CSTL and its application to the osCommerce case study

Albert Tort
atort@lsi.upc.edu

December 2008

(Last update: March 2009)
# Table of contents

1. **INTRODUCTION**............................................................................................................................................. 3

2. **THE CSTL LANGUAGE.......................................................................................................................... 3
   2.1 FIVE DESIGN PRINCIPLES OF CSTL ........................................................................................................ 3
   2.2 TEST PROGRAM STRUCTURE .................................................................................................................. 5
   2.3 KINDS OF TEST CASES .......................................................................................................................... 5
   2.4 TEST VERDICTS ..................................................................................................................................... 6
   2.5 CSTL TYPES AND VALUE EXPRESSIONS ............................................................................................... 7
   2.6 LANGUAGE SYNTAX ............................................................................................................................... 7
   2.7 CSTL STATEMENTS ................................................................................................................................. 9
      2.7.1 State statements ................................................................................................................................. 9
               Entity creation ................................................................................................................................. 9
               Event occurrence .......................................................................................................................... 9
               Entity deletion ............................................................................................................................... 11
               Binary property setting .................................................................................................................. 11
               N-ary relationship creation ........................................................................................................... 11
               Fixture component loading ........................................................................................................... 12
      2.7.2 Variable statements .......................................................................................................................... 12
               Variable declaration ....................................................................................................................... 12
               Variable assignment ...................................................................................................................... 12
               Variable assignment and declaration ............................................................................................ 13
      2.7.3 Assert statements ............................................................................................................................ 13
               Assert true ...................................................................................................................................... 13
               Assert false .................................................................................................................................... 13
               Assert equals ................................................................................................................................. 14
               Assert not equals ........................................................................................................................... 14
               Assert consistency .......................................................................................................................... 14
               Assert inconsistency ....................................................................................................................... 14
      2.7.4 Control flow statements ................................................................................................................... 15
               Conditional statement .................................................................................................................... 15
               For statement ................................................................................................................................. 15
               For each statement .......................................................................................................................... 16
               While ............................................................................................................................................... 16

3. **CSTL APPLICATION TO THE OSCOMMERCE CASE STUDY**............................................................... 17
   3.1 THE CASE STUDY ..................................................................................................................................... 17
   3.2 EXECUTABLE CSUT .................................................................................................................................. 17
   3.3 MAIN DOMAIN CONCEPTS .................................................................................................................... 18
   3.4 CSTL APPLICATION ............................................................................................................................... 20

REFERENCES ...................................................................................................................................................... 177

APPENDIX A: EXECUTABLE CONCEPTUAL SCHEMA OF THE OSCOMMERCE SYSTEM .................. 178

APPENDIX B: EXAMPLE CSTL METHODS OF THE CASE STUDY .................................................. 256
1. Introduction

Software testing is widely accepted as an important activity to enhance the quality of information systems during its development.

Our proposal is based on the idea that conceptual schemas are software artifacts and consequently, they can also be tested. Testing conceptual schemas has some similarities with the well-known activity of testing software programs but there are also important differences. In this context, our work is addressed to explore the use of testing during the elicitation of the conceptual schema as an early error detection practice to help increasing software quality.

We developed the Conceptual Schema Testing Language (CSTL), a language for writing automated tests of executable conceptual schemas.

In this report, we present this language and some example results of its application to the conceptual schema of a real-world information system. We use the conceptual schema of the osCommerce system [9], a widespread e-commerce solution which is the base of thousands of online stores around the world.

2. The CSTL language

2.1 Five design principles of CSTL

The essential purpose of CSTL is providing a textual, procedural, formal and executable notation for writing automated tests of conceptual schemas written in UML/OCL [5,6].

CSTL syntax has been designed by finding a balance between expressiveness, simplicity and understandability of the specified tests. In order to achieve this purpose, CSTL design is based in the following principles:

- **CSTL allows defining the most common tests kinds applicable to conceptual schemas.** We proposed a list of five kinds of tests that can be applied to conceptual schemas. CSTL is designed to allow specifying them:
  - Checking the consistency of an IB state.
  - Checking the inconsistency of an IB state.
  - Checking the contents of an IB state.
  - Checking the occurrence of domain events.
  - Checking that a domain event may not occur.

- **CSTL facilitates the task of writing tests.** Given that writing tests consume time, CSTL pretends to make possible the definition of tests guided by the idea to express as much information as possible by writing as less as possible. In other words, we find a
balance between simplicity and expressiveness. This objective is more feasible in a specialized conceptual schema testing language like CSTL.

- **CSTL is focused on enhancing tests understandability:** Tests of executable conceptual schemas specified in CSTL can be seen as executable specifications of concrete scenarios of requirements. CSTL tests, once defined, have the particularity that they can be executed automatically as many times as needed. Consequently, they are an interesting approach for requirements validation. In this context, CSTL syntax has been designed to be easy understandable and as closed as possible to the way of describing tests in the natural language. The definition of associated pattern sentences to each language statement was a key technique that guided the CSTL design.

- **CSTL follows the style of the modern xUnit testing frameworks:** CSTL syntax is inspired on existing languages that are used for testing in other context and fields, but not suitable at all to test conceptual schemas. CSTL follows the style of xUnit testing languages in the field of programming. CSTL includes the usual instantiation, assignment, conditional and iteration statements needed to write test cases but it also includes built-in constructs that correspond to the elements found in the modern xUnit testing frameworks, and the formalization of test assertions.

- **CSTL tests can be executed by an interpreter:** The proposed language has been designed as an executable language. We developed an interpreter that makes possible to execute tests written in CSTL. This interpreter assumes that the Conceptual Schema Under Test (CSUT) is formally specified in a syntax based on USE but enriched to enhance its expressiveness as explained in section 3.2.

**Fig. 1. Test processor screenshot**
2.2 Test program structure

Figure 2 shows the fragment of the metamodel of test programs. A test program is the top-level structure of CSTL. It consists of:

- A set of **test cases**: A test case is a “specification of one case to test the system including what to test with, which input, result, and under which conditions” [7]. The execution of a test case comprises the execution of an ordered set of statements addressed to test the test kinds enumerated in section 2.1.

- A **fixture**: The fixture is a set of statements that create a state of the IB and define the values of the common program variables. The fixture of a test program is the initial state configuration shared by all the test cases included in the test program.

- A set of **fixture components**: A fixture component is a named set of statements that create a fragment of the state of the IB and define the values of a set of variables. In contrast with the program fixture, fixture components must be load explicitly in test cases or in the program fixture only when needed.

![Diagram of CSTL metamodel fragment of test programs.](image)

**Figure 2.** CSTL metamodel fragment of test programs.

2.3 Kinds of test cases

CSTL allows specifying three kinds of tests:

- **Concrete test case**: A concrete test case is an executable set of statements that builds a state of the IB, define and assign values to variables and executes one or more test kinds.

- **Abstract test case**: An abstract test case is a parameterized test case that can be invoked several times in a test program. An abstract test case cannot be executed.

- **Abstract test case invocation**: Abstract test cases can be invoked by giving a concrete context (defined by the desired values assigned to parameters).
CSTL and its application to the osCommerce case study.
Albert Tort

![Diagram](https://example.com/diagram.png)

**Figure 3.** CSTL metamodel fragment of test cases.

2.4 Test verdicts

![Diagram](https://example.com/diagram.png)

**Figure 4.** CSTL metamodel fragment of test verdicts.
CSTL and its application to the osCommerce case study.
Albert Tort

The execution of a test case gives a **Verdict** as a result. Verdict values can be *Pass*, *Fail* or *Error*. The verdict of a test case is obtained from the verdicts of the test assertions executed by the test case. Test programs also have a verdict as a composite result of the test cases it groups.

Figure 4 shows the fragment of the CSTL metamodel corresponding to verdicts. Note that the derivation rules specify how test cases and test program verdicts are obtained.

### 2.5 CSTL types and value expressions

CSTL allows the value types defined in the OCL 2.0 metamodel [5]. Moreover, the language introduces a specific type called *FixtureComponentType*. This specific type allows declaring fixture components and using them as parameters for abstract test cases. CSTL permits the use, as values, of the different kinds of *ValueSpecifications* defined in the UML 2.0 metamodel [6]. A fixture is also a valid value in CSTL.

![Figure 5. CSTL metamodel fragment of CSTL values and types.](image)

### 2.6 Language syntax

In the previous sections we explained the abstract syntax of the main elements of CSTL. In this section we present the CSTL syntax used for defining test programs and test cases by means of the top-level fragment of the CSTL grammar. The syntax and the semantics of the leaf statements of this grammar fragment are detailed in the following section.

```
testProgram : testprogram <programName> { fixture? fixtureComponent* testCase* }  

fixture : (stateStatement ;)+  

fixtureComponent : fixturecomponent <fixtureComponentName> { (stateStatement ;)+ }  

testCase : concreteTest  
  | abstractTest  
  | abstractTestInvocation  

concreteTest : test <testName> { (testStatement ; | controlFlowStatement)* }  

abstractTest : abstract test <testName> (parameter*) { (testStatement ; | controlFlowStatement)* }  
```
CSTL and its application to the osCommerce case study.
Albert Tort

Parameter
  : type <parameterName>

AbstractTestInvocation
  : test <abstractTestName> (parameterAssignment*) ;

ParameterAssignment
  : <parameterName> := value

testStatement
  : stateStatement
   | variableStatement
   | assertion

stateStatement
  : entityCreation
   | entityDeletion
   | binaryPropertySetting
   | nAryRelationshipCreation
   | fixtureComponentLoading

variableStatement
  : variableDeclaration
   | variableAssignment

assertion
  : assertTrue
   | assertFalse
   | assertEquals
   | assertNotEquals
   | assertConsistency
   | assertInconsistency

controlFlowStatement
  : conditional
   | forLoop
   | foreachLoop
   | whileLoop

```
figure6.png
Figure 6. Generic test program structure that conforms to CSTL syntax.
```
2.7 CSTL statements

2.7.1 State statements

We can load a state of the Information Base by executing a set of state statements. In this section we present the syntax for:

- Creating and deleting entities.
- Setting binary properties of an entity (attributes or binary relationships).
- Creating new n-ary relationships between entities.
- Loading a fixture component.

State statements can be used in fixtures, fixture components and test cases.

**Entity creation**

**Syntax**

```
[entityID :=] new EntityType1,...,EntityTypen
[(propertyID1:=valueExpression1,..., propertyIDn:=valueExpressionn)];
```

**Pattern Sentence**

“An entity entityID is a new instance of the entity types EntityType1,...,EntityTypen.

The value of valueExpression1 is assigned to the property propertyID1,... and the value of valueExpressionn is assigned to the property propertyIDn.”

In some cases we don’t need to use an entity in other expressions after its creation. If the name of the new entity is not specified, it is identified by an internal Object Identifier (OID) that is not known by the conceptual modeler. Therefore, the entity is created, but it cannot be referenced in other expressions later on.

The order in which properties are specified is irrelevant. This is an interesting characteristic of CSTL. If we add, remove or reorder properties in the Conceptual Schema Under Test (CSUT) we don’t need to change already done tests. Moreover, properties can be attributes or binary association ends. If we change the way of representing a property, we don’t need to change the already written tests.

Note that we allow multiple classification. That is, an entity can be an instance of several entity types at the same time.

**Event occurrence**

We adopt the approach that events are modeled in the CSUT as stereotyped entities [4]. Therefore, the basic syntax used for checking that an event (domain event, action request event or query) occurs or not is very similar to the syntax used for entity creation. However, the semantics are more extensive and two options are included to check if an event can occur or not.
Event entities are created as any other entity, but they specify an operation effect(), with an associated method, that is the procedural specification of event effect in an executable form. If the created entity type is an event, the following semantics are applied:

**Syntax**

\[
\text{[eventID :=] new \text{EventEntityTypeID}}
\text{(c\textsubscript{1}=valueExpression\textsubscript{1},..., c\textsubscript{n}=valueExpression\textsubscript{n})}
\text{[occurs | may not occur];}
\]

**Pattern Sentence**

“The eventID is a new event of type EventEntityTypeId (with the characteristics c\textsubscript{1} with the value of valueExpression\textsubscript{1},... and the characteristic c\textsubscript{n} with the value of valueExpression\textsubscript{n}), that occurs | may not occur”

If we don’t specify any of the occurrence options (occurs or may not occur) the event effect is not applied, although the event entity is created in the IB. We can execute the effect later on by writing:

**Syntax**

\[
\text{eventID [occurs | may not occur];}
\]

**Pattern Sentence**

“The effect of the eventID occurs | may not occur”.

Using this syntax variant we can create the event entity and specify characteristics of it in separate statements. We can specify the event characteristics by using the state statements (as it is done for any other entity). Once we have specified all the required characteristics, we can explicitly indicate the effect execution.

The execution of an event is performed as follows:

1. Check that the current IB state is consistent. The verdict is Error if that checking fails.
2. Check that the constraints of the event are satisfied. The verdict is Fail if any of the event constraints is not satisfied.
3. Execute the method of the corresponding effect() operation.
4. Evaluate and update the derived information in the IB.
5. Check that the event postconditions are satisfied. The verdict is Fail if any of the postconditions is not satisfied.
6. Check that the new IB state is consistent (as defined in 3.2). The verdict is Fail if any of the constraints is not satisfied; otherwise the verdict is Pass.

Note that an event cannot occur if the event constraints are not satisfied in the IB state. Therefore, if all the event constraints are satisfied and we are checking that the domain event may not occur, the verdict is Fail.

Note that the order in which we specify the event characteristics is irrelevant.
CSTL and its application to the osCommerce case study.
Albert Tort

**Entity deletion**

Syntax

delete entityID;

*Pattern Sentence*

“Delete the entity entityID”

**Binary property setting**

Syntax

entityID.role := participant1,...,participantn;

*Pattern Sentence*

“The entity entityID is related with the role role to the entities participant1,...,participantn”

Note that this statement can be used for assigning UML attributes or association ends. CSTL considers that an entity has binary properties regardless how they are expressed in UML (as an association or as an attribute). This is a remarkable characteristic of CSTL if used in a test-driven conceptual modeling environment in which tests are written incrementally [10]. This abstraction avoids changing the already done tests if we decide to change the way of representing a binary property in UML.

**N-ary relationship creation**

Syntax

[objectID :=] new AssociationID (roleID1 := entityID1,...,roleIDn := entityIDn) ;

*Pattern Sentence*

“AssociationID is a new association that relates the entity entityID1 on the role roleID1,..., and the entityIDn on the role roleIDn”

This statement requires two or more entities to be related (n>2). For n=2, the *Binary property setting* can be applied with the same result in the IB state.

The order in which we assign entities to roles is irrelevant and it does not depend on the order in which they are specified in the CSUT.

If AssociationID is an association class, then the above statement returns the identifier of the instance of that class.
**Fixture component loading**

**Syntax**

```
load fixtureComponentID ;
```

**Pattern Sentence**

"Load the IB state changes as specified by the fixture component fixtureComponentID"

The loading process executes the state instructions specified by the fixture component. Therefore, the IB state is modified as indicated by the state instructions that constitute the loaded fixture component.

**2.7.2 Variable statements**

CSTL allows storing values in variables to be used in subsequent statements. In this section we present the syntax for declaring variables and for assigning values to these variables.

Variables are only visible in its scope which is determined by the location in which they are declared. The scope of a variable makes it visible in the structure (test program, fixture component, test case, or control flow statement) where it has been declared and its nested substructures.

**Variable declaration**

**Syntax**

```
varType varID ;
```

**Pattern Sentence**

"The variable varID of type varType is declared"

Note that variable declaration it is useful for declaring a variable in the desired context (in order to make it visible in the corresponding structures). Its value is undefined.

**Variable assignment**

**Syntax**

```
varID := valueExpr;
```

**Pattern Sentence**

"The resulting value of the expression valueExpr is assigned to the variable varID".

If the variable varID is not declared, the statement behaves as a VariableAssignmentAndDeclaration statement.

A value expression is an OCL expression evaluated on the current state of the IB.
Variable assignment and declaration

Syntax

\[
\text{[varType]} \ \text{varID} := \text{valueExpr};
\]

Pattern Sentence

“\textbf{The resulting value of the expression valueExpr is assigned to the new variable varID [of type variType]}”.

valueExpr is an OCL expression evaluated on the current state of the IB.

Note that this is a composite statement because it allows declaring a variable and assigning a value to the new variable.

If the \text{varType} is not specified, it is assumed that the type of the new variable corresponds to the predefined type of the assigned value expression.

2.7.3 Assert statements

Assert statements allow formalizing assertions about the current state of the Information Base. These assertions contribute to make the tests automatically executable. Once defined the assertions of a test, they can be checked automatically as many times as needed. If the current state of the IB is inconsistent, the verdict of \textit{assert true}, \textit{assert false}, \textit{assert equals} and \textit{assert not equals} statements is Error (any assertion about the contents of an inconsistent state is an error).

\textit{Assert true}

Syntax

\[
\text{assert true booleanExpression};
\]

Pattern Sentence

“\textit{Assert that the expression booleanExpression is true in the current state of the IB}”.

A Boolean expression is an OCL expression the result of which, after its evaluation on the current state of the IB, is a Boolean value.

\textit{Assert false}

Syntax

\[
\text{assert false booleanExpression};
\]

Pattern Sentence

“\textit{Assert that the expression booleanExpression is false in the current state of the IB}”. 
CSTL and its application to the osCommerce case study.
Albert Tort

**Assert equals**

*Syntax*

\[
\text{assert equals \ valueExpression}_1 \ \text{valueExpression}_2; \\
\]

*Pattern Sentence*

"*Assert that the expression valueExpression\_1 is equal to valueExpression\_2*."

A value expression is an OCL expression evaluated on the current state of the IB.

**Assert not equals**

*Syntax*

\[
\text{assert not equals \ valueExpression}_1 \ \text{valueExpression}_2; \\
\]

*Pattern Sentence*

"*Assert that the expression valueExpression\_2 is not equal to valueExpression\_2*."

A value expression is an OCL expression evaluated on the current state of the IB.

**Assert consistency**

*Syntax*

\[
\text{assert consistency; } \\
\]

*Pattern Sentence*

"*The current state of the IB is consistent*."

This statement checks that the IB satisfies the static and temporal constraints defined in the conceptual schema under test. If before this statement, there are instances of a derived type, the materialized state corresponding to the instantiated derived types is also checked.

**Assert inconsistency**

*Syntax*

\[
\text{assert inconsistency; } \\
\]

*Pattern Sentence*

"*The current state of the IB is inconsistent*."

This statement checks that the IB:

- does not satisfy at least one of the static or temporal constraints defined in the conceptual schema under test, or
- the materialized state corresponding to the previously instantiated derived types is not consistent.
2.7.4 Control flow statements

Control flow statements allow altering the sequential order in which a set of statements are executed. CSTL provides conditional statements to execute alternative sets of statements depending on the evaluation of a specified condition over the IB state. CSTL also provides loop structures to automatically repeat the execution of a set of statements while a specified condition is satisfied.

**Conditional statement**

**Syntax**

```cstl
if booleanExpression1 then statements1,
[else if booleanExpression2 then statements2]
[else if booleanExpressionn-1 then statementsn-1]
[else statementsn]
endif
```

**Pattern Sentence**

“*If the expression* booleanExpressioni *evaluates true, the set of statements* statementsi *is executed. Otherwise, the set of statements* statementsn *is executed*”.

**For statement**

**Syntax**

```cstl
for [varType] varID := valueExpr1 to valueExpr2 step valueExpr3
do statements
endfor
```

**Pattern Sentence**

“*Given a variable* varID *initialized with the value of valueExpr1, the set of statements* statements *are repeated until* varID *is equal to the value of valueExpr2. In each iteration the value obtained by evaluating the expression* valueExpr3 *is assigned to* varID”.

If the variable varID has not been declared yet in the scope, it is declared automatically with the specified type. The *for statement* is the scope of the variables declared inside it.

If the variable type is not explicitly specified and the variable has not been declared yet in the scope, the variable is declared automatically with the predefined type of the assigned expression (valueExpr1).

If the variable type is explicitly specified and the variable is already declared, varType must be of the type of the variable varID.

Note that value expressions should be compatible with the variable type. A type $T_1$ is compatible with type $T_2$ if values of type $T_1$ can be assigned to variables of type $T_2$. 
CSTL and its application to the osCommerce case study.
Albert Tort

**For each statement**

**Syntax**

```cstl
for each [varType] varID in collectionExpr
do statements
endfor
```

**Pattern Sentence**

“*For each element of the resultant collection of* `collectionExpr` *do the set of statements* `statements`.
*Statement can use the current element of the collection, which is stored in the variable* `varID`.”

`collectionExpr` is an OCL expression the result of its evaluation is a collection.

The type of the variable `varID` must be compatible with the type of the `collectionExpr`.

If the variable has not been declared yet, it is automatically declared. The already declared variable is reused to store the current element of the collection in each iteration.

If the type of the variable is not specified and the variable needs to be created, it is assumed that the type of the new value is the predefined type resulting of the evaluation of the expression `collectionExpr`.

**While**

**Syntax**

```cstl
while booleanExpr
do statements
endfor
```

**Pattern Sentence**

“*While* `booleanExpr` *evaluates true repeat the set of statements* `statements`.”

`booleanExpr` is an OCL expression the result of its evaluation is a Boolean value.
3. CSTL application to the osCommerce case study

3.1 The case study

*E-commerce* allows people exchanging goods and services with no barriers of time or distance.

*osCommerce* [8] is an e-commerce solution available as free software under the GNU (General Public License). *osCommerce* project was started in March 2000 in Germany and since then, it has become the base of thousands of online stores around the world. *osCommerce* can be customized to operate in different countries (with different languages, taxes, currencies,…) and to be used in several kinds of online stores.

In this section we provide a set of representative test programs taking the osCommerce conceptual schema [9] as the Conceptual Schema Under Test (CSUT).

The osCommerce conceptual schema models the real *osCommerce* system that includes a considerable number of concepts, relationships and events. Therefore, it is necessary to structure the schema in subschemas to improve its comprehension. The osCommerce CS models the structural knowledge of the system in UML/OCL and gives the specification of the more relevant use cases in an informal textual description. Uses cases are linked to the events wich are formally defined in UML/OCL by adopting the approach of modeling events as entities [4].

We start by introducing how we specify the osCommerce conceptual schema in an executable form. Then, we give a general overview of the main concepts of the osCommerce domain. After that, example test programs are presented as follows: for each substructural schema, we show the most relevant use cases that require the static knowledge represented in the substructural schema. Then, we show its associated events. Given that CSTL tests can be used to test incomplete fragments of conceptual schemas or concrete scenarios of use cases, test programs are inserted after them to exemplify relevant tests that could be applied to the parts of the conceptual schema.

Some of the example test programs are inspired in real and live online stores based on *osCommerce*.

3.2 Executable CSUT

The CSTL interpreter assumes that the CSUT is specified in a syntax based on USE. The USE syntax is explained in detail in [1]. Note that, although the syntax is much closed to the standard UML/OCL syntax, USE adopts some particular notation for some OCL expressions. For example: data types must be specified as UML classes and enumeration values are referenced with the symbol ‘#’.

USE does not allow the specification of derived types or the definition of event constraints.
CSTL and its application to the osCommerce case study.
Albert Tort

In order to address these important characteristics for testing conceptual schemas, we enriched the USE syntax as follows:

- **Derived Types.** An attribute \(Attr\) is assumed to be derived if it is preceded by the character \('_\)\. Therefore, it is assumed that \(\_Attr\) is a derived attribute named \(Attr\). The derivation rule must be specified as an operation without parameters named \(Attr()\). Consider the following class definition as an example:

```
class Category
attributes
  imagePath:String
  \_subcategories:Integer -This is a derived attribute
operations
  subcategories():Integer=self.child->size()
```

- **Initial Integrity Constraints.** Creation-time constraints are also allowed by using the enriched syntax of USE used in the CSTL interpreter. This particular type of constraints can be explicitly defined by adding the string \("_iniIC_"\) before the constraint name as indicated in the following example:

```
context OrderConfirmation inv _iniIC_ShippingMethodIsEnabled:
  self.shippingMethod.status= #enabled
```

Moreover, the information processor of USE has been extended to deal with richer conceptual schemas because: (1) it allows derived entity and relationship types; (2) in particular, it allows derived constant relationship types; (3) events and predefined queries are conceptualized as entities and not as operation invocations [4]; (4) it allows the definition and checking of temporal constraints; (5) it allows the materialization of derived properties; and (6) it deals with conceptual schemas that allow multiple classification of entities.

### 3.3 Main domain concepts

The products in the store are manufactured by manufacturers, are grouped into categories and belong to a tax class. Moreover, customers can write reviews of a product.

osCommerce is a multilingual system able to deal with any number of languages. Likewise, osCommerce allows working with different tax classes and currencies.

Products may have attributes. An attribute is an option/value pair which is used to offer multiple varieties of a product without needing to create many separate but very similar products. The price of a product is increased or decreased depending on the chosen attributes. The price variation produced by an attribute is indicated, for each product, by product attribute entity types.

Customers have one or more addresses. Each address is located in a country. If the country has zones (states or provinces) then the address must be located in one of its zones.
Every use of the online store is conceptually represented by a **session**. Sessions can be anonymous or belong to a customer. Moreover, every session has always a current currency and a current language.

In the context of sessions, users can surfing the online store. **Shopping carts** contain one or more selected items (not shown in the figure) each of which is a quantity of a product with a set of attributes.

When a customer confirms that he wants to buy the contents of his shopping cart the system generates an **order**. An order is made by a customer using a **payment method**. Furthermore, order prices are expressed in a specified **currency** and take into account the shipping costs, according to the chosen **shipping method**.

An order contains one or more **order lines**, each of which is a quantity of a product with a set of attributes.

Finally, osCommerce offers some administration tools like **banners**, used to customize the online advertisements in the store, and **newsletters**, used to send information by email to customers.
3.4 CSTL application

Store Data

Structural schema

*osCommerce* keeps general data about the store and some other information which is used to customize the behavior of the system.

```
<table>
<thead>
<tr>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>name : String [0..1]</td>
</tr>
<tr>
<td>owner : String [0..1]</td>
</tr>
<tr>
<td>eMailAddress : EMail  [0..1]</td>
</tr>
<tr>
<td>eMailFrom : EMail [0..1]</td>
</tr>
<tr>
<td>expectedSortOrder : SortOrder</td>
</tr>
<tr>
<td>expectedSortField : SortField</td>
</tr>
<tr>
<td>sendExtraOrderEMail : NameEMail [^]</td>
</tr>
<tr>
<td>displayCanAfterAddingProduct : Boolean</td>
</tr>
<tr>
<td>allowGuestToTellAFriend : Boolean</td>
</tr>
<tr>
<td>defaultSearchOperator : Operator</td>
</tr>
<tr>
<td>storeAddressAndPhone : String [0..1]</td>
</tr>
<tr>
<td>isoDecimalPlaces : Natural</td>
</tr>
<tr>
<td>displayPricesWithTax : Boolean</td>
</tr>
<tr>
<td>switchToDefaultLanguageCurrency : Boolean</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Zone</th>
</tr>
</thead>
<tbody>
<tr>
<td>name : String [0..1]</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>name : String [0..1]</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>OrderStatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>cancelledStatus : String [0..1]</td>
</tr>
<tr>
<td>defaultStatus : Boolean</td>
</tr>
</tbody>
</table>
```

[IC1] There is only one instance of *Store*

**context** Store::alwaysOneInstance: Boolean  
**body**: Store.allInstances() -> size() = 1

[IC2] The store’s zone is part of the country where the store is located.

