.parent;
        if (target.isDefined()) {
            var edge = new Graph!Edge;
            edge.source = n;
            edge.target = target;
        }
    }
```

Chapter 8

EWL

Taken from the Epsilon Book

```
wizard ClassToSingleton {
    // The wizard applies when a class is selected
    guard : self.isTypeOf(Class)

    title : "Convert " + self.name + " to a singleton"

    do {
        // Create the getInstance() operation
        var gi : new Operation;
        gi.owner = self;
        gi.name = "getInstance";
        gi.visibility = VisibilityKind#vk_public;
        gi.ownerScope = ScopeKind#sk_classifier;
        // Create the return parameter of the operation
        var ret : new Parameter;
        ret.type = self;
        ret.kind = ParameterDirectionKind#pdk_return;
        gi.parameter = Sequence{ret};

        // Create the instance field
        var ins : new Attribute;
        ins.name = "instance";
        ins.type = self;
        ins.visibility = VisibilityKind#vk_private;
        ins.ownerScope = ScopeKind#sk_classifier;
        ins.owner = self;

        // Attach the <<singleton>> stereotype
        self.attachStereotype("singleton");
    }
}


// Attaches a stereotype with the specified name
// to the Model Element on which it is invoked
operation ModelElement attachStereotype(name : String) {
    var stereotype : Stereotype;

    // Try to find an existing stereotype with this name
    stereotype = Stereotype.allInstances.selectOne(s|s.name = name);

    // If there is no existing stereotype
    // with that name, create one
    if (not stereotype.isDefined()){
        \makeatletter
        \lstnewenvironment{etl}[1][]
        {\lstset{style=etl,
                #1}%
            \csname\@lst @SetFirstNumber\endcsname}
        {\csname\@lst @SaveFirstNumber\endcsname}
        \makeatotherotype.namespace = self.namespace;
    }

    // Attach the stereotype to the model element
    self.stereotype.add(stereotype);
}
```

Chapter 9

EGL

Taken from the Epsilon Book

```
[% for (i in Sequence{1..5}) { %]
i is [%=i%]
[% } %]

[% for (c in Class.all) { %]
[%=c.name%]
[% } %]
```

```
[% c.declaration(); %]
[% operation Class declaration() { %]
[%=self.visibility%] class [%=self.name%] {}
[% } %]
```

```
[%=c.declaration()%]
[% @template
operation Class declaration() { %]
[%=self.visibility%] class [%=self.name%] {}
[% } %]
```