**context** Store::zoneIsPartOfCountry: Boolean  
**body**: self.zone -> notEmpty() implies self.country.zone -> includes (self.zone)

**Example test program**

```
testprogram InitializeStore{
    english := new Language(name:='English', code:='EN');
    dollar := new Currency(title:='US Dollar', code:='USD');
    newjersey := new Zone(name:='New Jersey', code:='NJ', country:='usa');
    catalonia := new Zone(name:='Catalonia', code:='CAT', country:='spain');
}```
CSTL and its application to the osCommerce case study.
Albert Tort

cos := new OrderStatus;
cosl := new OrderStatusInLanguage(language:=english, orderStatus:=cos);
cosl.name := 'cancelled';

dos := new OrderStatus;
dosl := new OrderStatusInLanguage(orderStatus:=dos, language:=english);
dosl.name := 'pending';

test StoreInitializationWithDefaultMandatoryValues{
    s := new Store(name:='JustArt');
    assert inconsistency;
    s.defaultLanguage := english;
    assert inconsistency;
    s.defaultCurrency := dollar;
    assert inconsistency;
    s.country := usa;
    assert inconsistency;
    s.cancelledStatus := cos;
    assert inconsistency;
    s.defaultStatus := dos;
    assert consistency;
}

test OnlyOneStoreInstance{
    // We create the store 'JustArt'
    s := new Store(name:='JustArt');
    s.defaultLanguage := english;
    s.defaultCurrency := dollar;
    s.country := usa;
    s.cancelledStatus := cos;
    s.defaultStatus := dos;
    assert consistency;

    // If we create another store, the state should be inconsistent
    s2 := new Store(name:='VirtualGallery');
    s2.defaultLanguage := english;
    s2.defaultCurrency := dollar;
    s2.country := usa;
    s2.cancelledStatus := cos;
    s2.defaultStatus := dos;
    assert inconsistency;
}

test StoreZoneMustBePartOfTheCountryWhereItIsLocated{
    // We create the store 'VirtualGallery'
    s := new Store(name:='VirtualGallery');
    s.defaultLanguage := english;
    s.defaultCurrency := dollar;
    s.country := usa;
    s.cancelledStatus := cos;
    s.defaultStatus := dos;
    assert consistency;

    // We specify a zone which is not part of the USA
    s.zone := catalonia;
    assert inconsistency;

    // We specify a correct zone
    s.zone := newjersey;
    assert consistency;
}
Use Cases

Change Store Data

Primary Actor: System administrator
Precondition: None.
Trigger: The system administrator wants to change the initial values of the store data.

Main Success Scenario:

1. The system displays the current values of the store data.
2. The system administrator provides a new value for one of the store attributes:
   - [→MameChange]
   - [→OwnerChange]
   - [→EMailAddressChange]
   - [→EMailFromChange]
   - [→ExpectedSortOrderChange]
   - [→ExpectedSortFieldChange]
   - [→SendExtraOrderChange]
   - [→DisplayCartAfterAddingProductChange]
   - [→AllowGuestToTellAFriendChange]
   - [→DefaultSearchOperatorChange]
   - [→StoreAddressAndPhoneChange]
   - [→TaxDecimalPlacesChange]
   - [→DisplayPricesWithTaxChange]
   - [→SwitchToDefaultLanguageCurrencyChange]
   - [→CountryChange]
   - [→ZoneChange]
3. The system validates that the value is correct.
4. The system saves the new value.
5. The system displays the new values of the store data.

   The system administrator repeats steps 2-5 until he is done.

Note that if there are many similar events, we only reproduce the complete specification of the selected representative events used in the test program examples. The other similar events can be found in [9].
CSTL and its application to the osCommerce case study.
Albert Tort

Events

NameChange

```
context NameChange::effect()
  post : self.myStore.name = self.newName
```

CountryChange

```
context CountryChange::effect()
  post : myStore.country = self.newCountry
```

Example test program

```
testprogram ChangeStoreData{

  //FIXTURE:InitializeStore
  s := new Store(name:='JustsArt');
  english := new Language(name:='English', code:='EN');
  s.defaultLanguage = english;
  dollar := new Currency(title:='USDollar', code:='USD');
  s.defaultCurrency := dollar;

  spain := new Country
  s.country = spain;

  cos := new OrderStatus;
  cosl := new OrderStatusInLanguage(language=english, orderStatus=cos);
  cosl.name := 'cancelled';
  s.cancelledStatus := cos;

  dos := new OrderStatus;
  dosl := new OrderStatusInLanguage(orderStatus=dos, language=english);
  dosl.name := 'pending';
  s.defaultStatus := dos;

  //We test that name and country can be correctly changed.
  test NameAndCountryChange{
    assert equals s.name 'JustsArt';
  }
```
Configuration values

Structural schema

osCommerce allows defining and changing the minimum and maximum length for some String attributes related to customer details.

```
new NameChange(newName:='JustArt') occurs;
assert equals s.name 'JustArt';

assert equals s.country spain;
usa := new Country
new CountryChange(newCountry:=usa);
assert equals s.country usa;
```
CSTL and its application to the osCommerce case study.
Albert Tort

[IC1] The package tare weight must be less than the maximum package weight.

context ShippingAndPackaging::tareIsLessThanMaximumWeight: Boolean
body : self.typicalPackageTareWeight < self.maximumPackageWeight

The system allows customizing the most important general downloadable product properties.

<table>
<thead>
<tr>
<th>&lt;&lt;utility&gt;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download</td>
</tr>
<tr>
<td>enableDownload : Boolean</td>
</tr>
<tr>
<td>daysExpiryDelay : Natural</td>
</tr>
<tr>
<td>maximumNumberOfDownloads : Natural</td>
</tr>
</tbody>
</table>

The system allows configuring some options about the stock administration.

<table>
<thead>
<tr>
<th>&lt;&lt;utility&gt;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stock</td>
</tr>
<tr>
<td>checkStockLevel : Boolean</td>
</tr>
<tr>
<td>subtractStock : Boolean</td>
</tr>
<tr>
<td>allowCheckout : Boolean</td>
</tr>
<tr>
<td>stockReOrderLevel : Natural</td>
</tr>
</tbody>
</table>

Use Cases

Assign minimum values

Primary Actor: System administrator
Precondition: None.
Trigger: The system administrator wants to change the minimum values of some attributes.

Main Success Scenario:

The system displays the current minimum values.

1. The system administrator provides a new value for one of the minimum values:
   - [⇒ FirstNameMinimumChange]
   - [⇒ LastNameMinimumChange]
   - [⇒ DateOfBirthMinimumChange]
   - [⇒ EMailAddressMinimumChange]
   - [⇒ StreetAddressMinimumChange]
   - [⇒ CompanyNameMinimumChange]
   - [⇒ PostCodeMinimumChange]
   - [⇒ CityMinimumChange]
   - [⇒ StateMinimumChange]
   - [⇒ TelephoneMinimumChange]
   - [⇒ PasswordMinimumChange]
   - [⇒ CreditCardOwnerNameMinimumChange]
CSTL and its application to the osCommerce case study.
Albert Tort

Assign maximum values

Primary Actor: System administrator
Precondition: None.
Trigger: The system administrator wants to change the maximum number of address book entries permitted for each customer.

Main Success Scenario:

1. The system displays the current maximum number of address book entries for each customer.
2. The system administrator provides the new maximum value:

   - [→AddressBookEntriesMaximumChange]
3. The system validates that the value is correct.
4. The system saves the new value.
5. The system displays the new current maximum value.

Change shown customer details

Primary Actor: System administrator
Precondition: None.
Trigger: The system administrator wants to change whether some customer attributes are shown.

Main Success Scenario:

1. The system displays the current values of customer details configuration (shown or not shown).
2. The system administrator provides the new value for one of the customer details:

   - [→GenderCustomerDetailChange]
   - [→DateOfBirthCustomerDetailChange]
   - [→CompanyCustomerDetailChange]
   - [→SuburbCustomerDetailChange]
   - [→StateCustomerDetailChange]
3. The system validates that the value is correct.
4. The system saves the new value.
5. The system displays the new current values of customer details configuration.

   The system administrator repeats steps 2-5 until he is done.
Assign shipping and packaging configuration values

Primary Actor: System administrator
Precondition: None.
Trigger: The system administrator wants to change the shipping and packaging configuration values.

Main Success Scenario:

1. The system displays the current shipping and packaging configuration values.
2. The system administrator provides the new value for one of the shipping and packaging configurable options:
   - PostCodeShippingConfigurationChange
   - MaximumPackageWeightShippingConfigurationChange
   - TypicalPackageTareWeightShippingConfigurationChange
   - PercentageIncreaseForLargerPackagesShippingConfigurationChange
   - CountryShippingConfigurationChange
3. The system validates that the value is correct.
4. The system saves the new value.
5. The system displays the new current shipping and packaging configuration values.
   The system administrator repeats steps 2-5 until he is done.

Change download configuration values

Primary Actor: System administrator
Precondition: None.
Trigger: The system administrator wants to change the download configuration values.

Main Success Scenario:

1. The system displays the current download configuration values.
2. The system administrator provides the new value for one of the download configuration options:
   - EnableDownloadConfigurationChange
   - DaysExpiryDelayDownloadConfigurationChange
   - MaximumNumberDownloadConfigurationChange
3. The system validates that the value is correct.
4. The system saves the new value.
5. The system displays the new current download configuration values.
   The system administrator repeats steps 2-5 until he is done.

Change stock configuration values

Primary Actor: System administrator
Precondition: None.
Trigger: The system administrator wants to change the stock configuration values.
CSTL and its application to the osCommerce case study.
Albert Tort

Main Success Scenario:

1. The system displays the current stock configuration values.
2. The system administrator provides the new value for one of the stock configuration options:
   - CheckLevelStockConfigurationChange
   - SubtractStockConfigurationChange
   - AllowCheckoutStockConfigurationChange
   - ReorderLevelStockConfigurationChange
3. The system validates that the value is correct.
4. The system saves the new value.
5. The system displays the new current stock configuration values.
   The system administrator repeats steps 2-5 until he is done.

Events

PasswordMinimumChange

context  PasswordMinimumChange::effect()
post :  MinimumValues.password = self.newMinimum

CreditCardNumberMinimumChange

context  CreditCardNumberMinimumChange::effect()
post :  MinimumValues.creditCardNumber = self.newMinimum
CSTL and its application to the osCommerce case study.
Albert Tort

**AddressBookEntriesMaximumChange**

```
context AddressBookEntriesMaximumChange::effect()
post : MaximumValues.addressBookEntries = self.newMaximum
```

**GenderCustomerDetailChange**

```
context GenderCustomerDetailChange::effect()
post : CustomerDetails.gender = self.newValue
```

**MaximumPackageWeightShippingConfigurationChange**

```
context MaximumPackageWeightShippingConfigurationChange::maxIsGreaterThanTypicalWeight():Boolean
body : self.newMaximum > ShippingAndPackaging.typicalPackageTareWeight
context MaximumPackageWeightShippingConfigurationChange::effect()
post : ShippingAndPackaging.maximumPackageWeight = self.newMaximum
```
CSTL and its application to the osCommerce case study.
Albert Tort

**TypicalPackageTareWeightShippingConfigurationChange**

```plaintext
context TypicalPackageTareWeightShippingConfigurationChange::effect()
post : ShippingAndPackaging.typicalPackageTareWeight = self.newValue
```

**MaximumNumberDownloadConfigurationChange**

```plaintext
context MaximumNumberDownloadConfigurationChange::effect()
post : Download.maximumNumberOfDownloads = self.newMaximum
```

**CheckLevelStockConfigurationChange**

```plaintext
context CheckLevelStockConfigurationChange::effect()
post : Stock.checkStockLevel = self.newValue
```
Example test program

```plaintext
testprogram ConfigurationValues{
    // We create an instance of the entity types
    // MaximumValues and MinimumValues (multiple classification)
    configurationValues := new MaximumValues, MinimumValues,
        CustomerDetails, ShippingAndPackaging, Download, Stock;
    spain := new Country(name:'Spain', isoCode2:'ES', isoCode3:'ESP');
    configurationValues.countryOfOrigin := spain;
    configurationValues.maximumPackageWeight := 30;
    configurationValues.typicalPackageTareWeight := 15;

test ChangeMinimumAndMaximumValues{
    // The postconditions of the following events are automatically checked
    new PasswordMinimumChange(newMinimum:=8) occurs;
    new CreditCardNumberMinimumChange(newMinimum:=16) occurs;
    new AddressBookEntriesMaximumChange(newMaximum:=3) occurs;
    new GenderCustomerDetailChange(newValue=true) occurs;
    new MaximumNumberDownloadConfigurationChange(newMaximum:=5) occurs;
    new TypicalPackageTareWeightShippingConfigurationChange(newValue=false) occurs;
    new MaximumPackageWeightShippingConfigurationChange(newMaximum:=25) occurs;
}

test InconsistentShippingConfiguration{
    // The typical package weight cannot be greater than the maximum package weight
    new TypicalPackageTareWeightShippingConfigurationChange(newValue:=40) may not occur;
    new MaximumPackageWeightShippingConfigurationChange(newValue:=10) may not occur;
}
}
```

Payment methods

**Structural schema**

The system allows operating with different payment methods.
CSTL and its application to the osCommerce case study.
Albert Tort

[IC1] There is at least one enabled payment method

context PaymentMethod::atLeastOneEnabled: Boolean
body : PaymentMethod.allInstances() -> select (pm | pm.status=Status::enabled) -> size() >= 1

Use Cases

Install a payment method

**Primary Actor:** Store administrator

**Precondition:** The payment method is not installed yet.

**Trigger:** The store administrator wants to install a payment method.

**Main Success Scenario:**

1. The system shows all the available payment methods and which of they are installed.
2. The store administrator selects a non installed payment method.
3. The store administrator provides the data of the payment method:
   - InstallAuthorizeNetPaymentMethod
   - InstallCreditCardPaymentMethod
   - InstallCashOnDeliveryPaymentMethod
   - InstallIPaymentMethod
   - InstallCheckMoneyPaymentMethod
   - InstallNochexPaymentMethod
   - InstallPayPalPaymentMethod
   - InstallTwoCheckOutPaymentMethod
   - InstallPSiGatePaymentMethod
   - InstallSECPaymentMethod
4. The system validates that the data is correct.
5. The system uninstalls the new payment method and enables it.

Uninstall a payment method

**Primary Actor:** Store administrator

**Precondition:** The payment method is installed and there is at least another payment method enabled.

**Trigger:** The store administrator wants to uninstall a payment method.

**Main Success Scenario:**

1. The system shows all the payment methods and which of they are installed.
2. The store administrator selects an installed payment method.
   - UninstallAuthorizeNetPaymentMethod
   - UninstallCreditCardPaymentMethod
   - UninstallCashOnDeliveryPaymentMethod
CSTL and its application to the osCommerce case study.
Albert Tort

[→ UninstallPaymentMethod]
[→ UninstallCheckMoneyPaymentMethod]
[→ UninstallNochexPaymentMethod]
[→ UninstallPayPalPaymentMethod]
[→ UninstallTwoCheckOutPaymentMethod]
[→ UninstallPSiGatePaymentMethod]
[→ UninstallSECPaymentMethod]

3. The system uninstalls the selected payment method.

Extensions:

2a. The payment method is used in an existing order:
   2a1. The system warns the store administrator that the payment method is used in the information of existing orders and that it is only possible to disable the payment method.
   2a2. The system changes the status of the payment method to disabled.
      [→ StatusPaymentMethodChange]
   2a3. The use case ends.

Change payment method values

Primary Actor: System administrator
Precondition: The payment method is installed.
Trigger: The system administrator wants to change the configuration values of an installed payment method.

Main Success Scenario:

1. The system displays the installed payment methods.
2. The customer selects an installed payment method.
3. The system displays the current values of the payment method.
4. The system administrator provides the new values for the configurable attributes of the payment method:
   [→ EditAuthorizeNetPaymentMethod]
   [→ EditCreditCardPaymentMethod]
   [→ EditCashOnDeliveryPaymentMethod]
   [→ EditPayPalPaymentMethod]
   [→ EditCheckMoneyPaymentMethod]
   [→ EditNochexPaymentMethod]
   [→ EditPSiGatePaymentMethod]
   [→ EditSECPaymentMethod]
5. The system validates that the new values are correct.
6. The system saves the new values.
7. The system displays the new values of the payment method.
CSTL and its application to the osCommerce case study.
Albert Tort

Events

**InstallCreditCardPaymentMethod**

```
context InstallCreditCardPaymentMethod::paymentMethodIsNotInstalled():Boolean
body : CreditCard.allInstances() -> isEmpty()
```

```
context InstallCreditCardPaymentMethod::effect()
post : pm.oclIsNew() and pm.oclIsTypeOf(CreditCard) and pm.status=Status::enabled
```

**UninstallCreditCardPaymentMethod**

```
context UninstallCreditCardPaymentMethod::paymentMethodCanBeUninstalled():Boolean
body : CreditCard.allInstances() -> notEmpty() and
(PaymentMethod.allInstances-Set{CreditCard.allInstances->any(true)})->exists(pm | pm.status=#enabled)
```

```
context UninstallCreditCardPaymentMethod::effect()
post : CreditCard.allInstances() -> any(true)@pre.oclIsKindOf(OclAny)
```

**EditCreditCardPaymentMethod**

```
context EditCreditCardPaymentMethod::paymentMethodIsInstalled():Boolean
body : CreditCard.allInstances() -> notEmpty()
```

```
context EditCreditCardPaymentMethod::newSplitCreditCardToMail : EMail
```

```
context EditCreditCardPaymentMethod::effect()
newSplitCreditCardToMail : EMail
```

```
context EditCreditCardPaymentMethod::status:Status
```

```
context EditCreditCardPaymentMethod::TaxZone
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```

```
context EditCreditCardPaymentMethod::EditCreditCardPaymentMethod
```

```
context EditCreditCardPaymentMethod::EditPaymentMethodEvent
```

```
context EditCreditCardPaymentMethod::OrderStatus
```
CSTL and its application to the osCommerce case study.
Albert Tort

```ini
context EditCreditCardPaymentMethod::atLeastOneEnabled():Boolean
  body:
  self.status=Status::disabled
  implies
  (PaymentMethod.allInstances-Set(CreditCard.allInstances->any(true))
    ->exists(pm | pm.status=Status::enabled))

context EditCreditCardPaymentMethod::effect()
post:
  let pm:CreditCard = CreditCard.allInstances() -> any(true) in
  pm.splitCreditCardToMail=self.newSplitCreditcardToMail and pm.status=self.status and
  pm.orderStatus=self.orderStatus and pm.taxZone=self.taxZone
```

**Example test program**

```plaintext
testprogram InstallUninstallAndEditPaymentMethods{
  test InstallCreditCardOnce{
    new InstallCreditCardPaymentMethod occurs;
  }

  test InstallCreditCardTwice{
    new InstallCreditCardPaymentMethod occurs;
    new InstallCreditCardPaymentMethod may not occur;
  }

  test UninstallCreditCardAlreadyInstalled{
    new InstallCreditCardPaymentMethod occurs;
    // We cannot uninstall the credit card method because
    // there is no other payment method enabled
    new UninstallCreditCardPaymentMethod may not occur;
    new UninstallCreditCardPaymentMethod occurs;
  }

  test AtLeastOnePaymentMethodEnabled{
    new InstallCreditCardPaymentMethod occurs;
    // We cannot disable the credit card method because
    // there is no other payment method enabled
    new EditCreditCardPaymentMethod(status:=#disabled) may not occur;
    new InstallCashOnDeliveryPaymentMethod occurs;
    new EditCreditCardPaymentMethod(status:=#disabled) occurs;
  }

  test UninstallCreditCardNotInstalledYet{
    new UninstallCreditCardPaymentMethod may not occur;
  }
```

```
Shipping methods

Structural schema

The system allows operating with different shipping methods.

**[IC1]** There is at least one enabled shipping method.

\[
\text{context} \quad \text{ShippingMethod::atLeastOneEnabled: Boolean} \\
\text{body} : \quad \text{ShippingMethod.allInstances() -> select (sm | sm.status=Status::enabled) -> size() >= 1}
\]

Use Cases

**Install a shipping method**

**Primary Actor:** Store administrator

**Precondition:** The shipping method is not installed yet.

**Trigger:** The store administrator wants to install a shipping method.

**Main Success Scenario:**

1. The system shows all the available shipping methods and which of they are installed.
2. The store administrator selects a non installed shipping method.
3. The store administrator provides the data of the shipping method.
CSTL and its application to the osCommerce case study.
Albert Tort

[⇒InstallZoneRatesShippingMethod]
[⇒InstallFlatRateShippingMethod]
[⇒InstallPerItemShippingMethod]
[⇒InstallTableRateShippingMethod]
[⇒InstallUSPostalServiceShippingMethod]

4. The system validates that the data is correct.
5. The system creates an instance of the new shipping method and enables it.

**Uninstall a shipping method**

**Primary Actor:** Store administrator

**Precondition:** The shipping method is installed and there is at least another shipping method enabled.

**Trigger:** The store administrator wants to uninstall a shipping method.

**Main Success Scenario:**

1. The system shows all the available shipping methods and which of them are installed.
2. The store administrator selects an installed shipping method.

   [⇒UninstallZoneRatesShippingMethod]
   [⇒UninstallFlatRateShippingMethod]
   [⇒UninstallPerItemShippingMethod]
   [⇒UninstallTableRateShippingMethod]
   [⇒UninstallUSPostalServiceShippingMethod]

3. The system deletes the instance of the selected shipping method.

**Extensions:**

2a. The shipping method is the shipping method used in an existing order:
   2a1. The system warns the store administrator that the shipping method is used in the information of existing orders and that is only possible to disable the shipping method.
   2a2. The system changes the enabled attribute of the shipping method to false:

      [⇒StatusShippingMethodChange]

   2a3. The use case ends.

**Change shipping method values**

**Primary Actor:** System administrator

**Precondition:** The shipping method is installed.

**Trigger:** The system administrator wants to change the configuration values of an installed shipping method.

**Main Success Scenario:**

1. The system displays the installed shipping methods.
2. The customer selects an installed shipping method.
CSTL and its application to the osCommerce case study.
Albert Tort

3. The system displays the current values of the selected shipping method.
4. The system administrator provides the new values for the configurable attributes of the shipping method:
   - EditZoneRatesShippingMethod
   - EditFlatRateShippingMethod
   - EditPerItemShippingMethod
   - EditTableRateShippingMethod
   - EditUSPostalServiceShippingMethod
5. The system validates that the new values are correct.
6. The system saves the new values.
7. The system displays the new values of the shipping method.

Events

InstallPerItemShippingMethod

UninstallPerItemShippingMethod

38
CSTL and its application to the osCommerce case study.
Albert Tort

### EditPerItemShippingMethod

![Diagram of EditPerItemShippingMethod]

- **<init>**
  - **context**: `EditPerItemShippingMethod::paymentMethodIsInstalled():Boolean`
  - **body**: `PerItem.allInstances() -> notEmpty()

- **<init>**
  - **context**: `EditPerItemShippingMethod::atLeastOneEnabled:Boolean`
  - **body**: `self.status=Status::disabled` implies `(ShippingMethod.allInstances-Set(PerItem.allInstances->any(true))) ->exists(pm | pm.status=Status::enabled)

- **effect**
  - **post**: let `sm: PerItem= PerItem.allInstances() -> any(true)` in
    `sm.cost=self.newCost and sm.handlingFee=self.handlingFee and sm.taxZone=self.taxZone and sm.taxClass=self.taxClass and sm.status = self.status`

#### Example test program

```plaintext
testprogram InstallUninstallShippingMethods{
    test InstallPerItemShippingMethodOnce{
        new InstallPerItemShippingMethod occurs;
    }

test InstallPerItemShippingMethodTwice{
    new InstallPerItemShippingMethod occurs;
    new InstallPerItemShippingMethod may not occur;
}

test UninstallPerItemShippingMethodAlreadyInstalled{
    new InstallPerItemShippingMethod occurs;
    //We cannot uninstall PerItem method because there is no
    //other shipping method enabled
    new InstallFlatRateShippingMethod occurs;
    new UninstallPerItemShippingMethod occurs;
}

test UninstallCreditCardNotInstalledYet{
    new UninstallPerItemShippingMethod may not occur;
}

test AtLeastOneShippingMethodEnabled{
    new InstallPerItemShippingMethod occurs;
}
```
CSTL and its application to the osCommerce case study.
Albert Tort

Languages

Structural schema

`osCommerce` is a multilingual system able to deal with any number of languages.

<table>
<thead>
<tr>
<th>Language</th>
<th>*</th>
<th>Currency</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>code</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>image</td>
<td>File [0..1]</td>
<td></td>
</tr>
<tr>
<td>directory</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>sortOrder</td>
<td>Natural</td>
<td></td>
</tr>
<tr>
<td>defaultCurrency</td>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

**[IC1] A language is identified by its name and by its code**

**context** Language::codeAndNameAreUnique: Boolean

**body**: Language.allInstances() -> isUnique(name) and Language.allInstances() -> isUnique(code)

Use Cases

Add a language

**Primary Actor**: Store administrator

**Precondition**: None.

**Trigger**: The store administrator wants to add a new language.

**Main Success Scenario:**

1. The store administrator provides the details of the new language:
   
   ```
   [→ NewLanguage]
   ```

2. The system validates that the data is correct.

3. The system saves the new language.

Edit a language

**Primary Actor**: Store administrator

**Precondition**: None.

**Trigger**: The store administrator wants to edit a language.
Main Success Scenario:

1. The store administrator selects the language to be edited.
2. The store administrator provides the new details of the selected language:
   
   \[ \rightarrow \text{EditLanguage} \]
3. The system validates that the data is correct.
4. The system saves the changes.

Delete a language

Primary Actor: Store administrator
Precondition: There are at least two languages.
Trigger: The store administrator wants to delete a language.

Main Success Scenario:

1. The store administrator selects the language to be deleted.
2. The store administrator confirms that he wants to delete the language:
   
   \[ \rightarrow \text{DeleteLanguage} \]
3. The system deletes the language.

Extensions:

2a. The deleted language is the default language of the store.
    
   2a1. The system sets any of the available languages as the default language:
       
       \[ \rightarrow \text{SetDefaultLanguage} \]
2b. The deleted language is the current language of any active session.
    
   2b1. The system sets any of the available languages as the current language:
       
       \[ \rightarrow \text{SetCurrentLanguage} \]

Set the default language

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to change the default language.

Main Success Scenario:

1. The store administrator selects the language which will become the default language.
2. The system updates the default language:
   
   \[ \rightarrow \text{SetDefaultLanguage} \]
CSTL and its application to the osCommerce case study.
Albert Tort

Events

**NewLanguage**

```
context NewLanguage::languageDoesNotExist(): Boolean
body:
not Language.allInstances() -> exists (l | l.name=self.name and l.code = self.code)
```

```
context NewLanguage::effect()
post:
l.oclIsNew() and l.oclIsTypeOf(Language) and l.name = self.name and l.code = self.code and l.defaultCurrency = self.defaultCurrency
```

**EditLanguage**

```
context EditLanguage::languageDoesNotExist(): Boolean
body:
not ((Language.allInstances-Set{self.language}) ->exists(name=self.newName or code=self.newCode))
```

```
context EditLanguage::effect()
post:
self.language.name = self.newName and self.language.code = self.newCode and self.language.defaultCurrency = self.newDefaultCurrency
```
CSTL and its application to the osCommerce case study.
Albert Tort

DeleteLanguage

```
context DeleteLanguage::AtLeastTwoLanguages(): Boolean
body : Language.allInstances() -> size() >= 2

context DeleteLanguage::effect()
post: not self.language@pre.oclIsKindOf(OclAny)
```

SetDefaultLanguage

```
context SetDefaultLanguage::effect()
post: Store.allInstances() -> any(true).defaultLanguage = self.language
```

Example test program

```
testprogram LanguageManagement{
    dollar:=new Currency(title:='USDollar', code:='USD');

    test InstallLanguage{
        new NewLanguage(newName='English', newCode='EN') occurs;
    }

    test InstallLanguagesTwice{
        new NewLanguage(newName='English', newCode='EN') occurs;
        new NewLanguage(newName='English', newCode='EN') may not occur;
    }

    test InstallLanguageWithDefaultCurrency{
        new NewLanguage(newName='English', newCode='EN', defaultCurrency:=dollar) occurs;
    }
}
```
CSTL and its application to the osCommerce case study.
Albert Tort

test EditLanguage{
    new NewLanguage(newName="Englishhh", newCode="EN") occurs;
    createdLanguage=Language.allInstances->select(name='Englishhh')->any(true);
    new EditLanguage
        (language:=createdLanguage, newName="English", newCode="EN") occurs;
    assert equals l.name 'English';

    // We cannot edit a language if it causes duplicated languages
    catalan := new Language(name="Catalan", code="CAT");
    new EditLanguage(language:=createdLanguage,newName="Catalan", newCode="EN") may not occur;
}
test DeleteLanguage{
    // We cannot delete a language if there are no other languages enabled
    english := new Language(name="English", code="EN", defaultCurrency=dollar);
    new DeleteLanguage(language:=english) may not occur;
    catalan := new Language(name="Catalan", code="CAT");
    new DeleteLanguage(language:=english) occurs;
}
test SetDefaultLanguage{
    // Initialize store
    english:=new Language(name="English", code="EN");
    usa:=new Country(name="United States", isoCode2="US", isoCode3="USA");
    cos:=new OrderStatus;
    cosl:=new OrderStatusInLanguage(language:=english,orderStatus:=cos);
    cosl.name:="cancelled";
    dos:=new OrderStatus;
    dosl:=new OrderStatusInLanguage(orderStatus:=dos, language:=english);
    dosl.name:="pending";
    s:=new Store(name='VirtualGallery');
    s.defaultCurrency=dollar;
    s.country=usa;
    s.canceledStatus=cos;
    s.defaultStatus=dos;
    s.defaultLanguage=english;

    // We test that a new language is set as default language
    spanish:=new Language(name="Spanish", code="ESP");
    new SetDefaultLanguage(language:=spanish) occurs;
    assert equals s.defaultLanguage spanish;
    assert not equals s.defaultLanguage english;
}

Currencies

Structural schema

osCommerce allows working with different currencies.

<table>
<thead>
<tr>
<th>Currency</th>
<th>title : String</th>
</tr>
</thead>
<tbody>
<tr>
<td>code : String</td>
<td></td>
</tr>
<tr>
<td>symbolLeft : String [0..1]</td>
<td></td>
</tr>
<tr>
<td>symbolRight : String [0..1]</td>
<td></td>
</tr>
<tr>
<td>decimalPlaces : Natural</td>
<td></td>
</tr>
<tr>
<td>value : Decimal</td>
<td></td>
</tr>
<tr>
<td>lastUpdate : DateTime [0..1]</td>
<td></td>
</tr>
<tr>
<td>status : Status</td>
<td></td>
</tr>
</tbody>
</table>

<<enumeration>>
Status
enabled
disabled
CSTL and its application to the osCommerce case study.
Albert Tort

[IC1] A currency is identified by its title and by its code.

context Currency::codeAndTitleAreUnique: Boolean
body :
    Currency.allInstances() -> isUnique(title) and
        Currency.allInstances() -> isUnique(code)

[IC2] At least one currency is enabled

context Currency::codeAndTitleAreUnique: Boolean
body :
    Currency.allInstances()->one(status=Status::enabled)

Use Cases

Add a currency

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to add a new currency.

Main Success Scenario:

1. The store administrator provides the details of the new currency:
   
   [⇒NewCurrency]

2. The system validates that the data is correct.

3. The system saves the new currency and enables it.

Edit a currency

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to edit a currency.

Main Success Scenario:

1. The store administrator selects the currency to be edited.

2. The store administrator provides the new details of the selected currency:
   
   [⇒EditCurrency]

3. The system validates that the data is correct.

4. The system saves the changes.

Delete a currency

Primary Actor: Store administrator
Precondition: There is at least another enabled currency.
Trigger: The store administrator wants to delete a currency.
Main Success Scenario:

1. The store administrator selects the currency to be deleted.
2. The store administrator confirms that he wants to delete the currency: 
   \[
   \rightarrow \text{DeleteCurrency}
   \]
3. The system deletes the currency.

Extensions:

2a. The deleted currency was the default currency.
   2a1. The system sets any of the available currencies as the default currency:
   \[
   \rightarrow \text{SetDefaultCurrency}
   \]
2b. The deleted currency is the current currency of an active session.
   2b1. The system sets any of the available currencies as the current currency:
   \[
   \rightarrow \text{SetCurrentCurrency}
   \]
2c. The currency is the currency of an order:
   2c1. The system changes the status of the currency to disable.
   \[
   \rightarrow \text{CurrencyStatusChange}
   \]
   2c2. The use case ends.

Update currencies

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to update automatically via Internet the change values for currencies.

Main Success Scenario:

1. The system connects to the change information server.
2. The value change is automatically updated for all the currencies:
   \[
   \rightarrow \text{UpdateCurrencyValueChange}
   \]

Set the default currency

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to change the default currency.

Main Success Scenario:

1. The store administrator selects the currency which will become the default currency.
2. The system updates the default currency:
   \[
   \rightarrow \text{SetDefaultCurrency}
   \]
CSTL and its application to the osCommerce case study.
Albert Tort

Events

NewCurrency

```plaintext
context NewCurrency::currencyDoesNotExist(): Boolean
body:
not Currency.allInstances() -> exists(c | c.title=self.title and c.code=self.code)
```

```plaintext
context NewCurrency::effect()
post:
c.oclIsNew() and c.oclIsTypeOf(Currency) and c.title = self.title and c.code = self.code and c.symbolLeft = self.symbolLeft and c.symbolRight = self.symbolRight and c.decimalPlaces = self.decimalPlaces and c.value = self.value and c.status = Status::enabled
```

EditCurrency

```plaintext
context EditCurrency::currencyDoesNotExist(): Boolean
body:
not ((Currency.allInstances-Set{self.currency})-exists(t | t.title=self.newTitle or t.code=self.newCode))
```

```plaintext
context EditCurrency::effect()
post:
newTitle : String
newCode : String
newSymbolLeft : String [0..1]
newSymbolRight : String [0..1]
newDecimalPlaces : Natural
newValue : Decimal
```
CSTL and its application to the osCommerce case study.
Albert Tort

context EditCurrency::effect()
post:
currency.title = self.newTitle and
currency.code = self.newCode and
currency.symbolLeft = self.newSymbolLeft and
currency.symbolRight = self.newSymbolRight and
currency.decimalPlaces = self.newDecimalPlaces and
currency.value = self.newValue

DeleteCurrency

context DeleteCurrency::AtLeastTwoCurrencies(): Boolean
body: Currency.allInstances() -> size() >= 2
context DeleteCurrency::effect()
post: not self.currency@pre.oclIsKindOf(OclAny)

SetDefaultCurrency

context SetDefaultCurrency::effect()
post: Store.allInstances() -> any(true).defaultCurrency = self.currency
CSTL and its application to the osCommerce case study.
Albert Tort

**CurrencyStatusChange**

```
context CurrencyStatusChange::atLeastOneCurrencyEnabled():Boolean
body:
  self.newStatus=Status::disabled
  implies
  (Currency.allInstances-Set{self.currency})->exists(c|c.status=Status::enabled)

context CurrencyStatusChange::effect()
post :  self.currency.status = self.newStatus
```

**UpdateCurrencyValueChange**

```
context UpdateCurrencyValueChange::effect()
post :  self.currency.value = self.newValue
post :  self.currency.lastUpdated = Now()
```

Example test program

```xml
testprogram CurrencyManagement{
test CreateCurrency{
  new NewCurrency(title:='Euro', code:='EUR', decimalPlaces:=2) occurs;
}
test CreateTheSameCurrencyTwice{
  new NewCurrency(title:='Euro', code:='EUR', decimalPlaces:=2) occurs;
  new NewCurrency(title:='Euro', code:='EUR', decimalPlaces:=2) may not occur;
}
```
CSTL and its application to the osCommerce case study.
Albert Tort

```java
test EditCurrency{
    new NewCurrency(title:='Euro', code:='EUR', decimalPlaces:=0) occurs;
    createdCurrency:=Currency.allInstances->select(title='Euro')->any(true);
    new EditCurrency(currency:=createdCurrency, newTitle:='Euro',
        newCode:='EUR', newDecimalPlaces:=2) occurs;
    assert equals createdCurrency.decimalPlaces 2;
    //Edition cannot cause duplicates
    euro:=new Currency(title:='Dollar', code:='USD', decimalPlaces:=2, status:=$enabled);
    new EditCurrency(currency:=createdCurrency, newTitle:='Euro',
        newCode:='USD', newDecimalPlaces:=2) may not occur;
}

test DeleteCurrency{
    euro:=new Currency(title:='Euro', code:='EUR', decimalPlaces:=2);
    //We cannot delete a currency if there is no other currency enabled
    new DeleteCurrency(currency:=euro) may not occur;
    new Currency(title:='Dollar', code:='USD', status:=$enabled);
    new DeleteCurrency(currency:=euro) occurs;
}

test ChangeCurrencyStatus{
    euro:=new Currency(title:='Euro', code:='EUR',
        decimalPlaces:=2,status:=$disabled);
    new CurrencyStatusChange(currency:=euro, newStatus:=$enabled) occurs;
    assert equals euro.status $enabled;
    //We cannot disable a currency if there is no other currency enabled
    new CurrencyStatusChange(currency:=euro, newStatus:=$disabled) may not occur;
}

test SetDefaultCurrency{
    //Initialize store
    franc:=new Currency(title:='Franc', code:='FR');
    french:=new Language(name:='French', code:='FR');
    cos:=new OrderStatus;
    cosl:=new OrderStatusInLanguage(language:=french, orderStatus:=cos);
    cosl.name:='annulé';
    dos:=new OrderStatus;
    dosl:=new OrderStatusInLanguage(orderStatus:=dos, language:=french);
    dosl.name:='en attente';
    s:=new Store(name:='CréaPlaisir');
    s.defaultCurrency:=franc;
    s.country:=france;
    s.cancelledStatus:=cos;
    s.defaultStatus:=dos;
    s.defaultLanguage:=french;
    //We test that a new currency is set as default currency
    euro := new Currency(title:='Euro', code:='EUR', decimalPlaces:=2);
    new SetDefaultCurrency(currency:=euro) occurs;
    assert equals s.defaultCurrency euro;
    assert not equals s.defaultCurrency franc;
}
```
Location & Taxes

Structural schema

In order to supply a flexible use of taxes, product prices are stored tax free. This allows calculating the final price of products depending on the customer's location and the tax class applied to it.

[IC1] A Country is identified either by its name or its ISO codes.

context Country::nameAndCodesAreUnique: Boolean
body :
Country.allInstances() -> isUnique (name) and
Country.allInstances() -> isUnique (isoCode2) and
Country.allInstances() -> isUnique (isoCode3)

[IC2] A Zone is identified either by its name and country or its code and country.

context Zone::nameAndCountryAndCodeAndCountryAreUnique: Boolean
body :
Zone.allInstances() -> isUnique (Tuple{n:name, c:country}) and
Zone.allInstances() -> isUnique (Tuple{n:code, c:country})

[IC3] A TaxZone is identified by its name.

context TaxZone::nameIsUnique: Boolean
body : TaxZone.allInstances() -> isUnique (name)

[IC4] A TaxClass is identified by its name

context TaxClass::nameIsUnique: Boolean
body : TaxClass.allInstances() -> isUnique (name)
CSTL and its application to the osCommerce case study. 
Albert Tort

Use Cases

Add a country

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to add a country.

Main Success Scenario:

1. The store administrator provides the details of the new country: 
   \[\rightarrow \text{NewCountry}\]
2. The system validates that the data is correct.
3. The system saves the new country.

Edit a country

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to edit a country.

Main Success Scenario:

1. The store administrator selects the country to be edited.
2. The store administrator provides the new details of the selected country:
   \[\rightarrow \text{EditCountry}\]
3. The system validates that the data is correct.
4. The system saves the changes.

Delete a country

Primary Actor: Store administrator
Precondition: The country is not the location of any address.
Trigger: The store administrator wants to delete a country.

Main Success Scenario:

1. The store administrator selects the country to be deleted.
2. The system warns the store administrator of the number of zones which are part of the country to be deleted.
3. The store administrator confirms that he wants to delete the country and their zones:
   \[\rightarrow \text{DeleteCountry}\]
4. The system deletes the country and their zones.
CSTL and its application to the osCommerce case study.
Albert Tort

### Add a zone

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to add a zone.

**Main Success Scenario:**

1. The store administrator provides the details of the new zone:  
   
   \[ \text{NewZone} \]

2. The system validates that the data is correct.
3. The system saves the new zone.

### Edit a zone

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to edit a zone.

**Main Success Scenario:**

1. The store administrator selects the zone to be edited.
2. The store administrator provides the new details of the selected zone:  
   
   \[ \text{EditZone} \]

3. The system validates that the data is correct.
4. The system saves the changes.

### Delete a zone

**Primary Actor:** Store administrator  
**Precondition:** The zone is not the location of any address.  
**Trigger:** The store administrator wants to delete a zone.

**Main Success Scenario:**

1. The store administrator selects the zone to be deleted.
2. The store administrator confirms that he wants to delete the zone:  
   
   \[ \text{DeleteZone} \]

3. The system deletes the zone.
CSTL and its application to the osCommerce case study.
Albert Tort

Events

NewCountry

```
context NewCountry::countryDoesNotExist(): Boolean
body:
    not Country.allInstances() -> exists(c | c.name=self.name and c.isoCode2=self.isoCode2 and c.isoCode3=self.isoCode3)
context NewCountry::effect()
pot:
    c.oclIsNew() and c.oclIsTypeOf(Country) and c.name = self.name and c.isoCode2 = self.isoCode2 and c.isoCode3 = self.isoCode3
```

EditCountry

```
context EditCountry::countryDoesNotExist(): Boolean
body:
    (Country.allInstances() - Set{self.country}).name->excludes(self.newName) and (Country.allInstances() - Set{self.country}).isoCode2->excludes(self.newIsoCode2)and (Country.allInstances() - Set{self.country}).isoCode3->excludes(self.newIsoCode3)
context EditCountry::effect()
pot:
    country.name = self.newName and country.isoCode2 = self.newIsoCode2 and country.isoCode3 = self.newIsoCode3
```
DeleteCountry

```
context DeleteCountry::countryIsNotALocation():Boolean
body:
  Store.allInstances() -> any(true).country <> self.country and
  Address.allInstances().country -> excludes(self.country)

context DeleteCountry::effect()
post:
  not self.country@pre.oclIsKindOf(OclAny)
```

NewZone

```
context NewZone::ZoneDoesNotExist(): Boolean
body:
  not Zone.allInstances() -> exists (z | z.name = self.name and z.country = self.country or z.code = self.code and z.country = self.country)

context NewZone::effect()
post:
  z.oclIsNew() and
  z.oclIsTypeOf(Zone) and
  z.name = self.name and
  z.code = self.code and
  z.country = self.country
```
CSTL and its application to the osCommerce case study.
Albert Tort

**EditZone**

- **Zone**
- **ExistingZoneEvent**
- **DomainEvent**

- **EditZone**
  - `newName : String`
  - `newCode : String`
  - `effect()`

```ini
context EditZone::zoneDoesNotExist(): Boolean
body: (Zone.allInstances() - Set{self.zone}).name->excludes(self.newName) and (Zone.allInstances() - Set{self.zone}).code->excludes(self.newCode)

context EditZone::effect()
post: self.zone.name = self.newName and self.zone.code = self.newCode
```

**DeleteZone**

- **Zone**
- **ExistingZoneEvent**
- **DomainEvent**

- **DeleteZone**
  - `effect()`

```ini
context DeleteZone::ZoneIsNotALocation(): Boolean
body: Store.allInstances() -> any(true).zone <> self.zone and Address.allInstances().zone -> excludes(self.zone)

context DeleteZone::effect()
post: not self.zone@pre.oclIsKindOf(OclAny)
post: self.country@pre.zone -> forAll(z | Zone.allInstances()->excludes(z))
```
CSTL and its application to the osCommerce case study.
Albert Tort

Example test programs

testprogram LocationsManagement[
  fixturecomponent DeutschlandCountryCreated{
    de:=new Country(name:'Deutschland', isoCode2:'GE', isoCode3:'DEU');
  }
  test CreateCountry{
    new NewCountry(name:'Deutschland', isoCode2:'DE', isoCode3:'DEU') occurs;
  }
  test CreateTheSameCountryTwice{
    new NewCountry(name:'Deutschland', isoCode2:'DE', isoCode3:'DEU') occurs;
    new NewCountry(name:'Deutschland', isoCode2:'DE', isoCode3:'DEU') may not occur;
  }
  test EditCountry{
    load DeutschlandCountryCreated;
    new EditCountry(country:=de,newName:'Deutschland',
      newIsoCode2:'DE', newIsoCode3:'DEU') occurs;
    assert equals de.isoCode2 'DE';
  }
  test DeleteCountryWithoutZones{
    load DeutschlandCountryCreated;
    new DeleteCountry(country:=de) occurs;
  }
  test DeleteTheCountryWhereTheStoreIsLocated{
    //Initialize store
    load DeutschlandCountryCreated;
    mark:=new Currency(title:'Mark', code:'MK');
    deutsch:=new Language(name:'Deutsch', code:'DE');
    cos:=new OrderStatus;
    cos1:=new OrderStatusInLanguage(language:=deutsch,orderStatus:=cos);
    cos1.name:'abgebrochen';
    dos:=new OrderStatus;
    dos1:=new OrderStatusInLanguage(orderStatus:=dos, language:=deutsch);
    dos1.name:'unentschieden';
    s:=new Store(name:'Geschenkwelt24');
    s.defaultCurrency:=mark;
    s.country:=de;
    s.cancelledStatus:=cos;
    s.defaultStatus:=dos;
    s.defaultLanguage:=deutsch;
    new DeleteCountry(country:=de) may not occur;
  }
  test CreateZone{
    load DeutschlandCountryCreated;
    new NewZone(country:=de,name:'Waden-Wurttemberg', code:'WW') occurs;
  }
  test CreateTheSameZoneTwice{
    load DeutschlandCountryCreated;
    ww:=new Zone(country:=de,name:'Waden', code:'WW');
    new NewZone(country:=de,name:'Waden-Wurttemberg', code:'WW') may not occur;
  }
  test EditZone{
    load DeutschlandCountryCreated;
    ww:=new Zone(country:=de,name:'Waden', code:'WW');
    new EditZone(zone:=ww, newName:'Waden-Wurttemberg', newCode:'WW') occurs;
    assert equals ww.name 'Waden-Wurttemberg';
  }
  test DeleteZone{
    load DeutschlandCountryCreated;
    new NewZone(country:=de,name:'Waden-Wurttemberg', code:'WW') occurs;
    ww:=Zone.allInstances->any(code='WW');
]
CSTL and its application to the osCommerce case study.
Albert Tort

```java
new DeleteZone(zone:=ww) occurs;
}
test DeleteCountryWithZones{
    load DeutschlandCountryCreated;
    new NewZone(country:=de, name:'Waden-Wurttemberg', code:'WW') occurs;
    new DeleteCountry(country:=de) occurs;
}
```

**Use Cases**

### Add a tax zone

**Primary Actor:** Store administrator

**Precondition:** None.

**Trigger:** The store administrator wants to add a tax zone.

**Main Success Scenario:**

1. The store administrator provides the details of the new tax zone:
   
   \[\Rightarrow \text{NewTaxZone}\]

2. The system validates that the data is correct.

3. The system saves the new tax zone.

### Edit a tax zone

**Primary Actor:** Store administrator

**Precondition:** None.

**Trigger:** The store administrator wants to edit a tax zone.

**Main Success Scenario:**

1. The store administrator selects the tax zone to be edited.

2. The store administrator provides the new details of the selected tax zone:
   
   \[\Rightarrow \text{EditTaxZone}\]

3. The system validates that the data is correct.

4. The system saves the changes.

### Delete a tax zone

**Primary Actor:** Store administrator

**Precondition:** None.

**Trigger:** The store administrator wants to delete a tax zone.

**Main Success Scenario:**

1. The store administrator selects the tax zone to be deleted.
2. The store administrator confirms that he wants to delete the tax zone:
   
   \[ \rightarrow DeleteTaxZone \]

3. The system deletes the tax zone and all the associated tax rates.

### Add a tax class

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to add a tax class.

**Main Success Scenario:**

1. The store administrator provides the details of the new tax class:
   
   \[ \rightarrow NewTaxClass \]

2. The system validates that the data is correct.

3. The system saves the new tax class.

### Edit a tax class

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to edit a tax class.

**Main Success Scenario:**

1. The store administrator selects the tax class to be edited.

2. The store administrator provides the new details of the selected tax class:
   
   \[ \rightarrow EditTaxClass \]

3. The system validates that the data is correct.

4. The system saves the changes.

### Delete a tax class

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to delete a tax class.

**Main Success Scenario:**

1. The store administrator selects the tax class to be deleted.

2. The system informs the store administrator about how many products are associated to the deleted tax class.

3. The store administrator confirms that he wants to delete the tax class:
   
   \[ \rightarrow DeleteTaxClass \]
4. The system deletes the tax class and all the associated tax rates.

**Extensions:**

2a. The store administrator don’t want to delete the tax class.
   2a1. The use case ends.

### Add a tax rate

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to add a tax rate.

**Main Success Scenario:**

1. The store administrator provides the details of the new tax rate:  
   - \( \Rightarrow \text{NewTaxRate} \)  
2. The system validates that the data is correct.  
3. The system saves the new tax rate.

### Edit a tax rate

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to edit a tax rate.

**Main Success Scenario:**

1. The store administrator selects the tax rate to be edited.  
2. The store administrator provides the new details of the selected tax rate:  
   - \( \Rightarrow \text{EditTaxRate} \)  
3. The system validates that the data is correct.  
4. The system saves the changes.

### Delete a tax rate

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to delete a tax rate.

**Main Success Scenario:**

1. The store administrator selects the tax rate to be deleted.  
2. The store administrator confirms that he wants to delete the tax rate:
CSTL and its application to the osCommerce case study.
Albert Tort

[\( \Rightarrow \text{DeleteTaxRate} \)]

3. The system deletes the tax rate.

**Events**

**NewTaxZone**

```mermaid
classDiagram
    DomainEvent <<interface>> NewTaxZone
    NewTaxZone -- Zone
    NewTaxZone "name : String description : String [0..1] effect()"

    context NewTaxZone::TaxZoneDoesNotExist(): Boolean
    body : not TaxZone.allInstances() -> exists (tz | tz.name = self.name)

    context NewTaxZone::effect()
    post :
      tz.oclIsNew() and
tz.oclIsTypeOf(TaxZone) and
tz.name = self.name and
tz.description = self.description and
tz.zone = self.zone
```

**EditTaxZone**

```mermaid
classDiagram
    DomainEvent <<interface>> EditTaxZone
    EditTaxZone -- Zone
    TaxZone "newName : String newDescription : String [0..1] effect()"

    context EditTaxZone::TaxZoneDoesNotExist(): Boolean
    body : (TaxZone.allInstances() - Set{self.taxZone}).name->excludes(self.newName)

    context EditTaxZone::effect()
    post :
      self.taxZone.name = self.newName and
self.taxZone.description = self.newDescription and
self.taxZone.zone = self.newZones
```

```
```
DeleteTaxZone

```
context DeleteTaxZone::effect()
post deleteTaxZone:
    not self.taxZone@pre.oclIsKindOf(OclAny)
post deleteAssociatedTaxRates:
    self.taxZone@pre.taxRate@pre -> forAll(tr | tr.oclIsKindOf(OclAny))
```

NewTaxClass

```
context NewTaxClass::TaxClassDoesNotExist(): Boolean
body : not TaxClass.allInstances() -> exists (tc | tc.name = self.name)
context NewTaxClass::effect()
post :
    tc.oclIsNew() and
    tc.oclIsTypeOf(TaxClass) and tc.name = self.name and tc.description = self.description
```

EditTaxClass

```
context EditTaxClass::effect()
post :
    tc.oclsIsNew() and
    tc.oclsIsTypeOf(TaxClass) and tc.name = self.name and tc.description = self.description
```
CSTL and its application to the osCommerce case study.
Albert Tort

`<InitC>`
context EditTaxClass::TaxClassDoesNotExist(): Boolean
body : (TaxClass.allInstances() - Set{self.taxClass}).name->excludes(self.newName)

context EditTaxClass::effect()
post:
  self.taxClass.name = self.newName and self.taxClass.description = self.newDescription

DeleteTaxClass

context DeleteTaxClass::effect()
post deleteTaxClass:
  not self.taxClass@pre.oclIsKindOf(OclAny)
post deleteAssociatedTaxRates:
  self.taxClass@pre.taxRate@pre -> forAll(tr | tr.oclIsKindOf(OclAny))

NewTaxRate

context NewTaxRate::TaxRateDoesNotExist(): Boolean
body:
  not TaxRate.allInstances() -> exists (tr | tr.taxClass = self.taxClass and tr.taxZone = self.taxZone)

context NewTaxRate::effect()
post:
  tr.oclIsNew() and tr.oclIsTypeOf(TaxRate) and tr.rate = self.rate and tr.priority = self.priority and tr.description = self.description and tr.taxClass = self.taxClass and tr.taxZone = self.taxZone
CSTL and its application to the osCommerce case study.
Albert Tort

**EditTaxRate**

```
context EditTaxRate::TaxRateDoesNotExist(): Boolean
body: (TaxRate.allInstances - Set{self.taxRate})->select(tr | tr.taxClass = self.newTaxClass and tr.taxZone = self.newTaxZone) -> size()=0
```  

```
context EditTaxRate::effect()
post: self.taxRate.rate = self.newRate and self.taxRate.priority = self.newPriority and self.taxRate.description = self.newDescription and self.taxRate.taxClass = self.newTaxClass and self.taxRate.taxZone = self.newTaxZone
```

**DeleteTaxRate**

```
context DeleteTaxRate::effect()
post: not self.taxRate@pre.oclIsKindOf(OclAny)
```  

**Example test programs**

```
testprogram TaxesConfigurationManagement{
catalonia:=new Zone(name:='Catalonia', code:='CAT', country:=spain);
```
CSTL and its application to the osCommerce case study.
Albert Tort

andalucia:=new Zone(name:='Andalucia', code:='AND', country:=spain);
zones:=spain.zone;

test AddTaxZone{
    new NewTaxZone(name:='SpanishVAT', zone:=catalonia, andalucia) occurs;
}

test EditTaxZone{
    zones:=spain.zone;
    tz:=new TaxZone(name:='SpanishVAT', zone:=zones);
    new EditTaxZone(taxZone:=tz, newName:='SpanishVAT', newZones:=catalonia) occurs;
    assert true tz.zone->excludes(andalucia);
    assert true tz.zone->includes(catalonia);
}

test DeleteTaxZoneWithoutTaxRates{
    tz:=new TaxZone(name:='SpanishVAT', zone:=zones);
    new DeleteTaxZone(taxZone:=tz) occurs;
}

test DeleteTaxZoneWithTaxRates{
    tz:=new TaxZone(name:='SpanishVAT', zone:=zones);
    tc:=new TaxClass(name:='GeneralVAT');
    tc2:=new TaxClass(name:='ReducedVAT');
    new TaxRate(taxClass:=tc,taxZone:=tz);
    new TaxRate(taxClass:=tc2,taxZone:=tz);
    new DeleteTaxZone(taxZone:=tz) occurs;
}

test AddTaxClass{
    new NewTaxClass(name:='SpanishVAT');
    new NewTaxClass(name:='SpanishVAT') may not occur;
}

test EditTaxClass{
    tc:=new TaxClass(name:='VAT');
    new EditTaxClass(taxClass:=tc,newName:='GeneralVAT') occurs;
}

test DeleteTaxClassWithoutZoneRates{
    tc:=new TaxClass(name:='GeneralVAT');
    new DeleteTaxClass(taxClass:=tc) occurs;
}

test DeleteTaxClassWithZoneRates{
    tz:=new TaxZone(name:='SpanishVAT', zone:=zones);
    tc:=new TaxClass(name:='GeneralVAT');
    new TaxRate(taxClass:=tc,taxZone:=tz);
    new DeleteTaxClass(taxClass:=tc) occurs;
}

test AddTaxRate{
    tz:=new TaxZone(name:='SpanishVAT', zone:=zones);
    tc:=new TaxClass(name:='GeneralVAT');
    new NewTaxRate(taxClass:=tc, taxZone:=tz, rate:=16, priority:=1) occurs;
}

test EditTaxRate{
    tz:=new TaxZone(name:='SpanishVAT', zone:=zones);
    tc:=new TaxClass(name:='GeneralVAT');
    tc2:=new TaxClass(name:='ReducedVAT');
    tr:=new TaxRate(taxClass:=tc,taxZone:=tz);
    tr.rate:=7;
    new EditTaxRate(taxRate:=tr,newTaxClass:=tc2,newTaxZone:=tz,newRate:=7) occurs;
}

test DeleteTaxRate{
    tz:=new TaxZone(name:='SpanishVAT', zone:=zones);
    tc:=new TaxClass(name:='GeneralVAT');
    tr:=new TaxRate(taxClass:=tc,taxZone:=tz);
    new DeleteTaxRate(taxRate:=tr) occurs;
}
CSTL and its application to the osCommerce case study.
Albert Tort

```java
// This test program checks that the default gross price (shown in the online store) of a product is well-calculated. The default gross price is calculated by taking into account the zone where the store is located

// FIXTURE
// Languages
english = new Language(name = 'English', code = 'EN');
spanish = new Language(name = 'Spanish', code = 'ES');

// Currencies
cad = new Currency(title = 'Canadian Dollar', code = 'CAD');
eur = new Currency(title = 'Euro', code = 'EUR');

// Countries
canada = new Country(name = 'Canada', isoCode2 = 'CA', isoCode3 = 'CAN');
spain = new Country(name = 'Spain', isoCode2 = 'ES', isoCode3 = 'ESP');

// Zones
andalucia = new Zone(name = 'Andalucia', code = 'AND', country = spain);
ontario = new Zone(name = 'Ontario', code = 'ONT', country = canada);
quebec = new Zone(name = 'Quebec', code = 'QUE', country = canada);

// Order Status
cos = new OrderStatus;
cosInEnglish = new OrderStatusInLanguage(language = english, orderStatus = cos);
cosInSpanish = new OrderStatusInLanguage(language = spanish, orderStatus = cos);

// FIXTURE COMPONENTS
// We create two different shop configurations:
// A canadian store (with only one tax class)
// An spanish store (with three different tax classes)
// We apply them in the test cases two check the gross price calculation in different tax configurations

fixturecomponent CanadianStoreInitialization{
    // Store initialization
    s = new Store(name = 'CanadianStore');
s.defaultLanguage = english;
s.defaultCurrency = cad;
s.country = canada;
s.cancelledStatus = cos;
s.defaultStatus = dos;

    // Tax configuration
    // We create a tax zone for Canada
    TaxZone canadaFederalTaxes = new TaxZone(name = 'Canada Federal Taxes');
    canadaFederalTaxes.zone = quebec, ontario;

    // We create an specific tax zone for Quebec
    TaxZone quebecLocalTaxes = new TaxZone(name = 'Quebec LocalTaxes');
    quebecLocalTaxes.zone = quebec;

    // We consider a single tax class
    TaxClass general = new TaxClass(name = 'general');

    // For each TaxClass, there is a different tax rate applied in each zone
    TaxRate canadianFederalTaxRate = new TaxRate(taxClass = general, taxZone = canadaFederalTaxes);
    canadianFederalTaxRate.rate = 7;
    canadianFederalTaxRate.priority = 1;

    quebecLocalTaxRate = new TaxRate(taxClass = general, taxZone = quebecLocalTaxes);
    quebecLocalTaxRate.rate = 7.5;
    quebecLocalTaxRate.priority = 2;
}

fixturecomponent SpanishStoreInitialization{
    // Store initialization
    s = new Store(name = 'SpanishStore');
```
CSTL and its application to the osCommerce case study.
Albert Tort

```plaintext
s.defaultLanguage:=spanish;
s.defaultCurrency:=cad;
s.country:=spain;
s.cancelledStatus:=cos;
s.defaultStatus:=dos;

//We create a specific tax zone
TaxZone spanishVAT:=new TaxZone(name:='SpanishVAT',
  description:='This zone includes all VAT varieties applied in Spain');
spainVAT.zone:=andalucia;

//In Spain there are three types of VAT: general VAT (16%),
//reduced VAT(7%) and super-reduced VAT(4%)
TaxClass general:=new TaxClass(name:='General VAT');
TaxClass reduced:=new TaxClass(name:='ReducedVAT');
TaxClass superreduced:=new TaxClass(name:='Super-reduced VAT');

//For each TaxClass, there is a different tax rate applied in each zone
TaxRate generalRate:=new TaxRate(taxClass:=general, taxZone:=spanishVAT);
generalRate.rate:=16;
generalRate.priority:=1;
TaxRate reducedRate:=new TaxRate(taxClass:=reduced, taxZone:=spanishVAT);
reducedRate.rate:=7;
reducedRate.priority:=1;
TaxRate superReducedRate:=new TaxRate(taxClass:=superreduced, taxZone:=spanishVAT);
superReducedRate.rate:=4;
superReducedRate.priority:=1;
}

test DefaultGrossPriceWithDifferentTaxClasses{
  load SpanishStoreInitialization;

  //We locate the store in the zone Andalucia
  s.zone := andalucia;

  //The reduced VAT is applied to cultural events, among others products
  Product greaseMusicalAdmission:=new Product(netPrice:=50);
  greaseMusicalAdmission.taxClass:=reduced;
  assert equals greaseMusicalAdmission.grossPrice() 53.5;

  //The super-reduced VAT is applied to books, among other products
  Product angelsAndDemonsBook:=new Product(netPrice:=25);
  angelsAndDemonsBook.taxClass:=superreduced;
  assert equals angelsAndDemonsBook.grossPrice() 26.0;

  //The general VAT is applied to those products which are not basic needs or
  //cultural products
  Product whiteWineBottle:= new Product(netPrice:=11);
  whiteWineBottle.taxClass:=general;
  assert equals whiteWineBottle.grossPrice() 12.76;
}

test DefaultGrossPriceInDifferentShopLocations{
  //We test that the gross price (netPrice + taxes) of
  //a product is different depending on the store location and the
  //taxes configuration."
  load CanadianStoreInitialization;

  //We create the example product
  Product theDaVinciCodeBook:= new Product(netPrice:=50);
  theDaVinciCodeBook.taxClass:=general;

  //First, we locate the store in the zone Ontario
  s.zone:=ontario;
  assert equals theDaVinciCodeBook.grossPrice() 53.5;

  //If the store is located in Quebec, the gross price
  //also takes into account the Quebec Local Tax which is
  //compounded with the Federal Tax"
  s.zone:=quebec;
  assert equals theDaVinciCodeBook.grossPrice() 57.5125;
}
```
CSTL and its application to the osCommerce case study.
Albert Tort

Products

Structural schema

The system must know the information about the products offered by the online store.

<table>
<thead>
<tr>
<th>Product</th>
<th>status : ProductStatus</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>available : DateTime [0..1]</td>
</tr>
<tr>
<td></td>
<td>netPrice : Money</td>
</tr>
<tr>
<td></td>
<td>specialNetPrice [0..1]</td>
</tr>
<tr>
<td></td>
<td>grossPrice : Money</td>
</tr>
<tr>
<td></td>
<td>quantityOnHand : Integer</td>
</tr>
<tr>
<td></td>
<td>model : String [0..1]</td>
</tr>
<tr>
<td></td>
<td>imagePath : String [0..1]</td>
</tr>
<tr>
<td></td>
<td>weight : Decimal &lt;&lt;constant&gt;&gt;</td>
</tr>
</tbody>
</table>

context Product def:
addTaxes(z:Zone, basePrice:Money) : Money =
  let appliedTaxRates:Set(TaxRate) =
    z.taxZone.taxRate -> select (tr | tr.taxClass = self.taxClass) in
  let priorities:Set(Natural) =
    if appliedTaxRate -> isEmpty() then set{}
    else appliedTaxRates -> sortedBy(priority).priority -> asSet()
  endif in
  if priorities -> isEmpty() then basePrice
  else priorities -> iterate (p:Natural; res:Money = 0 |
    res + (((appliedTaxRates -> select (tr | tr.priority = p).rate -> sum()) / 100)+1)*basePrice)
  endif

[DR1] Product::grossPrice is the product’s netPrice taking into account the applied taxes.

context Product::grossPrice(): Money
body : self.addTaxes(Store.allInstances() -> any(true).zone, self.netPrice)

[DR2] Product::specialNetPrice is the special price, if the product is an active special.

context Product::specialNetPrice(): Money
body :
  if self.oclIsTypeOf(Special) then
    if self.oclAsType(Special).specialStatus=Status::enabled and
      self.oclAsType(Special).expiryDate < Now() then self.oclAsType(Special).specialPrice
    else set{}
  endif
  else set{}
endif
CSTL and its application to the osCommerce case study.
Albert Tort

**[DR3]** Product::added is the DateTime of product creation.

**context** Product::added(): DateTime
**body :** Now()

**[IC1]** A product is identified by a name in a language.

**context** Language::nameIsUnique(): Boolean
**body :**
Language.allInstances->forAll(l | l.productInLanguage->isUnique(name))

### Use cases

#### Add a product

**Primary Actor:** Store administrator

**Precondition:** None.

**Trigger:** The store administrator wants to add a product to the store catalog.

**Main Success Scenario:**

1. The store administrator selects the product category.
2. The store administrator provides the product data:
   
   [⇒ NewProduct]

3. The system validates that the data is correct.
4. The system saves the new product.

5. The store administrator provides a product attribute:
   
   [⇒ NewProductAttribute]

6. The system validates that the product attribute is correct.
7. The system saves the new product attribute.

   The store administrator repeats steps 5-7 until he is done.

**Extensions:**

5a. The product does not have product attributes:
   
   5a1. The use case ends.
5b. The product option is new:
   
   5b1. Add a product option.
5c. The product option value is new:
   
   5c1. Add a product option value.
CSTL and its application to the osCommerce case study.
Albert Tort

### Edit a product

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to edit a product.

#### Main Success Scenario:

1. The store administrator selects the product to be edited.  
2. The store administrator provides the new values for the attributes of the product:  
   
   
   \[ \rightarrow \text{EditProduct} \]

3. The system validates that the data is correct.  
4. The system saves the changes.

### Delete a product

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to delete a product.

#### Main Success Scenario:

1. The store administrator selects the product to be deleted.  
2. The system asks for the confirmation of the store administrator.  
3. The store administrator confirms that he wants to delete the product:  
   
   
   \[ \rightarrow \text{DeleteProduct} \]

4. The system deletes the product and their product attributes.

#### Extensions:

3a. The product is part of an order:  
   3a1. The system changes the status of the product to *out of stock*.  
      
      \[ \rightarrow \text{ProductStatusChange} \]

   3a2. The use case ends.
CSTL and its application to the osCommerce case study.
Albert Tort

Events

NewProduct

context NewProduct::productDoesNotExist(): Boolean
body:
Language.allInstances() -> forAll ( l |
  l.productInLanguage.name
  -> excludes( self.hasNewProductName -> select(language=l).name))

context NewProduct::effect()
post:
  p.oclIsNew() and
  p.oclIsTypeOf(Product) and
  p.status = self.status and
  p.available = self.available and
  p.netPrice = self.netPrice and
  p.quantityOnHand = self.quantityOnHand and
  p.model = self.model and
  p.imagePath = self.imagePath and
  p.weight = self.weight and
  p.category = Set{self.category} and
  p.manufacturer = self.manufacturer and
  p.taxClass = self.taxClass and
  Language.allInstances() ->
    forAll ( l |
      self.hasNewProductName -> select(language=l).name =
      p.productInLanguage->select(language=l).name)
CSTL and its application to the osCommerce case study.
Albert Tort

**EditProduct**

```
context EditProduct::productDoesNotExist(): Boolean
body: Language.allInstances() -> forAll ( l |
  l.productInLanguage.name -> excludes(self.hasNewProductName -> any(languageOfProduct=l).nameOfProduct) or
  (self.hasNewProductName->any(languageOfProduct=l).nameOfProduct =
   self.product.productInLanguage->any(language=l).name))

context EditProduct::effect()
post : self.product.status = self.status and
  self.product.available = self.available and
  self.product.netPrice = self.netPrice and
  self.product.quantityOnHand = self.quantityOnHand and
  self.product.model = self.model and
  self.product.imagePath = self.imagePath and
  self.product.weight = self.weight and
  self.product.manufacturer = self.manufacturer and
  self.product.category = self.category and
  self.product.taxClass = self.taxClass and
  Language.allInstances() -> forAll ( l |
    self.hasNewProductName -> select(language=l).name =
     self.product.productInLanguage->select(language=l).name)

post: self.product.lastModified = Now()
```
CSTL and its application to the osCommerce case study.
Albert Tort

DeleteProduct

\[\text{context } \text{DeleteProduct::effect() }\]
\[\text{post: }\]
\[\begin{aligned}
\text{if } & \text{product@pre.orderLine -> size()}=0 \\
\text{then } & \text{Product.allInstances}->\text{excludes(product@pre)} \\
\text{else } & \text{psc.oclIsNew()} \text{ and} \\
& \text{psc.oclIsTypeOf(ProductStatusChange) and} \\
& \text{psc.newStatus} = \text{Status::outOfStock \text{ and}} \\
& \text{psc.product} = \text{self.product@pre}
\end{aligned}\]
\[\text{endif}\]

ProductStatusChange

\[\text{context } \text{ProductStatusChange::effect() }\]
\[\text{post: }\]
\[\text{self.product.status} = \text{self.newStatus}\]
Example test programs

testprogram AddNewProducts{

    // Test cases are based on a multilingual online shop with two languages
    italian := new Language(name="Italian", code="IT");
    english := new Language(name="English", code="EN");

    test NewProductWithoutNames{
        new NewProduct(netPrice:=30, quantityOnHand:=50) may not occur;
    }

    test NewProductWithoutNamesForSomeLanguages{
        // We should specify the product name in each language
        s:=new StringDT(string="Extra Virgin Oil Jar");
        np:=new NewProduct(netPrice:=10, quantityOnHand:=50);
        new HasNewProductName(nameOfProduct:=s,
            languageOfProduct:=english, productNameEvent:=this));
        np may not occur;
    }

    test NewProductWithAllNamesSpecified{
        // We test a valid invocation of the event
        englishName:=new StringDT(string="Extra Virgin Oil Jar");
        italianName:=new StringDT(string="Giara di olio");
        np:=new NewProduct(netPrice:=10, quantityOnHand:=50);
        new HasNewProductName(nameOfProduct:=italianName,
            languageOfProduct:=italian, productNameEvent:=this);
        new HasNewProductName(nameOfProduct:=englishName,
            languageOfProduct:=english, productNameEvent:=this));
        np occurs;
        createdProduct := Product.allInstances
        ->any(productInLanguage
            ->exists(name='Extra Virgin Oil Jar'));

        // Although postconditions are checked,
        // we ensure that we can get the product name in each language
        assert equals createdProduct.productInLanguage->any(language=english).name
            'Extra Virgin Oil Jar';
        assert equals createdProduct.productInLanguage->any(language=italian).name
            'Giara di olio';
    }

    test NewProductWithEqualNamesInSomeLanguages{
        // osCommerce allows the same product name for different languages
        s:=new StringDT(string="Lemoncello");
        np:=new NewProduct(netPrice:=30, quantityOnHand:=50);
        new HasNewProductName(nameOfProduct:=s,
            languageOfProduct:=italian, productNameEvent:=this);
        new HasNewProductName(nameOfProduct:=s,
            languageOfProduct:=english, productNameEvent:=this));
        np occurs;
    }

    test NewProductThatAlreadyExists{
        // IB state with a product
        acetoAromatizzato:=new Product(netPrice:=4, quantityOnHand:=70);
        productInItalian:=new ProductInLanguage
            (product:=acetoAromatizzato, language:=italian);
        productInItalian.name:='Aceto aromatizzato';
        productInEnglish:=new ProductInLanguage
            (product:=acetoAromatizzato, language:=english);
        productInEnglish.name:='Spicy wine vinegar';

        // The creation of a product with the same name in at least one
        // language should not occur
        italianName:=new StringDT(string="Aceto aromatizzato");
        englishName:=new StringDT(string="Spicy wine vinegar");
        differentName:=new StringDT(string="AnyName");
        np:=new NewProduct(netPrice:=10, quantityOnHand:=50);
        new HasNewProductName(nameOfProduct:=italianName,
            languageOfProduct:=italian, productNameEvent:=this);
        new HasNewProductName(nameOfProduct:=differentName,
            languageOfProduct:=english, productNameEvent:=this));
    }
}
CSTL and its application to the osCommerce case study.
Albert Tort

```plaintext
np may not occur;
np2:=new NewProduct(netPrice:=10,quantityOnHand:=50);
new HasNewProductName(nameOfProduct:=differentName,
    languageOfProduct:=italian,productNameEvent:=this);
new HasNewProductName(nameOfProduct:=englishName,
    languageOfProduct:=english,productNameEvent:=this));
np2 may not occur;
np3:=new NewProduct(netPrice:=10,quantityOnHand:=50);
new HasNewProductName(nameOfProduct:=italianName,
    languageOfProduct:=italian,productNameEvent:=this);
new HasNewProductName(nameOfProduct:=englishName,
    languageOfProduct:=english,productNameEvent:=this));
np3 may not occur;
}
```

testprogram EditProducts{

```plaintext
english := new Language(name:='English', code:='EN');
necklace:=new Product(netPrice:=4, quantityOnHand:=70, status:=#outOfStock);
productInEnglish:=new ProductInLanguage (product:=necklace, language:=english);
productInEnglish.name:='Necklace';

test EditProductStatus{
    englishName:=new StringDT(string:='Necklace');
    ep:=new EditProduct(product:=necklace,status:=#inStock,
        netPrice:=10,quantityOnHand:=50);
    new HasNewProductName(nameOfProduct:=englishName,
        languageOfProduct:=english,productNameEvent:=this));
    ep occurs;
    assert equals necklace.status #inStock;
}

test EditProductNameInALanguage{
    englishName:=new StringDT(string:='GoldNecklace');
    ep:=new EditProduct(product:=necklace,status:=#inStock,
        netPrice:=10,quantityOnHand:=50);
    new HasNewProductName(nameOfProduct:=englishName,
        languageOfProduct:=english,productNameEvent:=this));
    ep occurs;
}

test UnapplicableProductEdition{
    //IB state with a product
    goldnecklace:=new Product(netPrice:=4, quantityOnHand:=70, status:=#inStock);
    productInEnglish:=new ProductInLanguage (product:=goldnecklace, language:=english);
    productInEnglish.name:='Gold Necklace';
    //A product edition the effect of which violates the product identification
    //constraint cannot occur
    englishName:=new StringDT(string:='GoldNecklace');
    ep:=new EditProduct(product:=necklace,status:=#inStock,
        netPrice:=10,quantityOnHand:=50);
    new HasNewProductName(nameOfProduct:=englishName,
        languageOfProduct:=english,productNameEvent:=this));
    ep occurs;
}
```

testprogram DeleteProduct{

```plaintext
english := new Language(name:='English', code:='EN');
necklace:=new Product(netPrice:=4, quantityOnHand:=70, status:=#outOfStock);
productInEnglish:=new ProductInLanguage (product:=necklace, language:=english);
productInEnglish.name:='Necklace';

test DeleteProductNotSoldYet{
    new DeleteProduct(product:=necklace) occurs;
    assert true Product.allInstances->excludes(necklace);
}
```

Product attributes and options

Structural schema

osCommerce allows defining several attributes for each product. Product attributes are used to offer multiple options of a product.
CSTL and its application to the osCommerce case study.
Albert Tort

[IC1] In each language, each product option has a unique name.

\[
\text{context Language::optionNamesUnique(): Boolean} \\
\text{body : self.hasOptionName -> isUnique(optionName)}
\]

[IC2] In each language, each product value has a unique name.

\[
\text{context Language::valueNamesUnique(): Boolean} \\
\text{body : self.hasOptionValue -> isUnique(valueName)}
\]

Use cases

Add a product option

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to add a product option to the store catalog.

Main Success Scenario:

1. The store administrator provides the product option data:
   \[→NewProductOption\]
2. The system validates that the data is correct.
3. The system saves the new product option.

Edit a product option

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to edit a product option.

Main Success Scenario:

1. The store administrator selects the product option to be edited.
2. The store administrator provides the new details of the selected product option:
   \[→EditProductOption\]
3. The system validates that the data is correct.
4. The system saves the changes.

Delete a product option

Primary Actor: Store administrator
Precondition: The product option has no associated products.
Trigger: The store administrator wants to delete a product option.

Main Success Scenario:
CSTL and its application to the osCommerce case study.
Albert Tort

1. The store administrator selects the product option to be deleted.
2. The system asks for the confirmation of the store administrator.
3. The store administrator confirms that he wants to delete the product option:
   
   \[ \rightarrow \text{DeleteProductOption} \]
4. The system deletes the product option and its associated values if they are not values of other options.

**Add a product option value**

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to add a value to a product option.

**Main Success Scenario:**

1. The store administrator selects the product option.
2. The store administrator provides the product option value data:
   
   \[ \rightarrow \text{NewProductOptionValue} \]
3. The system validates that the data is correct.
4. The system saves the new product option value.

**Edit a product option value**

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to edit a product option value.

**Main Success Scenario:**

1. The store administrator selects the product option value to be edited.
2. The store administrator provides the new details of the selected product option value:
   
   \[ \rightarrow \text{EditProductOptionValue} \]
3. The system validates that the data is correct.
4. The system saves the changes.

**Delete a product option value**

**Primary Actor:** Store administrator  
**Precondition:** The product option value has not products linked to it.  
**Trigger:** The store administrator wants to delete a product option value.

**Main Success Scenario:**

1. The store administrator selects the product option value to delete.
CSTL and its application to the osCommerce case study.
Albert Tort

2. The system asks for the confirmation of the store administrator.
3. The store administrator confirms that he wants to delete the product option value:
   
   \[ \Rightarrow \text{DeleteProductOptionValue} \]
4. The system deletes the product option value.

### Add a product attribute

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to assign an attribute to a product.

**Main Success Scenario:**

1. The store administrator selects the product.
2. The store administrator provides the attribute and the product attribute data (increment and sign):
   
   \[ \Rightarrow \text{NewProductAttribute} \]
   \[ \Rightarrow \text{NewDownloadableProductAttribute} \]
3. The system validates that the data is correct.
4. The system saves the new product attribute.

**Extensions:**

2a. The product option is new:
   
   2a1. Add a product option.
   
2b. The product option value is new:
   
   2b1. Add a product option value.

### Edit a product attribute

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to edit a product attribute.

**Main Success Scenario:**

1. The store administrator selects the product attribute to be edited.
2. The store administrator provides the new details for the product attribute:
   
   \[ \Rightarrow \text{AttributeChange} \]
   \[ \Rightarrow \text{IncrementAndSignAttributeChange} \]
   \[ \Rightarrow \text{EditDownloadableAttribute} \]
3. The system validates that the data is correct.
4. The system saves the changes.

The system repeats steps 2-4 until he is done.
**Delete a product attribute**

**Primary Actor:** Store administrator

**Precondition:** None.

**Trigger:** The store administrator wants to delete a product attribute.

**Main Success Scenario:**

1. The store administrator selects the product attribute to be deleted.
2. The system asks for the confirmation of the store administrator.
3. The store administrator confirms that he wants to delete the product attribute:
   
   ![DeleteProductAttribute](image)

4. The system deletes the product attribute.

**Extensions:**

3a. The product attribute is part of an existing order line:

   3a1. The system changes the status of the product attribute to disable.

   ![ProductAttributeStatusChange](image)

   3a2. The use case ends

**Events**

**NewProductAttribute**

```plaintext
context NewProductAttribute::productAttributeDoesNotExist(): Boolean
body:
  not self.product.productAttribute
  -> exists(attribute.value=self.value and
           attribute.option = self.option)

context NewProductAttribute::optionValuesIsValid(): Boolean
body:
  self.option.value -> includes(self.value)
```
CSTL and its application to the osCommerce case study.
Albert Tort

context NewProductAttribute::effect()
post:
pa.oclIsNew() and
pa.oclIsTypeOf(ProductAttribute) and
pa.increment = self.increment and
pa.sign = self.sign and
pa.product = self.product and
pa.attribute.option = self.option and
pa.attribute.value = self.value

NewProductOption

context NewProductOption::productOptionDoesNotExist(): Boolean
body:
Language.allInstances() -> forAll ( l |
  l.hasOptionName.optionName
  -> excludes(self.hasNewOptionName -> select(language=l).name))

context NewProductOption::effect()
post:
po.oclIsNew() and
po.oclIsTypeOf(Option) and
Language.allInstances() ->
  forAll ( l | self.hasNewOptionName -> select(language=l).name =
    po.hasOptionName -> select(optionLanguage=l).optionName)

EditProductOption

context EditProductOption::effect()
post:
       po.oclIsNew() and
       po.oclIsTypeOf(Option) and
language.allInstances() ->
  forAll ( l | self.hasNewOptionName -> select(language=l).name =
    po.hasOptionName -> select(optionLanguage=l).optionName)
CSTL and its application to the osCommerce case study.
Albert Tort

context EditProductOption::OptionDoesNotExist(): Boolean
body: Language.allInstances() -> forAll (l | l.hasOptionName.optionName
  -> excludes(self.hasNewOptionName -> any(languageOfOption=l).nameOfOption
  or
  (self.hasNewOptionName->any(languageOfOption=l).nameOfOption =
  self.option.hasOptionName->any(optionLanguage=l).optionName))

context EditProductOption::effect()
post: Language.allInstances() -> forAll (l | self.hasNewOptionName -> select(language=l).name =
  option.hasOptionName->select(language=l).optionName)

DeleteProductAttribute

calendar

context DeleteProductAttribute::effect()
post: if OrderLineAttribute.allInstances() -> exists(ola |
  ola.attribute=productAttribute.attribute and
  ola.orderLine.product=productAttribute.product)
  then productAttribute.status=Status::disabled
  else ProductAttribute.allInstances->excludes(productAttribute@pre)
endif

NewProductOptionValue

context NewProductOptionValue::optionValueDoesNotExist(): Boolean
body: Language.allInstances() -> forAll (l | l.hasValueName.valueName
  -> excludes(self.hasNewValueName -> select(language=l).name))
CSTL and its application to the osCommerce case study.
Albert Tort

**context** NewProductOptionValue::effect()
**post**:
pov.oclIsNew() and/ppov.oclIsTypeOf(Value) and
/\Language.allInstances() -> forAll (l| self.hasNewValueName -> select(language=l).name =
pov.hasValueName->select(valueLanguage=l).valueName) and
/\pov.option = self.option

**EditProductOptionValue**

**context** EditProductOptionValue::productOptionValueDoesNotExist(): Boolean
**body**:
Language.allInstances() -> forAll ( l |
l.hasValueName.valueName
- > excludes(self.hasNewValueName -> any(language=l).name) or
(self.hasNewValueName->any(language=l).name =
self.value.hasValueName->any(valueLanguage=l).valueName))

**context** EditProductOptionValue::effect()
**post**:
Language.allInstances() -> forAll ( l |
\self.hasNewValueName->select(language=l).name =
\value.hasValueName->select(language=l).valueName) and
self.value.option = self.option

**DeleteProductOptionValue**

**context** DeleteProductOptionValue::HasNotProducts(): Boolean
**body**:
self.value.attribute.product -> isEmpty() and self.value.attribute.orderLineAttribute->isEmpty()
CSTL and its application to the osCommerce case study.
Albert Tort

context DeleteProductOptionValue::effect()
post : not self.value@pre.oclIsKindOf(OclAny)

NewProductAttribute

context NewProductAttribute::productAttributeDoesNotExist(): Boolean
body : not self.product.productAttribute -> exists(attribute.value=self.value and attribute.option = self.option)

context NewProductAttribute::optionValuesIsValid(): Boolean
body : self.option.value -> includes(self.value)

context NewProductAttribute::effect()
post : pa.oclIsNew() and pa.oclIsTypeOf(ProductAttribute) and pa.increment = self.increment and pa.sign = self.sign and pa.product = self.product and pa.attribute.option = self.option and pa.attribute.value = self.value

NewDownloadableProductAttribute
CSTL and its application to the osCommerce case study.
Albert Tort

```cstl
context NewDownloadableProductAttribute::productAttributeDoesNotExist(): Boolean
body:
  not ProductAttribute.allInstances() -> exists (pa | pa.attribute.option = self.option and
  pa.attribute.value = self.value and
  pa.product = self.product)

context NewDownloadableProductAttribute::effect()
post:
  dpa.oclIsNew() and
  dpa.oclIsTypeOf(Downloadable) and
  dpa.increment = self.increment and
  dpa.sign = self.sign and
  dpa.filename = self.filename and
  dpa.product = self.product and
  dpa.attribute.option = self.option and
  dpa.attribute.value = self.value and
  if self.expiryDays.notEmpty() then dpa.expiryDays = self.expiryDays
  else self.expiryDays = Download.daysExpiryDelay
  endif
  and
  if self.maximumDownloadCount.notEmpty() then
    dpa.maximumDownloadCount = self.maximumDownloadCount
  else self.maximumDownloadCount = Download.maximumNumberOfDownloads
  endif

AttributeChange

```
CSTL and its application to the osCommerce case study.  
Albert Tort

**IncrementAndSignAttributeChange**

```plaintext
context IncrementAndSignAttributeChange::effect()
post:  self.productAttribute.increment = self.newIncrement and
       self.productAttribute.sign = self.newSign
```

**EditDownloadableAttribute**

```plaintext
context EditDownloadableProductAttribute::effect()
post:  self.downloadable.filename = self.newFilename and
       self.downloadable.expiryDays = self.newExpiryDays and
       self.downloadable.maximumDownloadCount = self.newMaximumDownloadCount
```
CSTL and its application to the osCommerce case study.
Albert Tort

**Example test programs**

```plaintext
testprogram ProductOptionsManagement{
    catalan := new Language(name:='Catalan', code:='CAT');
    english := new Language(name:='English', code:='EN');

    fixturecomponent optionShirtSizeInitialized{
        shirtSize:=new Option;
        englishName:=new StringDT(string:='Shirt size');
        catalanName:=new StringDT(string:='Mida de samarretes');
        new HasOptionName
            (option:=shirtSize, optionName:=englishName, optionLanguage:=english);
        new HasOptionName
            (option:=shirtSize, optionName:=catalanName, optionLanguage:=catalan);
    }

    fixturecomponent valueSmallInitialized{
        small:=new Value;
        englishName:=new StringDT(string:='Small');
        catalanName:=new StringDT(string:='Petit');
        new HasValueName(value:=small, valueName:=englishName, valueLanguage:=english);
        new HasValueName(value:=small, valueName:=catalanName, valueLanguage:=catalan);
    }

    test NewProductOptionWithoutNamesForSomeLanguages{
        // We should specify the product option name in each language
        s:=new StringDT(string:='Size');
        npo:=new NewProductOption;
        new HasNewOptionName(nameOfOption:=s,
            languageOfOption:=english,productOptionNameEvent:=this));
        npo may not occur;
    }

    test NewProductOptionsWithAllNamesSpecified{
        // We test a valid invocation of the event
        englishName:=new StringDT(string:='Size');
        catalanName:=new StringDT(string:='Mida');
        npo:=new NewProductOption;
        new HasNewOptionName(nameOfOption:=englishName,
            languageOfOption:=english,productOptionNameEvent:=this);
        new HasNewOptionName(nameOfOption:=catalanName,
            languageOfOption:=catalan,productOptionNameEvent:=this);
        npo occurs;
    }

    test NewProductOptionThatAlreadyExists{
        load optionShirtSizeInitialized;
    }
}
```
CSTL and its application to the osCommerce case study.
Albert Tort

differentName:=new StringDT(string:='AnyName');
npo:=new HasNewOptionName(nameOfOption:=catalanName,
languageOfOption:=catalan,productOptionNameEvent:=this);
npo may not occur;
npo2:=new HasNewOptionName(nameOfOption:=differentName,
languageOfOption:=catalan,productOptionNameEvent:=this);
npo2 may not occur;
npo3:=new HasNewOptionName(nameOfOption:=catalanName,
languageOfOption:=catalan,productOptionNameEvent:=this);
npo3 may not occur;
}

test EditProductOptionWithoutNamesForSomeLanguages{
  load optionShirtSizeInitialized;
s:=new StringDT(string:='Size');
npo:=new HasNewOptionName(option:=shirtSize);
npo may not occur;
}

test EditProductOptionsWithAllNamesSpecified{
  load optionShirtSizeInitialized;
  englishName:=new StringDT(string:='Size');
  catalanName:=new StringDT(string:='Mida');
  epo:=new HasOptionName(option:=sleeveType,
  optionName:=englishName, optionLanguage:=english); assert consistency;
differentName:=new StringDT(string:='AnyName');
  epo2:=new HasNewOptionName(option:=sleeveType,
  languageOfOption:=catalan,productOptionNameEvent:=this);
  epo2 may not occur;
  epo3:=new HasNewOptionName(option:=sleeveType,
  languageOfOption:=catalan,productOptionNameEvent:=this);
  epo3 may not occur;
  }
}
CSTL and its application to the osCommerce case study.
Albert Tort

testprogram DeleteProductOptions{
  shoesSize:= new Option;
  shirtSize:= new Option;
  small:= new Value;

test deleteOptionWithoutValues{
  new DeleteProductOption(option:=shirtSize);
}

test deleteOptionThatIsPartOfAProductAttribute{
  barcelonaTShirt:= new Product;
  smallShirt:= new Attribute(option:=shirtSize, value:=small);
  new ProductAttribute(product:=barcelonaTShirt, attribute:=smallShirt);
  new DeleteProductOption(option:=shirtSize) may not occur;
}

test deleteOptionWithAssociatedValuesNotUsedInOtherOptions{
  new Attribute(option:=shirtSize, value:=small);
  new DeleteProductOption(option:=shirtSize) may not occur;
}

test deleteOptionWithAssociatedValuesUsedInOtherOptions{
  new Attribute(option:=shirtSize, value:=small);
  new Attribute(option:=shoesSize, value:=small);
  new DeleteProductOption(option:=shirtSize) occurs;
  assert true Value.allInstances->includes(small);
}

testprogram ProductOptionsValuesManagement{
  catalan := new Language(name:='Catalan', code:='CAT');
  english := new Language(name:='English', code:='EN');

  shirtSize:= new Option;
  englishName:= new StringDT(string:='Shirt size');
  catalanName:= new StringDT(string:='Mida de samarretes');
  new HasOptionName(option:=shirtSize, optionName:=englishName, optionLanguage:=english);
  new HasOptionName(option:=shirtSize, optionName:=catalanName, optionLanguage:=catalan);

  fixturecomponent valueSmallInitialized{
    smallInEnglish:= new StringDT(string:='Small');
    smallInCatalan:= new StringDT(string:='Petit');
    small:= new Value;
    new HasValueName(value:=small, valueName:=smallInEnglish, valueLanguage:=english);
    new HasValueName(value:=small, valueName:=smallInCatalan, valueLanguage:=catalan);
  }

test NewProductOptionValueWithoutNamesForSomeLanguages{
  //We should specify the product option name in each language and an option
  smallInEnglish:= new StringDT(string:='Small');
  npov:= new NewProductOptionValue;
  new HasNewValueName(nameOfValue:=smallInEnglish, languageOfValue:=english, productValueNameEvent:=this));
  npov may not occur;
}

test NewProductOptionValueWithAllNamesSpecified{
  //We test a valid invocation of the event
  smallInEnglish:= new StringDT(string:='Small');
  npov:= new NewProductOptionValue(option:=shirtSize);
  new HasNewValueName(nameOfValue:=smallInEnglish, languageOfValue:=english, productValueNameEvent:=this));
  npov occurs;
}
CSTL and its application to the osCommerce case study.
Albert Tort

test NewProductOptionValueThatAlreadyExists{
  //IB state with a product option value
  load valueSmallInitialized;
  smallInEnglish:=new StringDT(string:='Small');
  smallInCatalan:=new StringDT(string:='Petit');
  //The creation of a product option value with the same name in at least one
  //language should not occur
  differentName:=new StringDT(string:='AnyName');
  npov1:=new NewProductOptionValue(option:=shirtSize,
        nameOfValue:=smallInCatalan,
        languageOfValue:=catalan,productValueNameEvent:=this);
  new HasNewValueName(nameOfValue:=differentName,
        languageOfValue:='english',productValueNameEvent:=this));
  npov1 may not occur;
  npov2:=new NewProductOptionValue(option:=shirtSize,
        nameOfValue:=smallInCatalan,
        languageOfValue:=catalan,productValueNameEvent:=this);
  new HasNewValueName(nameOfValue:=differentName,
        languageOfValue:='english',productValueNameEvent:=this));
  npov2 may not occur;
  npov3:=new NewProductOptionValue(option:=shirtSize,
        nameOfValue:=smallInCatalan,
        languageOfValue:=catalan,productValueNameEvent:=this);
  new HasNewValueName(nameOfValue:=smallInEnglish,
        languageOfValue:='english',productValueNameEvent:=this));
  npov3 may not occur;
}

test EditProductOptionValueWithoutNamesForSomeLanguages{
  //We should specify the product Value name in each language
  load valueSmallInitialized;
  s:=new StringDT(string:='Small');
  epov:=new EditProductOptionValue(option:=shirtSize, value:=small,
        nameOfValue:=s,
        languageOfValue:='english',productValueNameEvent:=this));
  epov may not occur;
}

test EditProductValuesWithAllNamesSpecified{
  load valueSmallInitialized;
  smallInEnglish:=new StringDT(string:='Small');
  smallInCatalan:=new StringDT(string:='Petit');
  //We test a valid invocation of the event
  epov:=new EditProductOptionValue(option:=shirtSize,value:=small,
        nameOfValue:=smallInCatalan,
        languageOfValue:=catalan,productValueNameEvent:=this);
  new HasNewValueName(nameOfValue:=smallInCatalan,
        languageOfValue:='english',productValueNameEvent:=this));
  epov occurs;
}

test UnapplicableProductValueEdition{
  load valueSmallInitialized;
  //We add to the IB another Value
  large:=new Value;
  englishName:=new StringDT(string:='Large');
  catalanName:=new StringDT(string:='Gran');
  new HasValueName(value:=large,
        valueName:=englishName, valueLanguage:=english);
  new HasValueName(value:=large,
        valueName:=catalanName, valueLanguage:=catalan);
  assert consistency;
  differentName:=new StringDT(string:='AnyName');
  epov:=new EditProductOptionValue(value:=small,option:=shirtSize,
        nameOfValue:=differenceName,
        languageOfValue:=catalan,productValueNameEvent:=this));
  epov occurs;
  epov:=new EditProductOptionValue(value:=small,option:=shirtSize,
        nameOfValue:=differenceName,
        languageOfValue:=catalan,productValueNameEvent:=this));
  epov may not occur;
  epov:=new EditProductOptionValue(value:=small,option:=shirtSize,
        nameOfValue:=englishName,
        languageOfValue:=english,productValueNameEvent:=this));
  epov may not occur;
  epov:=new EditProductOptionValue(value:=small,option:=shirtSize,
        nameOfValue:=catalanName,
        languageOfValue:=catalan,productValueNameEvent:=this));
  epov may not occur;
  epov:=new EditProductOptionValue(value:=small,option:=shirtSize,
        nameOfValue:=catalanName,
        languageOfValue:=catalan,productValueNameEvent:=this));
  epov may not occur;
}
CSTL and its application to the osCommerce case study.
Albert Tort

```plaintext
new HasNewValueName(nameOfValue:=englishName,
languageOfValue:=english,productNameEvent:=this));
epov may not occur;
}
}
testprogram DeleteProductOptionsValues{
  shoesSize:=new Option;
  shirtSize:=new Option;
  small:=new Value;

  fixturecomponent barcelonaTShirtInitialized{
    barcelonaTShirt:=new Product;
    smallShirt:=new Attribute(option:=shirtSize,value:=small);
    barcelonaSmallTShirt:=new ProductAttribute
    (product:=barcelonaTShirt,attribute:=smallShirt);
  }

test deleteValueNotUsed{
  new DeleteProductOptionValue(value:=small) occurs;
}
test deleteValueOfTwoOptions{
  smallShirt:=shoesSize,shirtSize;
  new DeleteProductOptionValue(value:=small) occurs;
}
test deleteValueThatIsPartOfAProductAttribute{
  load barcelonaTShirtInitialized;
  new DeleteProductOptionValue(value:=small) may not occur;
}
test deleteValueThatIsPartOfAnOrder{
  load barcelonaTShirtInitialized;
  //We create an order
  o:= new Order;
  ol:=new OrderLine(product:=barcelonaTShirt,order:=o);
  euro:=new Currency;
  o.currency:=euro;
  dos:=new OrderStatus;
  osc :=new OrderStatusChange(order:=o,orderStatus:=dos);
  sm:= new FlatRate(status:=#enabled);
  pm:= new Nochex(status:=#enabled);
  o.shippingMethod:=sm;
  o.paymentMethod:=pm;
  usa:=new Country;
  a:= new Address(country:=usa);
  c :=new Customer(address:=a,primary:=a);
  o.customer:=c;
  ola:=new OrderLineAttribute(attribute:=smallShirt, orderLine:=ol);
  //We cannot delete a value wich is part of an attribute of an order...
  new DeleteProductOptionValue(value:=small) may not occur;
  delete barcelonaSmallTShirt;
  assert consistency;
  //...although the product attribute is not offered
  new DeleteProductOptionValue(value:=small) may not occur;
}
}

testprogram ProductOptionsManagement{
  edition:=new Option; version:=new Option;
  special:=new Value;
  specialWithDirectorComments:=new Value;
  catalan:=new Value;
  vickyCristinaBarcelonaDVD:=new Product(netPrice:=20);
  specialEdition:=new Attribute(option:=edition,value:=special);
  specialWithCommentsEdition:=new Attribute
  (option:=edition,value:=specialWithDirectorComments);
  catalanVersion:=new Attribute(option:=version,value:=catalan);

  fixturecomponent vickyCristinaBarcelonaSpecialDVDEditionInitialize{
    vcbSpecialDVDEdition:=new ProductAttribute
    (product:=vickyCristinaBarcelonaDVD, attribute:=specialEdition);
    vcbSpecialDVDEdition.increment:=3;
    vcbSpecialDVDEdition.sign:=$plus;
  }
```

91
CSTL and its application to the osCommerce case study.
Albert Tort

```plaintext
test NewProductAttributeWithValidOptionValuePair{
    new NewProductAttribute
    (product:=vickyCristinaBarcelonaDVD, option:=edition, value:=special, increment:=3, sign:=#plus) occurs;
}

test NewProductAttributeWithInvalidOptionValuePair{
    new NewProductAttribute(product:=vickyCristinaBarcelonaDVD, option:=edition, value:=catalan, increment:=3, sign:=#plus) may not occur;
}

test NewProductAttributeThatAlreadyExists{
    load vickyCristinaBarcelonaSpecialDVDEditionInitialize;
    // If a product attribute with the same option and value already exists in the
    // I/B, the event NewProductAttribute should not occur
    new NewProductAttribute(product:=vickyCristinaBarcelonaDVD, option:=edition, value:=special, increment:=5, sign:=#minus) may not occur;
}

test EditProductAttribute{
    load vickyCristinaBarcelonaSpecialDVDEditionInitialize;
    new AttributeChange
    (productAttribute:=vcbSpecialDVDEdition, newValue:=specialWithDirectorComments, newOption:=edition) occurs;
}

test EditIncrementAndSign{
    load vickyCristinaBarcelonaSpecialDVDEditionInitialize;
    new IncrementAndSignAttributeChange(productAttribute:=vcbSpecialDVDEdition, newIncrement:=5, newSign:=#plus) occurs;
}

test InvalidEditProductAttribute{
    load vickyCristinaBarcelonaSpecialDVDEditionInitialize;
    vcbCatalanVersion:=
    new ProductAttribute(product:=vickyCristinaBarcelonaDVD, attribute:=catalanVersion);
    new AttributeChange(productAttribute:=vcbCatalanVersion, newValue:=catalan, newOption:=edition) may not occur;
}

test DeleteProductAttributeNotUsedInAnyOrder{
    load vickyCristinaBarcelonaSpecialDVDEditionInitialize;
    new DeleteProductAttribute(productAttribute:=vcbSpecialDVDEdition) occurs;
    assert true ProductAttribute.allInstances->size()=0;
}

test DeleteProductAttributeUsedInAnOrder{
    load vickyCristinaBarcelonaSpecialDVDEditionInitialize;
    // We create an order
    o:= new Order;
    ol:= new OrderLine(product:=vickyCristinaBarcelonaDVD, order:=o);
    euro:=new Currency;
    o.currency:=euro;
    dos:=new OrderStatus;
    osc := new OrderStatusChange(order:=o, orderStatus:=dos);
    pm:= new Nochex(status:=#enabled);
    o.shippingMethod:=sm;
    o.paymentMethod:=pm;
    spain:=new Country;
    a:= new Address(country:=spain);
    c := new Customer(address:=a, primary:=a);
    o.customer:=c;
    ol:=new OrderLineAttribute(attribute:=specialEdition, orderLine:=ol);
    new DeleteProductAttribute(productAttribute:=vcbSpecialDVDEdition) occurs;
    assert true ProductAttribute.allInstances->includes(vcbSpecialDVDEdition);
    assert equals vcbSpecialDVDEdition.status #disabled;
}
```
Product categories

Structural schema

Products are grouped into categories which are arranged hierarchically.

```
context Category def:
allParents() : Set(Category) = self.parent -> union(self.parent.allParents())
```

**[DR1]** *Category::added* is the *DateTime* of category creation.

```
context Category::added():DateTime
body : Now()
```

**[DR2]** *Category::subcategories* is the number of subcategories owned by the category.

```
context Category::subcategories(): Natural
body : self.child -> size()
```

**[DR3]** *Category::products* is the number of products owned by the category.

```
context Category::products(): Natural
body : Category.allInstances() -> select(c | c.allParents() -> includes(self)) -> union(Set{self}).product -> size()
```

**[IC1]** In each language, each category has a unique name.

```
context Language::categoryNameIsUnique(): Boolean
body : self.hasCategoryName -> isUnique(name)
```

**[IC2]** There are no cycles in category hierarchy.

```
context Category::isAHierarchy(): Boolean
body : not self.allParents() -> includes(self)
```

Use cases

**Add a product category**

Primary Actor: Store administrator
Precondition: None.
CSTL and its application to the osCommerce case study.
Albert Tort

Trigger: The store administrator wants to add a category.

Main Success Scenario:

1. The store administrator provides the details of the new product category, including its parent category, if any:
   \[\Rightarrow \text{NewCategory}\]
2. The system validates that the data is correct.
3. The system saves the new category.

Edit a product category

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to edit a category.

Main Success Scenario:

1. The store administrator selects the category to be edited.
2. The store administrator provides the new details of the selected category:
   \[\Rightarrow \text{EditCategory}\]
3. The system validates that the data is correct.
4. The system saves the changes.

Move a product category

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to change the placement of a category in the category hierarchy.

Main Success Scenario:

1. The store administrator selects the category to be moved.
2. The store administrator indicates the new parent category, if any:
   \[\Rightarrow \text{MoveCategory}\]
3. The system validates that the data is correct.
4. The system saves the new placement.

Delete a product category

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to delete a category.
Main Success Scenario:

1. The store administrator selects the category to be deleted.
2. The system warns the store administrator of the number of subcategories and products linked to the category to be deleted.
3. The store administrator confirms that he wants to delete the category:
   
   [→DeleteCategory]

4. The system deletes the selected category and its subcategories. The products linked to the deleted category or its subcategories are removed from the system if they do not participate in any orders. The system changes the status of the products which participate in orders to out of stock.

Move a product

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to change the category of a product.

Main Success Scenario:

1. The store administrator selects the product to be moved.
2. The store administrator indicates the new category of the selected product, if any:
   
   [→MoveProduct]

3. The system validates that the data is correct.
4. The system saves the new placement.

Link a product

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to link a product to another category.

Main Success Scenario:

1. The store administrator selects the product to be linked.
2. The store administrator indicates the new category of the selected product, if any:
   
   [→LinkProduct]

3. The system links the product.
CSTL and its application to the osCommerce case study.
Albert Tort

Events

NewCategory

- \textbf{context} NewCategory::categoryDoesNotExist(): Boolean
- \textbf{body}: Language.allInstances() -> forAll (l | 
  l.hasCategoryName.categoryName ->
  excludes(self.hasNewName->select(language=l)->any(true).name))

- \textbf{context} NewCategory::effect()
- \textbf{post}:
  - c.oclIsNew() \textbf{and}
  - c.oclIsTypeOf(Category) \textbf{and}
  - c.imagePath = self.imagePath \textbf{and}
  - c.sortOrder = self.sortOrder \textbf{and}
  - c.parent = self.parent \textbf{and}
  - Language.allInstances() ->
    forAll (l | self.hasNewName -> select(language=l).name =
    c.hasCategoryName->select(language=l).categoryName)

EditCategory
CSTL and its application to the osCommerce case study.
Albert Tort

context EditCategory::categoryDoesNotExist():Boolean
body: Language.allInstances -> forAll ( l | l.hasCategoryName.categoryName.string -> excludes(self.hasNewName -> any(language=l).name) or (self.hasNewName->any(language=l).name = self.category.hasCategoryName->any(language=l).categoryName))

context EditCategory::cyclesDoNotAppear():Boolean
self.category.allParents()->union(Set{self.newParent})->excludes(self.category)

context EditCategory::effect()
post : self.category.imagePath = self.imagePath and self.category.sortOrder = self.sortOrder and self.category.parent = self.parent and Language.allInstances() -> forAll(l |

self.hasNewName->select(language=l)->any(true).name= self.category.hasCategoryName->select(language=l).categoryName)
post : self.category.lastModified = Now()

context MoveCategory::cyclesDoNotAppear():Boolean
self.newParent.allParents()->excludes(self.category)

context MoveCategory::effect()
post : self.category.parent = self.newParent

context DeleteCategory::cyclesDoNotAppear():Boolean
self.category.allParents()->excludes(self.category)

context DeleteCategory::effect()
post : self.category.parent = self.newParent
CSTL and its application to the osCommerce case study.
Albert Tort

context DeleteCategory::effect()
  post deleteCategoryAndSubcategories:
    Category.allInstances->excludes(self.category@pre) and
    self.allChilds(category@pre) -> forAll(c | Category.allInstances->excludes(c))
  post deleteProductsOfCategory:
    self.category@pre.product@pre -> forAll(p | if p.orderLine -> notEmpty() then p.status = ProductStatus::outOfStock else p@pre.oclIsKindOf(OclAny) endif)
  post deleteProductsOfChildCategory:
    self.category@pre.child@pre.product@pre -> forAll(p | if p.orderLine -> notEmpty() then p.status = ProductStatus::outOfStock else p.oclIsKindOf(OclAny) endif)

MoveProduct

context MoveProduct::oldCategoryIsValid(): Boolean
  body: product.category->includes(self.oldCategory)

context MoveProduct::effect()
  post: self.product.category -> includes(self.newCategory) and
    self.product.category -> excludes(self.oldCategory)

LinkProduct

context MoveProduct::effect()
  post: self.product.category -> includes(self.newCategory) and
    self.product.category -> excludes(self.oldCategory)
CSTL and its application to the osCommerce case study.
Albert Tort

context LinkProduct::effect()
  post: self.product.category -> includes(self.newCategory)

Example test programs

testprogram ProductCategoriesManagement{
  // Test cases are based on a multilingual online shop with two languages
  italian := new Language(name:='Italian', code:='IT');
  english := new Language(name:='English', code:='EN');

  fixturecomponent woodenToysCategoryInitialized{
    woodenToysInEnglish:=new StringDT(string:='Wooden toys');
    woodenToysInItalian:=new StringDT(string:='Giocattoli di legno');
    woodenToys:=new Category;
    new HasCategoryName(category:=woodenToys, categoryName:=woodenToysInEnglish, language:=english);
    new HasCategoryName(category:=woodenToys, categoryName:=woodenToysInItalian, language:=italian);
  }

  fixturecomponent gamesCategoryInitialized{
    gamesInEnglish:=new StringDT(string:='Games');
    gamesInItalian:=new StringDT(string:='Giocchi di società');
    games:=new Category;
    new HasCategoryName(category:=games, categoryName:=gamesInEnglish, language:=english);
    new HasCategoryName(category:=games, categoryName:=gamesInItalian, language:=italian);
  }

  test NewCategory{
    // We should specify the product option name in each language and an option
    gamesInEnglish:=new StringDT(string:='Games');
    gamesInItalian:=new StringDT(string:='Giocchi di società');
    nc:=new NewCategory;
    new HasNewName(name:=gamesInEnglish, languageOfCategory:=english, categoryNameEvent:=this);
    new HasNewName(name:=gamesInItalian, languageOfCategory:=italian, categoryNameEvent:=this));
    nc occurs;
  }

  test NewSubcategory{
    load woodenToysCategoryInitialized;
    // We should specify the product option name in each language and an option
    trainsInEnglish:=new StringDT(string:='Trains');
    trainsInItalian:=new StringDT(string:='Trenini');
    nc:=new NewCategory(parent:=woodenToys);
    new HasNewName(name:=trainsInEnglish, languageOfCategory:=english, categoryNameEvent:=this);
    new HasNewName(name:=trainsInItalian, languageOfCategory:=italian, categoryNameEvent:=this));
    nc occurs;
  }

  test EditCategory{
    load woodenToysCategoryInitialized;
    trainsInEnglish:=new StringDT(string:='Trains');
    trainsInItalian:=new StringDT(string:='Trenini');
    nc:=new NewCategory(parent:=woodenToys);
    new HasNewName(name:=trainsInEnglish, languageOfCategory:=english, categoryNameEvent:=this);
    new HasNewName(name:=trainsInItalian, languageOfCategory:=italian, categoryNameEvent:=this));
    nc occurs;
    ec:=new EditCategory(category:=woodenToys);
    new HasNewName(name:=trainsInEnglish, languageOfCategory:=english, categoryNameEvent:=this);
    new HasNewName(name:=woodenToysInItalian, languageOfCategory:=italian, categoryNameEvent:=this));
    ec may not occur;
  }
}
CSTL and its application to the osCommerce case study.
Albert Tort

```java
test EditCategoryCausingACycle{
    load woodenToysCategoryInitialized;
    woodenToysInEnglish := new StringDT(string:='Wooden toys');
    woodenToysInItalian := new StringDT(string:='Giocattoli di legno');
    ed := new EditCategory(category:=woodenToys, newParent:=woodenToys);
    new HasNewName(name:=woodenToysInEnglish, languageOfCategory:=english,
        categoryNameEvent:=this);
    new HasNewName(name:=woodenToysInItalian, languageOfCategory:=italian,
        categoryNameEvent:=this));
    ec may not occur;
}
test MoveCategory{
    load woodenToysCategoryInitialized;
    load gamesCategoryInitialized;
    new MoveCategory(category:=games, newParent:=woodenToys) occurs;
    assert equals games.parent woodenToys;
}
test MoveCategoryCausingCycles{
    load woodenToysCategoryInitialized;
    load gamesCategoryInitialized;
    games.parent := woodenToys;
    trainsInEnglish := new StringDT(string:='Trains');
    trainsInItalian := new StringDT(string:='Trenini');
    nc := new NewCategory(parent:=games);
    new HasNewName(name:=trainsInEnglish, languageOfCategory:=english,
        categoryNameEvent:=this);
    new HasNewName(name:=trainsInItalian, languageOfCategory:=italian,
        categoryNameEvent:=this));
    nc occurs;
    trains := HasCategoryName.allInstances
        ->any(categoryName=trainsInEnglish).category;
    new MoveCategory(category:=woodenToys, newParent:=trains) may not occur;
}
test DeleteCategoryWithoutSubcategories{
    load woodenToysCategoryInitialized;
    new DeleteCategory(category:=woodenToys) occurs;
}
test DeleteCategoryWithSubcategories{
    load woodenToysCategoryInitialized;
    load gamesCategoryInitialized;
    new MoveCategory(category:=games, newParent:=woodenToys) occurs;
    new DeleteCategory(category:=woodenToys) occurs;
    assert true Category.allInstances->excludes(woodenToys);
    assert true Category.allInstances->excludes(games);
}
}
testprogram ProductMovementsInCategories{
    p := new Product;
    c1 := new Category;
    c2 := new Category;
    c3 := new Category;
    test MoveBetweenCategories{
        p.category := c1;
        new MoveProduct(product:=p, oldCategory:=c1, newCategory:=c2) occurs;
        assert equals p.category Set{c2};
    }
    test InvalidMoveBetweenCategories{
        new MoveProduct(product:=p, oldCategory:=c1, newCategory:=c2) may not occur;
    }
    test LinkProduct{
        /*Link a product makes possible to assign a product
            to more than one categories
        LinkProduct add categories of a product
            preserving the already assigned categories*/
        p.category := c1;
    }
```
CSTL and its application to the osCommerce case study.
Albert Tort

```plaintext
new LinkProduct(product:=p, newCategory:=c2) occurs;
assert equals p.category Set{c2,c1};
}

test SubcategoriesAndProductsDerivedInformation{
  // We add two new products to the IB
  p2:=new Product;
p3:=new Product;

  // We establish the categories hierarchy
  c1.child:=c2,c3;
  // We organize products
  c1.product:=p;
c2.product:=p2,p3;
  // We materialize the derived attributes
  c1._subcategories:=2;
c2._subcategories:=0;
c3._subcategories:=0;
c1._products:=3;
c2._products:=2;
c3._products:=0;
assert consistency;
}
}
```

Specials

**Structural schema**

*osCommerce* allows offering specials. That is, lower prices for a set of products can be offered during a specific time period.

```
Product

Special

<table>
<thead>
<tr>
<th>specialPrice</th>
<th>Money</th>
</tr>
</thead>
<tbody>
<tr>
<td>expiryDate</td>
<td>DateTime [0..1]</td>
</tr>
<tr>
<td>specialAdded</td>
<td>DateTime [&quot;constant&quot;]</td>
</tr>
<tr>
<td>specialLastModified</td>
<td>DateTime [0..1]</td>
</tr>
<tr>
<td>specialStatus</td>
<td>Status</td>
</tr>
<tr>
<td>dateStatusChanged</td>
<td>DateTime [0..1]</td>
</tr>
</tbody>
</table>

<<enumeration>>
- enabled
- disabled
<<dataType>>
- Money
text Special::added is the DateTime when the special was created
context Special::added():DateTime
body: Now()
```

Add a special

**Primary Actor:** Store administrator

**Precondition:** None.

**Trigger:** The store administrator wants to add a special.

**Main Success Scenario:**

```
CSTL and its application to the osCommerce case study.
Albert Tort

1. The store administrator selects the product which will be offered in a special price.
2. The store administrator provides the details of the special:
   \[\rightarrow \text{NewSpecial}\]
3. The system validates that the data is correct.
4. The system saves the new special.

**Edit a special**

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to edit a special.

**Main Success Scenario:**

1. The store administrator selects the special to be edited.  
2. The store administrator provides the new details of the selected special:  
   \[\rightarrow \text{EditSpecial}\]
3. The system validates that the data is correct.  
4. The system saves the changes.

**Delete a special**

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to delete a special.

**Main Success Scenario:**

1. The store administrator selects the special to be deleted.  
2. The system asks for the confirmation of the store administrator.  
3. The store administrator confirms that he wants to delete the special:  
   \[\rightarrow \text{DeleteSpecial}\]
4. The system deletes the special.

**NewSpecial**

```
DomainEvent

NewSpecial
  specialPrice : Money
  expiryDate : DateTime [0..1]
  status : SpecialStatus
  effect()
  1 Product
```
CSTL and its application to the osCommerce case study.
Albert Tort

context NewSpecial::effect()
post:
self.product.oclIsTypeOf(Special) and
self.product.oclAsTypeOf(Special).specialPrice=self.specialPrice and
self.product.oclAsTypeOf(Special).expiryDate=self.expiryDate and
self.product.oclAsTypeOf(Special).status=self.status

context EditSpecial::effect()
post:
self.special.specialPrice = self.newSpecialPrice and
self.special.expiryDate = self.newExpiryDate and
self.special.status = self.newStatus
post:
self.special.lastModified = Now()
post:
self.special@pre.status <> self.newStatus implies
self.special.dateStatusChanged = Now()

context DeleteSpecial::effect()
post:
Special.allInstances() ->excludes(special@pre) and
(Product.allInstances() - Product.allInstances()@pre) -> one(p:Product |
p.status = special@pre.status@pre and
p.available = special@pre.available@pre and
p.netPrice = special@pre.netPrice@pre and

EditSpecial

DeleteSpecial
CSTL and its application to the osCommerce case study.
Albert Tort

```csharp
p.quantityOnHand = special@pre.quantityOnHand@pre and
p.model = special@pre.model@pre and
p.imagePath = special@pre.imagePath@pre and
p.weight = special@pre.weight@pre and
p.category = special@pre.category@pre and
p.manufacturer = special@pre.manufacturer@pre and
p.taxClass = special@pre.taxClass@pre and
p.lastModified=Now() and
Language.allInstances ->
   forAll (l|
      special@pre.productInLanguage->select(language=l).name =
      p.productInLanguage->select(language=l).name))
```

Example test program

```csharp
testprogram SpecialsManagement{
   skypePhone:=new Product(netPrice:=90); 
   test AddEditAndDeleteSpecials{
      ns:=new NewSpecial(product:=skypePhone,specialPrice:=60,status:=#disabled)
      occurs;
      assert true ns.product.specialNetPrice().isUndefined();
      new EditSpecial(special:=ns.product,newSpecialPrice:=60,newStatus:=#enabled)
      occurs;
      assert equals ns.product.specialNetPrice() 60;
      new EditSpecial(special:=ns.product,newSpecialPrice:=55,newStatus:=#enabled)
      occurs;
      assert equals ns.product.specialNetPrice() 55;
      specialProduct:=ns.product;
      new DeleteSpecial(special:=specialProduct) occurs;
      assert true ns.product.specialNetPrice().isUndefined();
   }
}
```

Manufacturers

Structural schema

In osCommerce, the products in the store are manufactured by manufacturers.

![Diagram of Manufacturer and Language structural schema]

**[DR1]** Manufacturer::added is the Date\(\text{Time}\) when the Manufacturer was created.

- context Manufacturer::added():DateTime
- body : Now()}
CSTL and its application to the osCommerce case study.
Albert Tort

[IC1] A manufacturer is identified by its name

context Manufacturer::nameIsUnique(): Boolean
body: Manufacturer.allInstances() -> isUnique(name)

[IC2] Each manufacturer must have a URL in each language

context Manufacturer::aURLInEachLanguage(): Boolean
body: self.language ->size() = Language.allInstances() -> size()

Use cases

Add a manufacturer

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to add a manufacturer.

Main Success Scenario:

1. The store administrator provides the details of the new manufacturer:
   
   [⇒NewManufacturer]

2. The system validates that the data is correct.
3. The system saves the new manufacturer.

Edit a manufacturer

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to edit a manufacturer.

Main Success Scenario:

1. The store administrator selects the manufacturer to be edited.
2. The store administrator provides the new details of the selected manufacturer:
   
   [⇒EditManufacturer]

3. The system validates that the data is correct.
4. The system saves the changes.

Delete a manufacturer

Primary Actor: Store administrator
Precondition: None.
CSTL and its application to the osCommerce case study.
Albert Tort

**Trigger:** The store administrator wants to delete a manufacturer.

**Main Success Scenario:**

1. The store administrator selects the manufacturer to delete.
2. The system warns the store administrator of the number of products linked to the manufacturer to be deleted.
3. The store administrator confirms that he wants to delete the manufacturer:

   ![DeleteManufacturer](image)

4. The system deletes the manufacturer and, if requested, changes the status of the products manufactured by it to out of stock.

**NewManufacturer**

```
<<initial>>
context NewManufacturer::manufacturerDoesNotExist(): Boolean
body:
   not Manufacturer.allInstances() -> exists (m | m.name=self.name)

context NewManufacturer::effect()
post:
   m.oclIsNew() and
   m.oclIsTypeOf(Manufacturer) and
   m.name = self.name and
   m.imagePath = self.imagePath and
   Language.allInstances() -> forAll (l | self.hasURL -> select(language=l).url =
   m.manufacturerInLanguage->select(language=l).url)
```
CSTL and its application to the osCommerce case study.
Albert Tort

**EditManufacturer**

```asciidoc
<<interface>>
context EditManufacturer::manufacturerDoesNotExist(): Boolean
body:
(Manufacturer.allInstances() - Set(self.manufacturer).name -> excludes(self.name))
```

```asciidoc
context EditManufacturer::effect()
post:
self.manufacturer.name = self.name and
self.manufacturer.imagePath = self.imagePath and
Language.allInstances() ->
forAll(l ->
  self.hasURL -> select(language = l).url =
  self.manufacturer.manufacturerInLanguage ->
  select(language = l).url)
post:
self.manufacturer.lastModified = Now()
```

**DeleteManufacturer**

```asciidoc
<<interface>>
context DeleteManufacturer::effect()
post deleteManufacturer:
not self.manufacturer@pre.oclIsKindOf(OclAny)
post changeProductsToOutOfStock:
deleteProds implies
  manufacturer@pre.product@pre ->
  forAll(status = ProductStatus::outOfStock)
```
CSTL and its application to the osCommerce case study.
Albert Tort

Example test program

```java
//Test cases are based on a multilingual online shop with two languages
spanish := new Language(name:='Spanish', code:='ES');
english := new Language(name:='English', code:='EN');
test NewManufacturerWithoutURLs{
    new NewManufacturer(name:='BooksEditorial') may not occur;
}
test NewManufacturer{
    //We test a valid invocation of the event
    englishURL := new URL(url:='bookseditorial.com/english');
    spanishURL := new URL(url:='bookseditorial.com/spanish');
    nm := new NewManufacturer(name:='bookseditorial');
    new HasURL(url:=englishURL,languageOfURL:=english,manufacturerURLEvent:=this);
    nm occurs;
    createdManufacturer := Manufacturer.allInstances->any(name='bookseditorial');
    assert
        equals createdManufacturer.manufacturerInLanguage->any(language=english).url.url
            'bookseditorial.com/english';
    assert
        equals createdManufacturer.manufacturerInLanguage
            ->any(language=spanish).url.url
            'bookseditorial.com/spanish';
    //We cannot create the same manufacturer again
    nm2 := new NewManufacturer(name:='bookseditorial');
    new HasURL(url:=englishURL,languageOfURL:=english,manufacturerURLEvent:=this);
    new HasURL(url:=spanishURL,languageOfURL:=spanish,manufacturerURLEvent:=this);
    nm2 may not occur;
}
test EditManufacturer{
    //IB state with already existing manufacturers
    englishURL1 := new URL(url:='bookseditorial.com/english');
    spanishURL1 := new URL(url:='bookseditorial.com/spanish');
    bookseditorial := new Manufacturer(name:='bookseditorial');
    miEnglish := new ManufacturerInLanguage(manufacturer:=bookseditorial,language:=english);
    miEnglish.url := englishURL1;
    miSpanish := new ManufacturerInLanguage(manufacturer:=bookseditorial,language:=spanish);
    miSpanish.url := spanishURL1;
    //We create the manufacturer to be modified
    englishURL2 := new URL(url:='www.salamandra.info');
    spanishURL2 := new URL(url:='www.salamandra.info');
    nm := new NewManufacturer(name:='Salamandra');
    new HasURL(url:=englishURL2,languageOfURL:=english,manufacturerURLEvent:=this);
    new HasURL(url:=spanishURL2,languageOfURL:=spanish,manufacturerURLEvent:=this);
    nm occurs;
    salamandra := Manufacturer.allInstances->any(name='Salamandra');
    em := new EditManufacturer(manufacturer:=salamandra, name:='Ediciones Salamandra');
    new HasURL(url:=englishURL2,languageOfURL:=english,manufacturerURLEvent:=this);
    new HasURL(url:=spanishURL2,languageOfURL:=spanish,manufacturerURLEvent:=this);
    em occurs;
    assert
        equals salamandra.name 'Ediciones Salamandra';
    em2 := new EditManufacturer(manufacturer:=salamandra, name:='bookseditorial');
    new HasURL(url:=englishURL2, languageOfURL:=english,manufacturerURLEvent:=this);
    new HasURL(url:=spanishURL2, languageOfURL:=spanish,manufacturerURLEvent:=this);
    em2 may not occur;
}
```
CSTL and its application to the osCommerce case study.
Albert Tort

```java
test DeleteManufacturerWithNoProducts{
  englishURL1:=new URL(url:='bookseditorial.com/english');
  spanishURL1:=new URL(url:='bookseditorial.com/english');
  nm:=new NewManufacturer(name:='bookseditorial');
  new HasURL(url:=englishURL1,languageOfURL:=english, manufacturerURLEvent:=this);
  new HasURL(url:=spanishURL1,languageOfURL:=spanish, manufacturerURLEvent:=this);
  nm occurs;
  bookseditorial:=Manufacturer.allInstances->any(name='bookseditorial');
  new DeleteManufacturer(manufacturer:=bookseditorial, deleteProds:=false) occurs;
  assert true Manufacturer.allInstances->excludes(bookseditorial);
}

abstract test DeleteManufacturerWithProducts(Boolean deleteProds){
  englishURL2:=new URL(url:='www.salamandra.info');
  spanishURL2:=new URL(url:='www.salamandra.info');
  nm:=new NewManufacturer(name:='Salamandra');
  new HasURL(url:=englishURL2,languageOfURL:=english, manufacturerURLEvent:=this);
  new HasURL(url:=spanishURL2,languageOfURL:=spanish, manufacturerURLEvent:=this);
  nm occurs;
  salamandra:=Manufacturer.allInstances->any(name='Salamandra');
  bookNameInEnglish:=new StringDT(string:='The Boy in the Striped Pyjamas');
  bookNameInSpanish:=new StringDT(string:='El niño con el pijama de rayas');
  np:=new NewProduct(manufacturer:=salamandra,netPrice:=30,quantityOnHand:=50);
  new HasNewProductName(nameOfProduct:=bookNameInEnglish, languageOfProduct:=english, productNameEvent:=this);
  new HasNewProductName(nameOfProduct:=bookNameInSpanish, languageOfProduct:=spanish, productNameEvent:=this);
  np occurs;
  book:=Product.allInstances->any(productInLanguage
  ->exists(name='El niño con el pijama de rayas'));
  new DeleteManufacturer(manufacturer:=salamandra, deleteProds:=$deleteProds) occurs;
  assert true Manufacturer.allInstances->excludes(salamandra);
  if $deleteProds
    then assert equals book.status #outOfStock;
  endif
}

test DeleteManufacturerWithProducts($deleteProds:=false);
test DeleteManufacturerWithProducts($deleteProds:=true);
```
CSTL and its application to the osCommerce case study.
Albert Tort

Banners

Structural schema

*osCommerce* allows administrating banners published in the *online* store.

![Diagram of Banner and BannerGroup classes]

- **[DR1]** *Banner::added* is the *DateTime* when the banner was created.
  
  **context** Banner::added():DateTime  
  **body** : Now()

- **[IC1]** A Banner is identified by its title.
  
  **context** Banner::titleIsUnique: Boolean  
  **body** : Banner.allInstances() -> isUnique(title)

- **[IC2]** A Banner Group is identified by its name.
  
  **context** BannerGroup::nameIsUnique: Boolean  
  **body** : BannerGroup.allInstances() -> isUnique(name)

Use Cases

Add a banner

- **Primary Actor:** Store administrator  
- **Precondition:** None.  
- **Trigger:** The store administrator wants to add a new banner.
CSTL and its application to the osCommerce case study.
Albert Tort

Main Success Scenario:

1. The store administrator provides the details of the new banner:
   \[\Rightarrow \text{NewBanner}\]
2. The system validates that the data is correct.
3. The system saves the new banner.

**Edit a banner**

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to edit a banner.

Main Success Scenario:

1. The store administrator selects the banner to be edited.
2. The store administrator provides the new details of the selected banner:
   \[\Rightarrow \text{EditBanner}\]
3. The system validates that the data is correct.
4. The system saves the changes.

**Delete a banner**

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to delete a banner.

Main Success Scenario:

1. The store administrator selects the banner to be deleted.
2. The store administrator confirms that he wants to delete the banner:
   \[\Rightarrow \text{DeleteBanner}\]
3. The system deletes the banner.

**Add a banner group**

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to add a new banner group.

Main Success Scenario:

1. The store administrator provides the details of the new banner group:
   \[\Rightarrow \text{NewBannerGroup}\]
2. The system validates that the data is correct.
3. The system saves the new banner.
**Edit a banner group**

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to edit a banner group.

**Main Success Scenario:**

1. The store administrator selects the banner group to be edited.  
2. The store administrator provides the new details of the selected banner group:  
   
   ![EditBannerGroup]

3. The system validates that the data is correct.  
4. The system saves the changes.

**Delete a banner group**

**Primary Actor:** Store administrator  
**Precondition:** The banner group doesn’t contain any banners.  
**Trigger:** The store administrator wants to delete a banner.

**Main Success Scenario:**

1. The store administrator selects the banner group to be deleted.  
2. The store administrator confirms that he wants to delete the banner group:  
   
   ![DeleteBannerGroup]

3. The system deletes the banner.

**Events**

**NewBanner**

```
NewBanner
  title : String
  url : URL [0..1]
  imagePath : String
  html : HtmlText [0..1]
  expires : Date [0..1]
  scheduled : Date [0..1]

context NewBanner::bannerDoesNotExist(): Boolean
body : not Banner.allInstances() ->exists (b | b.title= self.title)
```

```
CSTL and its application to the osCommerce case study.
Albert Tort

**NewBannerGroup**

```plaintext
context NewBanner::effect()
post :
  b.oclIsNew() and
  b.oclIsTypeOf(Banner) and
  b.title = self.title and
  b.url = self.url and
  b.imagePath = self.imagePath and
  b.html = self.html and
  b.expires = self.expires and
  b.scheduled = self.scheduled and
  b.status = BannerStatus::enabled and
  b.bannerGroup = self.bannerGroup

NewBannerGroup
```

```plaintext
context NewBannerGroup::effect()
post :
  bg.oclIsNew() and
  bg.oclIsTypeOf(BannerGroup) and
  bg.name = self.name
```

**EditBanner**

```plaintext
context EditBanner::bannerDoesNotExist(): Boolean
body :
  Banner.allInstances() ->exists (bg | bg.name = self.name)

context EditBanner::effect()
post :
  self.banner.title = self.newTitle and
  self.banner.newTitle = self.newUrl and
  self.banner.newImagePath = self.newHtml and
  self.banner.newExpires = self.newScheduled and
  self.banner.newStatus = BannerStatus

```
CSTL and its application to the osCommerce case study.
Albert Tort

```plaintext
self.banner.url = self.newUrl and
self.banner.imagePath = self.newImagePath and
self.banner.html = self.newHtml and
self.banner.expires = self.newExpires and
self.banner.scheduled = self.newScheduled and
self.banner.status = self.newStatus and
self.banner.bannerGroup = self.bannerGroup
post:
  self.banner@pre.status <> self.newStatus implies self.banner.statusChanged = Now()
```

### EditBannerGroup

```
<initC>
context EditBannerGroup::bannerGroupDoesNotExist():Boolean
  body: (BannerGroup.allInstances - Set{self.bannerGroup}).name->excludes(self.newName)
context EditBannerGroup::effect()
post: self.bannerGroup.name = self.newName
```

### DeleteBanner

```
context DeleteBanner::effect()
post: not self.banner@pre.oclIsKindOf(OclAny)
```
CSTL and its application to the osCommerce case study.
Albert Tort

**DeleteBannerGroup**

```
context DeleteBannerGroup::BannerGroupsIsEmpty():Boolean
body : self.bannerGroup.banner -> isEmpty()
```

```
context DeleteBannerGroup::effect()
post : not self.bannerGroup@pre.oclIsKindOf(OclAny)
```

**Example test program**

```java
testprogram BannersManagement{

test NewBannerGroup{
    new NewBannerGroup(name:='Advertisements') occurs;
    //We cannot create an already existing banner group
    new NewBannerGroup(name:='Advertisements') may not occur;
}

test EditBannerGroup{
    new NewBannerGroup(name:='Advertisements') occurs;
    bgroup:=BannerGroup.allInstances->any(name='Advertisements');
    new EditBannerGroup(bannerGroup:=bgroup,newName:='TopAdvertisements') occurs;
    assert equals bgroup.name 'TopAdvertisements';

    //We can edit a banner group without changes
    new EditBannerGroup(bannerGroup:=bgroup,newName:='TopAdvertisements') occurs;

    //We cannot create duplicates when editing a banner group
    new EditBannerGroup(bannerGroup:=bgroup,newName:='ChristmasSpecials') occurs;
    new EditBannerGroup(bannerGroup:=bgroup,newName:='ChristmasSpecials')
    may not occur;
}

test BannerGroupRequiredForEachBanner{
    new Banner(title:='ChristmasSpecialOffer', imagePath:='special.jpg');
    assert inconsistency;
}

test NewBanner{
    bg:=new BannerGroup(name:='Advertisements');
    new NewBanner(title:='ChristmasSpecialGift',bannerGroup:=bg) occurs;
    //We cannot create already existing banners
    new NewBanner(title:='ChristmasSpecialGift',bannerGroup:=bg) may not occur;
}

test EditBanner{
    bg:=new BannerGroup(name:='Advertisements');
    bg2:=new BannerGroup(name:='CustomerFidelityCampaign');
    bl:=new Banner(title:='WinTheSpecialPrix', bannerGroup:=bg);
    new EditBanner(banner:=bl,newTitle:='WinACar!', newBannerGroup:=bg2) occurs;
    assert equals bl.title 'WinACar!';
    assert equals bl.bannerGroup bg2;
}
CSTL and its application to the osCommerce case study.
Albert Tort

// We cannot generate duplicate banners when editing
b2:=new Banner(title:='25% off', bannerGroup:=bg2);
new EditBanner(banner:=b2,newTitle:='25% off', newBannerGroup:=bg2) occurs;
new EditBanner(banner:=b1,newTitle:='25% off', newBannerGroup:=bg) may not occur;
}

test deleteBanner{  
bg:=new BannerGroup(name:='Advertisements');  
b1:=new Banner(title:='NewBabiesSection', bannerGroup:=bg);  
new DeleteBanner(banner:=b1) occurs;  
assert true Banner.allInstances->size()==0;
}

test deleteBannerGroup{  
// A banner group with banners cannot be deleted  
bg:=new BannerGroup(name:='Sponsors');  
b1:=new Banner(title:='ParisTourism', bannerGroup:=bg);  
new DeleteBannerGroup(bannerGroup:=bg) may not occur;  
new DeleteBanner(banner:=b1) occurs;  
new DeleteBannerGroup(bannerGroup:=bg) occurs;
}

Newsletters

osCommerce allows store administrators sending emails and product notifications to customers.

---

<table>
<thead>
<tr>
<th>Newsletter</th>
</tr>
</thead>
<tbody>
<tr>
<td>title : String</td>
</tr>
<tr>
<td>content : String</td>
</tr>
<tr>
<td>added : DateTime</td>
</tr>
<tr>
<td>sent : DateTime [0..1]</td>
</tr>
<tr>
<td>status : NewsletterStatus</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ProductNotification</th>
</tr>
</thead>
<tbody>
<tr>
<td>global : Boolean</td>
</tr>
<tr>
<td>explicitRelatedProduct : *</td>
</tr>
<tr>
<td>explicitNotifications : *</td>
</tr>
<tr>
<td>/ relatedProduct</td>
</tr>
<tr>
<td>/ notifications</td>
</tr>
</tbody>
</table>

<<enumeration>>

NewsletterStatus

locked
unlocked

[DR1] ProductNotification::notifications is the set of implied products in the notification.

class ProductNotification::notifications():Set(Product)
body:
if self.global then Product.allInstances()
else self.explicitNotifications
end

[DR2] ProductNotification::added is the DateTime when the newsletter was created.

class Newsletter::added():DateTime
body: Now()
CSTL and its application to the osCommerce case study.
Albert Tort

![IC1] A Newsletter is identified by its title.

context Newsletter::titleIsUnique: Boolean
body : Newsletter.allInstances() -> isUnique(title)

Use Cases

Create a newsletter

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to create a new newsletter.

Main Success Scenario:

1. The store administrator selects the type of the newsletter (newsletter or product notification).
2. The store administrator provides the title and the content of the newsletter:
   
   [⇒ NewNewsletter]  
   [⇒ NewProductNotification]
3. The system validates that the data is correct.
4. The system saves the newsletter.

Edit a newsletter

Primary Actor: Store administrator
Precondition: The newsletter is unlocked.
Trigger: The store administrator wants to edit a newsletter.

Main Success Scenario:

1. The store administrator selects the newsletter to be edited.
2. The store administrator provides the new details of the selected newsletter:
   
   [⇒ EditNewsletter]  
   [⇒ EditProductNotification]
3. The system validates that the data is correct.
4. The system saves the changes.

Delete a newsletter

Primary Actor: Store administrator
Precondition: The newsletter is unlocked.
Trigger: The store administrator wants to delete a newsletter.

Main Success Scenario:

1. The store administrator selects the newsletter to be deleted.
2. The store administrator confirms that he wants to delete the newsletter:
CSTL and its application to the osCommerce case study.
Albert Tort

[→DeleteNewsletter]

3. The system deletes the newsletter.

Lock a newsletter

Primary Actor: Store administrator
Precondition: The newsletter is unlocked.
Trigger: The store administrator wants to indicate to the other administrators that a newsletter is pending to be delivered.

Main Success Scenario:

1. The store administrator selects the newsletter to be locked.
   [→LockNewsletter]
2. The system saves the change.

Unlock a newsletter

Primary Actor: Store administrator
Precondition: The newsletter is locked.
Trigger: The store administrator wants to indicate to the other administrators that a newsletter ceases to be locked.
Main Success Scenario:

1. The store administrator selects the newsletter to be unlocked.
   [→UnlockNewsletter]
2. The system saves the change.

Events

NewNewsletter

```
context NewNewsletter::newsletterDoesNotExist(): Boolean
body : not Newsletter.allInstances() -> exists (n | n.title=self.title)
```

```
context NewNewsletter::effect()
post :
  n.oclsNew() and
  n.oclsTypeOf(Newsletter) and
  n.title = self.title and
```
CSTL and its application to the osCommerce case study.

Albert Tort

```
n.content = self.content and
n.status = NewsletterStatus::unlocked
```

### NewProductNotification

```
context NewProductNotification::ProductNotificationDoesNotExist(): Boolean
body: not Newsletter.allInstances() -> exists (n | n.title = self.title)
```

```
context NewProductNotification::effect()
pot:
n.oclIsNew() and
n.oclIsTypeOf(ProductNotification) and
n.title = self.title and
n.content = self.content and
n.global = self.global and
n.explicitNotifications = self.explicitNotifications and
n.status = self.NewsletterStatus::unlocked
```

### EditNewsletter

```
context EditNewsletter::newsletterIsUnlocked(): Boolean
body: self.newsletter.status = Status::unlocked
```

```
context EditNewsletter::newsletterDoesNotExist(): Boolean
body: (Newsletter.allInstances - Set{self.newsletter}).title->excludes(self.newTitle)
```

```
context EditNewsletter::effect()
pot:
    newsletter.title = self.newTitle and
    newsletter.content = self.newContent
```
CSTL and its application to the osCommerce case study.
Albert Tort

**EditProductNotification**

```
class EditProductNotification
def newGlobal : Boolean
def newExplicitNotifications
```

**DeleteNewsletter**

```
class DeleteNewsletter

def newsletterIsUnlocked() : Boolean

context EditProductNotification::effect()
    post:
        self.productNotification.global = self.newGlobal and
        self.productNotification.explicitNotifications = self.newExplicitNotifications
```

**LockNewsletter**

```
class LockNewsletter
```

```
class Newsletter
```
CSTL and its application to the osCommerce case study.
Albert Tort

«InitIC»
context LockNewsletter::newsletterIsNotLocked():Boolean
body: self.newsletter.status <> Status::locked
context LockNewsletter::effect()
post: self.newsletter.status = NewsletterStatus::locked

UnlockNewsletter

ExistingNewsletter

DomainEvent

UnlockNewsletter

effect()

«InitIC»
context UnlockNewsletter::newsletterIsLocked():Boolean
body: self.newsletter.status <> Status::unlocked
context UnlockNewsletter::effect()
post: self.newsletter.status = NewsletterStatus::unlocked

Example test programs

testprogram NewslettersManagement{

test NewNewsletter{
  new NewNewsletter(title:='NewSection',
                    content:='Our new sports section is now opened !') occurs;

  // We cannot create an already existing newsletter
  new NewNewsletter(title:='NewSection',
                    content:='Our new sports section is now opened !')
     may not occur;

  // ... even if it is a product notification (because a product notification is also a newsletter
  p:=new Product;
  new NewProductNotification(title:='NewSection',
                              content:='New section of products similar to p is now opened',
                              explicitNotifications:=p)
     may not occur;
}

test EditNewsletter{
  new NewNewsletter(title:='NewSection',
                    content:='Our new sports section is now opened !') occurs;
  n1:=Newsletter.allInstances->any(title='NewSection');

  // We cannot lock already locked newsletters
  new LockNewsletter(newsletter:=n1) occurs;
  new LockNewsletter(newsletter:=n1)
     may not occur;

  // We cannot edit locked newsletters
  new EditNewsletter(newsletter:=n1,newTitle:='NewTitle')
     may not occur;
  new UnlockNewsletter(newsletter:=n1)
     may not occur;
}
CSTL and its application to the osCommerce case study.
Albert Tort

// Valid newsletter editions
new EditNewsletter(newsletter:=n1,newTitle:='NewSection') occurs;
new EditNewsletter(newsletter:=n1,newTitle:='NewSectionAnnouncement') occurs;
assert equals n.title 'NewSectionAnnouncement';

// We cannot create duplicates when editing a newsletter
new NewNewsletter(title:='NewSpringFashionSection',
                    content:='Our new spring fashion section is now opened !') occurs;
n2:=Newsletter.allInstances->any(title='NewSpringFashionSection');
new EditNewsletter(newsletter:=n2,newTitle:='NewSectionAnnouncement') may not occur;
}


test DeleteNewsletter{
  new NewNewsletter(title:='NewSection',
                    content:='Our new sports section is now opened !') occurs;
n:=Newsletter.allInstances->any(title='NewSection');
  // A locked newsletter cannot be deleted
  new LockNewsletter(newsletter:=n) occurs;
  new DeleteNewsletter(newsletter:=n) may not occur;
  // Only unlocked newsletter can be deleted
  new UnlockNewsletter(newsletter:=n) occurs;
  new DeleteNewsletter(newsletter:=n) occurs;
  assert true Newsletter.allInstances->excludes(n);
}


testprogram ProductNotifications{
  // In this test program we exercise the specific properties of product notifications
  aucaSenyorEsteveBook := new Product;
  tirantLoBlancBook := new Product;
  new NewProductNotification(title:='Frankfurt 2007',
                               content:='Catalan culture will be the guest of honour at
                               the 2007 Frankfurt Book Fair.',
                               global:=false,
                               explicitNotifications := aucaSenyorEsteveBook) occurs;
  pn1:=ProductNotification.allInstances->any(title='Frankfurt 2007');
  test globalNotificationsDisabled{
    // We test the derived relationship notifications using materialization
    pn1._notifications:=Set{aucaSenyorEsteveBook};
    assert consistency;
  }
  test globalNotificationsEnabled{
    pn1.global:=true;
    // We test the derived relationship notifications using materialization
    pn1._notifications:=Set{aucaSenyorEsteveBook,tirantLoBlancBook};
    assert consistency;
  }
}

122
Customers

Structural schema

osCommerce keeps information about customers and their addresses, one of which is the primary address.

---

[DR1] **Customer::notifications** is the set of subscriptions to product notifications.

**context** Customer::notifications():Set(Product)

**body** :

if self.globalNotifications then Product.allInstances()
else self.explicitNotifications
endif

[DR2] **Customer::added** is the DateTime of the customer creation.

**context** Customer::added():DateTime

**body** : Now()

[IC1] Customers are identified by their email address.

**context** Customer::eMailIsUnique(): Boolean

**body** : Customer.allInstances() -> isUnique(eMailAddress)

[IC2] Addresses have zone if needed.

**context** Country::addressesHaveZoneIfNeeded(): Boolean

**body** : self.zone -> notEmpty() implies self.address -> forAll (a | a.state = a.zone.name and self = a.zone.country)
Use Cases

Create a customer

**Primary Actor:** Customer  
**Precondition:** None.  
**Trigger:** A customer wants to open an account in the store.

**Main Success Scenario:**

1. The customer provides the required customer data:  
   
   \[\rightarrow \text{NewCustomer}\]

2. The system validates the customer data.
3. The system saves the new account.

Change password

**Primary Actor:** Customer  
**Precondition:** The customer is logged in.  
**Trigger:** A customer wants to change his password.

**Main Success Scenario:**

1. The customer provides the old password.
2. The customer provides the new password twice:  
   
   \[\rightarrow \text{PasswordChange}\]

3. The system validates that the data is correct.
4. The system saves the changes.

Change customer details

**Primary Actor:** Customer  
**Precondition:** The customer is logged in.  
**Trigger:** A customer wants to change its customer details.

**Main Success Scenario:**

1. The customer provides the new customer details:  
   
   \[\rightarrow \text{EditCustomerDetails}\]

2. The system validates that the data is correct.
3. The system saves the changes.

Administrate address book

**Primary Actor:** Customer  
**Precondition:** The customer is logged in and the number of addresses is less than the maximum number of address entries permitted.
CSTL and its application to the osCommerce case study.
Albert Tort

Trigger: A customer wants to view or change the address book.

Main Success Scenario:

1. The system displays the current address book entries of the customer.
2. The customer selects an address book entry to be edited:
   \[ \Rightarrow EditCustomerAddress \]
3. The system validates that the data is correct.
4. The system saves the changes and displays the new address book.
   The customer repeats steps 1-4 until he is done.

Extensions:

2a. The customer doesn’t want to change the address book:
   2a1. The use case ends.
2b. The customer wants to add a new address book entry:
   2b1. The customer provides the required data:
      \[ \Rightarrow NewCustomerAddress \]
   2b2. The use case continues at step 3.
2c. The customer wants to delete an address book entry:
   2c1. The customer selects the address book entry:
      \[ \Rightarrow DeleteCustomerAddress \]
   2c2. The use case continues at step 3.
2d. The customer wants to change the default address book entry:
   2d1. The customer selects the new default address book entry:
      \[ \Rightarrow PrimaryCustomerAddressChange \]
   2d2. The use case continues at step 3.

Edit a customer

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to edit a customer.

Main Success Scenario:

1. The store administrator selects the customer to be edited.
2. The store administrator provides the new details of the selected customer:
   \[ \Rightarrow EditCustomer \]
3. The system validates that the data is correct.
4. The system saves the changes.
CSTL and its application to the osCommerce case study.
Albert Tort

Delete a customer

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to delete a customer.

**Main Success Scenario:**

1. The store administrator selects the customer to be deleted.  
2. The system asks for the confirmation of the store administrator.  
3. The store administrator confirms that he wants to delete the customer:  
   
   \[ DeleteCustomer \]  
4. The system deletes the customer and their addresses, reviews, notification subscriptions and shopping carts.

**Extensions:**

3a. The customer has orders:  
   3a1. The system changes the status of the customer to disable.  
       \[ CustomerStatusChange \]  
   3a2. The system deletes customer's addresses, reviews, notification subscriptions and shopping carts.  
   3a3. The use case ends.

Administrate subscriptions

**Primary Actor:** Customer  
**Precondition:** The customer is logged in.  
**Trigger:** A customer wants to view or change their product notification subscriptions.

**Main Success Scenario:**

1. The system displays the details of the current product notification subscriptions of the customer.  
2. The customer adds a new product subscription:  
   \[ NewProductNotificationSubscription \]  
3. The system validates that the data is correct.  
4. The system saves the changes and displays the new product notification subscriptions.  
   The customer repeats steps 1-4 until he is done.

**Extensions:**

2a. The customer doesn’t want to change their product notification subscriptions:  
   2a1. The use case ends.  
2b. The customer wants to be subscribed or unsubscribed to all product notifications:  
   \[ EditGlobalNotifications \]
CSTL and its application to the osCommerce case study.
Albert Tort

2c. The customer wants to delete a product notification subscription:
   2c1. The customer selects the product:

   $\rightarrow$ DeleteProductNotificationSubscription

   2c2. The use case continues at step 3.

Events

NewCustomer

\begin{center}
\textbf{DomainEvent}
\end{center}

\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{NewCustomer} \\
\hline
\textbf{eMailAddress : EMail} \\
\textbf{dateOfBirth : Date}[0..1] \\
\textbf{phone : String} \\
\textbf{fax : String}[0..1] \\
\textbf{primary : Address} \\
\textbf{newsletter : Boolean} \\
\textbf{password : String} \\
\textbf{passwordConfirmation : String} \\
\hline
\end{tabular}
\end{center}

\begin{itemize}
\item \textbf{InitIC}
  \textit{context} NewCustomer::customerDoesNotExist(): Boolean
  
  \textit{body} : \textbf{not Customer.allInstances()} -> exists (c | c.eMailAddress = self.eMailAddress)

\item \textbf{InitIC}
  \textit{context} NewCustomer::passwordCorrect(): Boolean
  
  \textit{body} : password = passwordConfirmation

\item \textbf{InitIC}
  \textit{context} NewCustomer::firstNameRight(): Boolean
  
  \textit{body} : self.primary.firstName.size() >= MinimumValues.firstName

\item \textbf{InitIC}
  \textit{context} NewCustomer::lastNameRight(): Boolean
  
  \textit{body} : self.primary.lastName.size() >= MinimumValues.lastName

\item \textbf{InitIC}
  \textit{context} NewCustomer::dateOfBirthRight(): Boolean
  
  \textit{body} : CustomerDetails.dateOfBirth implies self.dateOfBirth -> notEmpty() and
  self.dateOfBirth.size() >= MinimumValues.dateOfBirth

\item \textbf{InitIC}
  \textit{context} NewCustomer::genderRight(): Boolean
  
  \textit{body} : CustomerDetails.gender implies self.gender->notEmpty()

\item \textbf{InitIC}
  \textit{context} NewCustomer::suburbRight(): Boolean
  
  \textit{body} : CustomerDetails.suburb implies self.suburb->notEmpty()

\item \textbf{InitIC}
  \textit{context} NewCustomer::eMailRight(): Boolean
  
  \textit{body} : self.eMailAddress.size() >= MinimumValues.eMailAddress
\end{itemize}
CSTL and its application to the osCommerce case study.
Albert Tort

context NewCustomer::streetAddressRight(): Boolean
body : self.primary.street.size() >= MinimumValues.streetAddress

context NewCustomer::companyRight(): Boolean
body :
  CustomerDetails.company implies self.primary.company -> notEmpty() and self.primary.company.size() >= MinimumValues.companyName

context NewCustomer::postCodeRight(): Boolean

context NewCustomer::cityRight(): Boolean
body : self.primary.city.size() >= MinimumValues.city

context NewCustomer::stateRight(): Boolean
body :
  CustomerDetails.state implies self.primary.state -> notEmpty() and self.primary.state.size() >= MinimumValues.state

context NewCustomer::telephoneRight(): Boolean
body : self.telephone.size() >= MinimumValues.telephoneNumber

context NewCustomer::passwordRight(): Boolean
body : self.password.size() >= MinimumValues.password

context NewCustomer::effect()
post :
  c.oclIsNew() and c.oclIsTypeOf(Customer) and c.gender = self.primary.gender and c.firstName = self.primary.firstName and c.lastName = self.primary.lastName and c.dateOfBirth = self.dateOfBirth and c.eMailAddress = self.eMailAddress and c.phone = self.phone and c.fax = self.fax and c.newsletter = self.newsletter and c.password = self.password and c.numberOfLogons = 0 and c.address = Set{primary} and c.primary = primary
CSTL and its application to the osCommerce case study.
Albert Tort

PasswordChange

```plaintext
context ChangePassword::passwordRight(): Boolean
body: self.password.size() >= MinimumValues.password

context ChangePassword::OldPasswordIsCorrect(): Boolean
body: customer.password = self.oldPassword

context ChangePassword::effect()
post: self.customer.password = self.newPassword
```

EditCustomerDetails

```plaintext
context EditCustomerDetails::firstNameRight(): Boolean
body: self.newFirstName.size() >= MinimumValues.firstName

context EditCustomerDetails::lastNameRight(): Boolean
body: self.newLastName.size() >= MinimumValues.lastName

context EditCustomerDetails::effect()
```

```plaintext
newGender : Gender [0..1]
newFirstName : String
newLastName : String
newDateOfBirth : Date [0..1]
newEMailAddress : EMail
newPhone : String
newFax : String [0..1]
newNewsletter : Boolean
```

```plaintext
context EditCustomerDetails::firstNameRight(): Boolean
body: self.newFirstName.size() >= MinimumValues.firstName

context EditCustomerDetails::lastNameRight(): Boolean
body: self.newLastName.size() >= MinimumValues.lastName
```
CSTL and its application to the osCommerce case study.
Albert Tort

«InitIC»
context EditCustomerDetails::dateOfBirthRight(): Boolean
  body :
    CustomerDetails.dateOfBirth implies
    self.newDateOfBirth->notEmpty()
    self.newDateOfBirth.size() >= MinimumValues.dateOfBirth

«InitIC»
context EditCustomerDetails::genderRight(): Boolean
  body : CustomerDetails.gender implies self.newGender->notEmpty()

«InitIC»
context EditCustomerDetails::eMailRight(): Boolean
  body : self.newEMailAddress.size() >= MinimumValues.eMailAddress

«InitIC»
context EditCustomerDetails::telephoneRight(): Boolean
  body : self.newTelephone.size() >= MinimumValues.telephoneNumber

context EditCustomerDetails::effect()
  post :
    customer.gender = self.newGender and
    customer.firstName = self.newFirstName and
    customer.lastName = self.newLastName and
    customer.dateOfBirth = self.newDateOfBirth and
    customer.eMailAddress = self.newEMailAddress and
    customer.phone = self.newPhone and
    customer.fax = self.newFax and
    customer.newsletter = self.newNewsletter

EditCustomerAddress

<<dataType>> Address
  
  1

Customer
  1

ExistingAddressEvent DomainEvent ExistingCustomerEvent

EditCustomerAddress

newAddress : Address

effect()

«InitIC»
context EditCustomerAddress::AddressOfCustomer(): Boolean
  body : self.customer.address -> includes(self.address)

«InitIC»
context EditCustomerAddress::firstNameRight(): Boolean
  body : self.newAdress.firstName.size() >= MinimumValues.firstName

«InitIC»
context EditCustomerAddress::lastNameRight(): Boolean
  body : self.newAdress.lastName.size() >= MinimumValues.lastName

«InitIC»
context EditCustomerAddress::genderRight(): Boolean
  body : CustomerDetails.gender implies self.newAdress.gender->notEmpty()
CSTL and its application to the osCommerce case study.

Albert Tort

context EditCustomerAddress::suburbRight(): Boolean
body : CustomerDetails.suburb \(\implies\) self.newAdress.suburb ->notEmpty()

context EditCustomerAddress::streetAddressRight(): Boolean
body : self.newAdress.street.size() >= MinimumValues.streetAddress

context EditCustomerAddress::companyRight(): Boolean
body :
  CustomerDetails.company \(\implies\)
  self.newAdress.company -> notEmpty() and
  self.newAdress.company.size() >= MinimumValues.companyName

context EditCustomerAddress::postCodeRight(): Boolean

context EditCustomerAddress::cityRight(): Boolean
body : self.newAdress.city.size() >= MinimumValues.city

context EditCustomerAddress::stateRight(): Boolean
body :
  CustomerDetails.state \(\implies\)
  self.newAdress.state -> notEmpty() and
  self.newAdress.state.size() >= MinimumValues.state

context EditCustomerAddress::addressesHaveZoneIfNeeded(): Boolean
body :
  self.newAdress.zone -> notEmpty() \(\implies\)
  self.newAdress.state = self.newAdress.zone.name and
  self.newAdress.country = self.newAdress.zone.country

context EditCustomerAddress::effect()
post :
  self.customer.address -> excludes(self.address) and
  self.customer.address -> includes(self.newAddress)
CSTL and its application to the osCommerce case study.
Albert Tort

NewCustomerAddress

```
context NewCustomerAddress::firstNameRight(): Boolean
body : self.primary.firstName.size() >= MinimumValues.firstName
```

```
context NewCustomerAddress::lastNameRight(): Boolean
body : self.primary.lastName.size() >= MinimumValues.lastName
```

```
context NewCustomerAddress::genderRight(): Boolean
body : CustomerDetails.gender implies self.gender->notEmpty()
```

```
context NewCustomerAddress::suburbRight(): Boolean
body : CustomerDetails.suburb implies self.suburb->notEmpty()
```

```
context NewCustomerAddress::streetAddressRight(): Boolean
body : self.primary.street.size() >= MinimumValues.streetAddress
```

```
context NewCustomerAddress::companyRight(): Boolean
body :
  CustomerDetails.company implies
  self.primary.company -> notEmpty() and
  self.primary.company.size() >= MinimumValues.companyName
```

```
context NewCustomerAddress::postCodeRight(): Boolean
```

```
context NewCustomerAddress::cityRight(): Boolean
body : self.primary.city.size() >= MinimumValues.city
```
CSTL and its application to the osCommerce case study.
Albert Tort

context NewCustomerAddress::stateRight(): Boolean
body : CustomerDetails.state implies
  self.primary.state -> notEmpty() and
  self.primary.state.size() >= MinimumValues.state

context NewCustomerAddress::addressesHaveZoneIfNeeded(): Boolean
body : 
  self.country.zone->size()>0
  implies
  (self.state = self.zone.name and
   self.country = self.zone.country)

context NewCustomerAddress::numberOfAddressesRight(): Boolean
body : self.customer.address -> size() < MaximumValues.addressBookEntries

context NewCustomerAddress::effect()
post :
  Address.allInstances() ->exists (a | 
    a.gender = self.gender and
    a.firstName = self.firstName and
    a.lastName = self.lastName and
    a.company = self.company and
    a.street = self.street and
    a.suburb = self.suburb and
    a.postCode = self.postCode and
    a.city = self.city and
    a.state = self.state and
    a.zone = self.zone and
    a.country = self.country and
    self.customer.address -> includes(a))

DeleteCustomerAddress

context DeleteCustomerAddress::AddressOfCustomer(): Boolean
body : 
  self.customer.address -> includes(self.address)

context DeleteCustomerAddress::AtLeastTwoAddresses(): Boolean
body : self.customer.address.size() >= 2
CSTL and its application to the osCommerce case study.
Albert Tort

```<IniIC>
context DeleteCustomerAddress::PrimaryAddressCannotBeDeleted():Boolean
  self.address <> self.customer.primary

context DeleteCustomerAddress::effect()
  post : self.customer.address -> excludes(self.address)
```

**PrimaryCustomerAddressChange**

```<IniIC>
context PrimaryCustomerAddressChange::AddressOfCustomer(): Boolean
  body : self.customer.address -> includes(self.address)

context PrimaryCustomerAddressChange::effect()
  post : self.customer.primary = self.address
```

**EditCustomer**

```<IniIC>
context EditCustomer::firstNameRight(): Boolean
  body : self.newFirstName.size() >= MinimumValues.firstName
  newGender : Gender [0..1]
  newFirstName : String
  newLastName : String
  newDateOfBirth : Date [0..1]
  newEMailAddress : EMail
  newPhone : String
  newFax : String [0..1]
  newNewsletter : Boolean
  newPassword : String
  newGlobalNotifications : Boolean
  effect()
```

```<IniIC>
context EditCustomer::firstNameRight(): Boolean
  body : self.newFirstName.size() >= MinimumValues.firstName
```
CSTL and its application to the osCommerce case study.
Albert Tort

«InitIC»
context EditCustomer::lastNameRight(): Boolean
body : self.newLastName.size() >= MinimumValues.lastName

«InitIC»
context EditCustomer::dateOfBirthRight(): Boolean
body :
  CustomerDetails.dateOfBirth implies
  self.newDateOfBirth->notEmpty() and
  self.newDateOfBirth.size() >= MinimumValues.dateOfBirth

«InitIC»
context EditCustomer::genderRight(): Boolean
body : CustomerDetails.gender implies self.newGender->notEmpty()

«InitIC»
context EditCustomer::eMailRight(): Boolean
body : self.newEMailAddress.size() >= MinimumValues.eMailAddress

«InitIC»
context EditCustomer::telephoneRight(): Boolean
body : self.newTelephone.size() >= MinimumValues.telephoneNumber

class EditCustomer
context EditCustomer::effect() post:
  customer.gender = self.newGender and
customer.firstName = self.newFirstName and
customer.lastName = self.newLastName and
customer.dateOfBirth = self.newDateOfBirth and
customer.eMailAddress = self.newEMailAddress and
customer.phone = self.newPhone and
customer.fax = self.newFax and
customer.newsletter = self.newNewsletter and
customer.password = self.newPassword and
customer.globalNotifications = self.newGlobalNotifications and
post:
  customer.lastModified = Now()

DeleteCustomer

class DeleteCustomer
context DeleteCustomer::effect() post:
  deleteCustomer;
  deleteReviewsAndShoppingCart;
  not customer@pre.oclIsKindOf(OclAny)
  (customer@pre.customerShoppingCart->notEmpty(). and
   customer@pre.review@pre -> forAll (r | r.oclIsKindOf(OclAny)) and
   customer@pre.customerShoppingCart->notEmpty())
CustomerStatusChange

context CustomerStatusChange::effect()
post : self.customer.status = self.newStatus

NewProductNotificationSubscription

context NewProductNotificationSubscription::ProductIsUnsubscribed(): Boolean

context NewProductNotificationSubscription::effect()
post : self.customer.explicitNotifications -> includes(self.newSubscribedProduct)
CSTL and its application to the osCommerce case study.
Albert Tort

**EditGlobalNotifications**

```plaintext
context EditGlobalNotifications::effect()
post : self.customer.globalNotifications = self.newGlobalNotifications
```

**DeleteProductNotificationSubscription**

```plaintext
context DeleteProductNotificationSubscription::effect()
post : customer.explicitNotifications -> excludes(self.deletedSubscribedProduct)
```

**Example test programs**

```plaintext
testprogram NewCustomer{
    textConfigurationValues := new MinimumValues, MaximumValues;
    textConfigurationValues.firstName:=1;
    textConfigurationValues.lastName:=1;
    textConfigurationValues.dateOfBirth:=6;
    textConfigurationValues.eMailAddress:=1;
    textConfigurationValues.streetAddress:=1;
    textConfigurationValues.companyName:=0;
    textConfigurationValues.postCode:=1;
    textConfigurationValues.city:=1;
    textConfigurationValues.state:=1;
    textConfigurationValues.telephoneNumber:=9;
    textConfigurationValues.password:=4;
    textConfigurationValues.addressBookEntries:=2;
    customerDetailsConfiguration := new CustomerDetails;
    customerDetailsConfiguration.gender:=false;
}
CSTL and its application to the osCommerce case study.
Albert Tort

customerDetailsConfiguration.dateOfBirth:=false;
customerDetailsConfiguration.company:=true;
customerDetailsConfiguration.state:=false;
customerDetailsConfiguration.suburb:=false;
d:= new Date(date:='X/XX/XXXX');

abstract test validNewCustomer(String mail, String phone, String company,
String fax, String firstName, String lastName,
String street, String postCode, String city,
String country, Boolean newsletter,
String password, String passwordConfirmation){
e := new EMail(eMail:=$mail);
pc:= new PostalCode(postalCode:=$postCode);
c := new Country(name:=$country);
a := new Address
(firstName:=$firstName, lastName:=$lastName, company:=$company,
street:=$street, postCode:pc, city:=$city, country:c);
new NewCustomer(eMailAddress:=e, dateOfBirth:=d, phone:$phone,
fax:=$fax, primary:=a, newsletter:$newsletter,
password:=$password,
passwordConfirmation:=$passwordConfirmation) occurs;
}

abstract test invalidNewCustomer(String mail, String phone, String company,
String fax, String firstName, String lastName,
String street, String postCode, String city,
String country, Boolean newsletter, String password,
String passwordConfirmation){
e := new EMail(eMail:=$mail);
pc:= new PostalCode(postalCode:=$postCode);
c := new Country(name:=$country);
a := new Address
(firstName:=$firstName, lastName:=$lastName, company:=$company,
street:=$street, postCode:pc, city:=$city, country:c);
new NewCustomer(eMailAddress:=e, dateOfBirth:=d, phone:$phone,
fax:=$fax, primary:=a, newsletter:$newsletter,
password:=$password,
passwordConfirmation:=$passwordConfirmation) may not occur;
}

//We can easily test the NewCustomer event in different valid or invalid contexts

test validNewCustomer
($mail:='atort@lsi.upc.edu', $phone:='XXXXXXXXX', $company:='UPC',
$fax:='XXXXXXXXX', $firstName:='Albert', $lastName:='Tort',
$street:='Jordi Girona,1', $postCode:='08034', $city:='Barcelona',
$country:='Espanya', $newsletter:=true, $password:='password',
$passwordConfirmation:='password');

test validNewCustomer
($mail:='olive@lsi.upc.edu', $phone:='XXXXXXXXX', $company:='UPC',
$fax:='XXXXXXXXX', $firstName:='Antoni', $lastName:='Olive',
$street:='Jordi Girona,1', $postCode:='08034', $city:='Barcelona',
$country:='Espanya', $newsletter:=true, $password:='password',
$passwordConfirmation:='password');

//Incorrect password confirmation

test invalidNewCustomer
($mail:='olive@lsi.upc.edu', $phone:='XXXXXXXXX', $company:='UPC',
$fax:='XX XX XX XX', $firstName:='Antoni', $lastName:='Olive',
$street:='Jordi Girona,1', $postCode:='08034', $city:='Barcelona',
$country:='Espanya', $newsletter:=false, $password:='password',
$passwordConfirmation:='password2');

//Incorrect minimumValues

test invalidNewCustomer
($mail:='atort@lsi.upc.edu', $phone:='XXXXXXXXX', $company:='UPC',
$fax:='XXXXXXXXX', $firstName:='Albert', $lastName:='Tort',
$street:='Jordi Girona,1', $postCode:='08034', $city:='Barcelona',
$country:='Espanya', $newsletter:=true, $password:='password',
$passwordConfirmation:='password');

test invalidNewCustomer($mail:='', $phone:='XXXXXXXXX', $company:='UPC',
$fax:='XXXXXXXXX', $firstName:='Albert', $lastName:='Tort',
$country:='Espanya', $newsletter:=true, $password:='password',
$passwordConfirmation:='password');
CSTL and its application to the osCommerce case study.
Albert Tort

```cstl
$street:='Jordi Girona,1', $postCode:='08034',
$city:='Barcelona', $country:='Espanya', $newsletter:=true,
$password:='pass', $passwordConfirmation:='pass');
test invalidNewCustomer($mail:='olive@lsi.upc.edu', $phone:='XX', $company:='UPC',
$fax:='XXXXXXXXX', $firstName:='Antoni', $lastName:='Olive',
$street:='Jordi Girona,1', $postCode:='08034',
$city:='Barcelona', $country:='Espanya', $newsletter:=false,
$password:='password', $passwordConfirmation:='password');
test invalidNewCustomer($mail:='atort@lsi.upc.edu', $phone:='XXXXXXXXX',
$company:='UPC', $fax:='XXXXXXXXX', $firstName:='Albert',
$lastName:='Tort', $street:='', $postCode:='',
$city:='Barcelona', $country:='Espanya', $newsletter:=true,
$password:='password', $passwordConfirmation:='password');
}

```
CSTL and its application to the osCommerce case study.
Albert Tort

//Edit customer details

test validCustomerDetailsEditions{
    e2 := new EMail(eMail:='john@yyyyy.yyy');
d2:= new Date(date:='YY/YY/YYYY');
    new EditCustomerDetails(customer:=john,
        newFirstName:='Johnatan', newLastName:='JR.',
        newEMailAddress:=e2, newDateOfBirth:=d2,
        newPhone:='YYYYYYYYY', newFax:='YYYYYYYYY') occurs;
}

test invalidCustomerDetailsEditions{
    e2 := new EMail(eMail:='');
d2:= new Date(date:='YY/YY');
    new EditCustomerDetails(customer:=john,
        newFirstName:='', newLastName:='',
        newEMailAddress:=e2, newDateOfBirth:=d2,
        newPhone:='YYYYYY', newFax:='YY') may not occur;
}

//Edit customer

/*Edit customer can only be executed by the store administrator
(who can edit the customer details including its password and the
global notifications option)*/

test validCustomerEdition{
    e2 := new EMail(eMail:='john@yyyyy.yyy');
d2:= new Date(date:='YY/YY/YYYY');
    new EditCustomer(customer:=john,newPassword:='zxcvxcv',
        newGlobalNotifications:=false,
        newFirstName:='Johnatan', newLastName:='JR.',
        newEMailAddress:=e2, newDateOfBirth:=d2,
        newPhone:='YYYYYYYYY', newFax:='YYYYYYYYY') occurs;
}

test invalidCustomerEdition{
    e2 := new EMail(eMail:='');
d2:= new Date(date:='YY/YY');
    new EditCustomer(customer:=john,
        newPassword:='xy', newGlobalNotifications:=false,
        newFirstName:='', newLastName:='', newEMailAddress:=e2,
        newDateOfBirth:=d2, newPhone:='YYYYYY', newFax:='YY') may not occur;
}
}

testprogram CustomerAddressesManagement{

    //Customer initialization
catalonia:=new Zone(name:='Catalonia', code:='CAT', country:spain);
a:= new Address(country:=spain, zone:=catalonia,
c := new Customer(address:=a,primary:=a);

    //Other locations to be used
saxony:=new Zone(name:='Saxony', code:='SAX', country:germany);
    pc:=new PostalCode(postalCode:='XXXXX');

    //Minimum and maximum values
    textConfigurationValues := new MinimumValues, MaximumValues;
textConfigurationValues.firstName:=1;
textConfigurationValues.lastName:=1;
textConfigurationValues.dateOfBirth:=6;
textConfigurationValues.eMailAddress:=1;
textConfigurationValues.streetAddress:=1;
textConfigurationValues.companyName:=0;
textConfigurationValues.postCode:=1;
textConfigurationValues.city:=1;
textConfigurationValues.state:=1;
textConfigurationValues.telephoneNumber:=9;
textConfigurationValues.password:=4;
textConfigurationValues.addressBookEntries:=2;
customerDetailsConfiguration := new CustomerDetails;
customerDetailsConfiguration.gender:=false;
CSTL and its application to the osCommerce case study.
Albert Tort

customerDetailsConfiguration.dateOfBirth:=true;
customerDetailsConfiguration.company:=false;
customerDetailsConfiguration.state:=false;
customerDetailsConfiguration.suburb:=false;

test validAddressCreations{
  pc:=new PostalCode(postalCode:='XXXXX');
  new NewCustomerAddress(customer:=c, firstName:='XXXX', lastName:='XXXXXX',
                          street:='XXXXX', postCode:=pc, city:='XXXXX',
                          country:=spain, zone:=catalonia, state:='Catalonia')
    occurs;
}

test invalidAddressCreations{
  //Zone must be coherent with the state if it is assigned
  new NewCustomerAddress(customer:=c, zone:=catalonia, firstName:='XXXX',
                          lastName:='XXXXXX', street:='XXXXX', postCode:=pc,
                          city:='XXXXX', country:=spain, state:='Catalonia')
    may not occur;
  new NewCustomerAddress(customer:=c, zone:=saxony, country:=spain, firstName:='XXXX',
                          lastName:='XXXXXX', street:='XXXXX', postCode:=pc,
                          city:='XXXXX', state:='Saxony')
    may not occur;
  //Minimum values cannot be violated
  new NewCustomerAddress(customer:=c, zone:=saxony, country:=spain,
                          firstName:='', lastName:='', street:='XXXXX',
                          postCode:=pc, city:='')
    may not occur;
}

test AddressEdition{
  //We add to the customer another address
  new NewCustomerAddress(customer:=c, zone:=saxony, country:='Germany',
                          firstName:='XXXXXXXX', lastName:='XXXXXXXX',
                          street:='XXXXX', postCode:=pc, city:='Dresden',
                          state:='Saxony')
    occurs;
  //Now, the customer has addresses in Spain and in Germany
  assert equals c.address.country->asSet() Set{spain,germany};
  assert true c.address->exists(street='Lluis Companys');
  //We try to change the spanish address
  //In order to edit an address of a customer we should provide the new address
  na:=new Address(country:=spain, zone:=catalonia, state:='Catalonia',
                  city:='Sitges', street:='Passeig Maritim',
                  postCode:=pc, firstName:='XXXX', lastName:='XXXXXX');
  new EditCustomerAddress(customer:=c, address:=a, newAddress:=na)
    occurs;
  assert false c.address->exists(street='Lluis Companys');
  assert true c.address->exists(street='Passeig Maritim');
  //We can change the primary address
  //We put the address from Germany as the primary
  new PrimaryCustomerAddressChange(address:=c.address->any(country=germany),
                                    customer:=c)
    occurs;
  //We cannot put as primary an address which is not an address of the customer
  a2:= new Address(country:=spain, zone:=catalonia, state:='Catalonia',
                  street:='Anselm Clave', city:='Tarragona');
  new PrimaryCustomerAddressChange(address:=a2, customer:=c)
    may not occur;
  //Minimum values cannot be violated when editing an address
  //We try to edit an address with no city and street information
  na2:=new Address(country:=spain, zone:=catalonia, state:='Catalonia',
                  city:='', street:='', postCode:=pc, firstName:='XXXX',
                  lastName:='XXXXXX');
  new EditCustomerAddress(customer:=c, address:=a, newAddress:=na2)
    may not occur;
  //Finally, we delete an address of a customer;
  assert equals c.address->size() 2;
  new DeleteCustomerAddress(address:=c.address->any(country=spain),
                           customer:=c)
    occurs;
  //We cannot delete the primary address
  new DeleteCustomerAddress(address:=c.primary, customer:=c)
    may not occur;
}
CSTL and its application to the osCommerce case study.
Albert Tort

testcontext ProductSubscriptionsManagement{
    //Customer initialization
    catalonia:=new Zone(name:="Catalonia", code:="CAT", country:=spain);
    a:= new Address(country:=spain, zone:=catalonia, state:="Catalonia",
     street:="Lluis Companys", city:="Sitges");
    c := new Customer(address:=a, primary:=a, globalNotifications:=false);

    //Products initialization
    p1:=new Product;
    p2:=new Product;

test ProductNotificationSubscriptions{
    assert equals c.notifications()->size() 0;
    new NewProductNotificationSubscription(customer:=c, newSubscribedProduct:=p1)
    occurs;
    assert equals c.notifications() Set{p1};

    //We cannot subscribe an already subscribed product
    new NewProductNotificationSubscription(customer:=c, newSubscribedProduct:=p1)
    may not occur;

    //We can subscribe more than one product
    new NewProductNotificationSubscription(customer:=c, newSubscribedProduct:=p2)
    occurs;
    assert equals c.notifications() Set{p1,p2};

    //We can delete subscriptions
    new DeleteProductNotificationSubscription(customer:=c,
     deletedSubscribedProduct:=p2) occurs;
    assert equals c.notifications() Set{p1};

    //If global notifications is enabled, explicit notification subscriptions
    //are not taken into account and all products are considered to be subscribed
    new EditGlobalNotifications(customer:=c, newGlobalNotifications:=true) occurs;
    assert equals c.notifications() Set{p1,p2};
}

testprogram DeleteCustomers{
    //Customer initialization
    co:= new Country;
    a:= new Address(country:=co);
    c:= new Customer(address:=a, primary:=a);
    cu:=new Currency(status:=#enabled);

    //Language initialization
    l:= new Language;

    //Products initialization
    p1:=new Product;
    p2:=new Product;

    //MinimumValues
    mv:=new MinimumValues;
    mv.reviewText:=0;

    //The customer wrote reviews
    new NewReview(customer:=c, product:=p1, language:=l, rating:=#fourStars,
     review:="reviewText") occurs;
    new NewReview(customer:=c, product:=p2, language:=l, rating:=#twoStars,
     review:="reviewText2") occurs;

    //The customer has an active shopping cart
    sc := new CustomerShoppingCart(customer:=c);
    item1 := new ShoppingCartItem(product:=p1, quantity:=3, shoppingCart:=sc);

test deleteCustomerWithNoOrders{
    //The customer is deleted and also its active shopping carts and reviews
    new DeleteCustomer(customer:=c) occurs;
    assert equals p1.review->size() 0;
    assert equals p2.review->size() 0;
}
CSTL and its application to the osCommerce case study.
Albert Tort

//The active shopping cart of the customer is also deleted
assert true c.customerShoppingCart->isEmpty();
}

test deleteCustomerWithOrders{
   //Store initialization
   s:= new Store;
   s.defaultLanguage:=l;
   s.defaultCurrency:=cu;
   s.country:=co;
   cos:= new OrderStatus;
   cos1:= new OrderStatusInLanguage(language:=l, orderStatus:=cos);
   cos1.name:='cancelled';
   s.cancelledStatus:=cos;
   dos:= new OrderStatus;
   dosl:= new OrderStatusInLanguage(orderStatus:=dos, language:=l);
   dosl.name:='pending';
   s.defaultStatus:=dos;

   //We create an order of the customer
   stock := new Stock;
   stock.checkStockLevel:=false;
   stock.allowCheckout:=true;
   stock.substractStock:=false;
   pm:= new CashOnDelivery(status: #enabled);
   sm:= new PerItem(status: #enabled, handlingFee:=5, cost:=10);
   new OrderConfirmation(shoppingCart:=sc, currency:=cu, shippingMethod:=sm, paymentMethod:=pm) occurs;

   new DeleteCustomer(customer:=c) occurs;

   //The customer becomes disabled and also its active shopping carts and reviews
   assert equals c.status #disabled;

   //Reviews of customer are also deleted
   assert equals p1.review->size() 0;
   assert equals p2.review->size() 0;

   //The active shopping cart of the customer is also deleted
   assert true c.customerShoppingCart->isEmpty();
}

Reviews

Structural schema

In order to allow users reading evaluations of a product, customers can write reviews.

<table>
<thead>
<tr>
<th>Review</th>
<th>Language</th>
<th>Product</th>
<th>Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>review : String</td>
<td>rating : Rating</td>
<td>added : DateTime</td>
<td>timesRead : Natural</td>
</tr>
</tbody>
</table>

[1] Review::added is the DateTime of the review creation.

context Review::added():DateTime

body : Now()
Use cases

Add a review

**Primary Actor:** Customer

**Precondition:** None.

**Trigger:** A customer wants to write a review of a product.

**Main Success Scenario:**

1. The customer selects a product.
2. The customer provides the content and the rate of the review:
   
   \[ \Rightarrow \text{NewReview} \]
3. The system validates that the data is correct.
4. The system saves the review.

**Extensions:**

2a. The customer is not logged in:
   
   2a1. The customer logs in:
   
   \[ \Rightarrow \text{LogIn} \]
   
   2a2. The use case continues at step 2.

Edit a review

**Primary Actor:** Store administrator

**Precondition:** None.

**Trigger:** The store administrator wants to edit a review.

**Main Success Scenario:**

1. The store administrator selects the review to be edited.
2. The store administrator provides the modified text and the new rating of the selected review:
   
   \[ \Rightarrow \text{EditReview} \]
3. The system validates that the data is correct.
4. The system saves the changes.

Delete a review

**Primary Actor:** Store administrator

**Precondition:** None.

**Trigger:** The store administrator wants to delete a review.

**Main Success Scenario:**

1. The store administrator selects the review to be deleted.
2. The system asks for the confirmation of the store administrator.
CSTL and its application to the osCommerce case study.
Albert Tort

3. The store administrator confirms that he wants to delete the review:
   $\rightarrow \text{DeleteReview}$

4. The system deletes the review.

Events

**NewReview**

```mermaid
Diagram

Language -> NewReview

NewReview
- review : String
- rating : Rating
- effect() : Boolean

context NewReview::reviewRight(): Boolean
body : self.review.size() >= MinimumValues.reviewText

context NewReview::effect()
post : 
  r.oclIsNew() and  
  r.oclIsTypeOf(Review) and  
  r.review = self.review and  
  r.rating = self.rating and  
  r.customer = self.customer and  
  r.product = self.product and  
  r.language = self.language
```

**EditReview**

```mermaid
Diagram

Review -> EditReview

ExistingReviewEvent

DomainEvent

EditReview
- newReview : String
- newRating : Rating
- newLanguage
- newProduct
- newCustomer
```
CSTL and its application to the osCommerce case study.
Albert Tort

context EditReview::effect()
post:
self.review.review = self.newReview and
self.review.rating = self.newRating and
self.review.language = self.newLanguage and
self.review.product = self.newProduct and
self.review.customer = self.newCustomer
post:
self.review.lastModified = Now()

Example test programs

```

testprogram ReviewsManagement{
  english:=new Language(name:='English', code:='EN');
  spanish:=new Language(name:='Spanish', code:='ES');
  usa:=new Country;
  a1:=new Address(country:=usa);
  e1:=new EMail(eMail:='xxxx1@x.com');
  c1:=new Customer(eMailAddress:=e1, address:=a1, primary:=a1);
  a2:=new Address(country:=usa);
  e2:=new EMail(eMail:='xxxx2@x.com');
  c2:=new Customer(eMailAddress:=e2, address:=a2, primary:=a2);
  hotelcomfort:=new Product;
  new MinimumValues(reviewText:=1);
  test newReview{
    new NewReview(customer:=c1, product:=hotelcomfort,
      language:=english, rating:=#fourStars,
      review:='Very easy to find the hotel near Notting Hill gate. Generally very polite and helpful people in the area') occurs;
  }
  test ThreeReviewsOfProduct{
    new NewReview(customer:=c1, product:=hotelcomfort,
      language:=english, rating:=#fourStars,
      review:='Very easy to find the hotel near Notting Hill gate. Generally very polite and helpful people in the area') occurs;
    new NewReview(customer:=c2, product:=hotelcomfort,
      language:=spanish, rating:=#twoStars,
      review:='Muy bien localizado, al lado del mercado de Porto Bello. És un hotel con una distribución extraña al ocupar varios edificios lo que hace que el laberinto de pasillos sea de lo más divertido. El personal es distante.'). occurs;
```
//A customer can review a product more than once
new NewReview(customer:=c1, product:=hotelcomfort, 
    language:=english, rating:=#fourStars, 
    review:='Easy accessible by public transport') occurs;

assert equals hotelcomfort.review->size() 3;
}

test InvalidReviewCreation{
    //Minimum values configuration must be taken into account
    new NewReview(customer:=c1, product:=hotelcomfort, 
        language:=english, rating:=#fourStars, 
        review:='') may not occur;
}

test ReviewEdition{
    //A customer can publish a review
    new NewReview(customer:=c1, product:=hotelcomfort, 
        language:=english, rating:=#fiveStars, 
        review:='I hate this hotel. Call me for more details 12345') occurs;

    //And the store administrator can edit it
    new EditReview(review:=nr.createdReview, newLanguage:=english, 
        newCustomer:=c1, newRating:=#oneStar, 
        newProduct:=hotelcomfort, 
        newReview:='I do not like this hotel') occurs;
}

test DeleteReview{
    //A customer can publish a review
    nr:=new NewReview(customer:=c1, product:=hotelcomfort, 
        language:=english, rating:=#fiveStars, 
        review:='asdfñjasdfasfasdfjñhasdf');

    assert equals hotelcomfort.review->size() 1;

    //And the store administrator can delete it
    r:=nr.createdReview;
    new DeleteReview(review:=r) occurs;

    assert equals hotelcomfort.review->size() 0;
}
CSTL and its application to the osCommerce case study.
Albert Tort

Shopping carts & Orders

Structural schema

Customers can add or remove products from their shopping carts while they are surfing the online store.

**[DR1]** ShoppingCartItem::price is the net price for an item taking into account the selected product attributes.

context ShoppingCartItem::price():Money
body:
let netPriceWithSpecial:Money =
  if self.product.specialNetPrice ->notEmpty() then self.product.specialNetPrice
  else self.product.netPrice
endif
in
if self.attribute -> isEmpty() then netPriceWithSpecial
else
  self.attribute.productAttribute -> select (pa | pa.product = self.product) -> collect
  (if sign = Sign::plus
    then increment
  else increment
  endif) -> sum() + netPriceWithSpecial
endif

**[DR2]** ShoppingCartItem::added is the DateTime when the item was created.

context ShoppingCartItem::added():DateTime
body: Now()

**[IC1]** If a customer shopping cart exists in the context of a session then its customer is the customer of the session.

context CustomerShoppingCart::sameCustomer(): Boolean
body: self.session.customer -> notEmpty() implies self.session.customer = self.customer
CSTL and its application to the osCommerce case study.
Albert Tort

[IC2] The shopping cart item specifies the selected product attributes, which must be a subset of all the product attributes.

context ShoppingCartItem::productHasTheAttributes(): Boolean
body : self.product.attribute -> includesAll(self.attribute)

[IC3] The shopping cart item specifies only one attribute per option.

context ShoppingCartItem::onlyOneAttributePerOption(): Boolean
body : self.attribute -> isUnique(option)

[IC4] Sessions are identified by its sessionID.

context Session::sessionIdIsUnique(): Boolean
body : Session.allInstances() -> isUnique(sessionId)

Orders are the confirmation that a customer wants to buy the contents of his shopping cart.
CSTL and its application to the osCommerce case study.
Albert Tort

```plaintext
res + 
(((appliedTaxRates -> select (tr | tr.priority = p).rate 
  -> sum()) / 100)+1)*basePrice)
```

```plaintext
context ShippingMethod def:
  shippingCosts(totalWeight:Decimal, totalPrice:Money, quantity:PositiveInteger): Money = 0

category FlatRate def:
  shippingCosts(totalWeight:Decimal, totalPrice:Money, quantity:PositiveInteger): Money = self.cost

category PerItem def:
  shippingCosts(totalWeight:Decimal, totalPrice:Money, quantity:PositiveInteger): Money = self.cost*quantity

category TableRate def:
  shippingCosts(totalWeight:Decimal, totalPrice:Money, quantity:PositiveInteger): Money = 
  if self.method = ShippingTableMethod::weight 
  then 
    self.items -> select (i | i.number <= (totalWeight*quantity)) -> sortedBy(number) ->last().cost 
  else 
    self.items -> select (i | i.number <= (totalPrice*quantity)) -> sortedBy(number) ->last().cost 
  endif

category USPostalService def:
  shippingCosts(totalWeight:Decimal, totalPrice:Money, quantity:PositiveInteger): Money = 
  calculateFromUSPS (self.userID, self.password, self.server, totalWeight, totalPrice, quantity)
```

[DR1] *Order::id* identifies the order and it is assigned automatically.

```plaintext
category Order::id():PositiveInteger
body : 
  if Order.allInstances() -> size() = 0 then 0 
  else Order.allInstances() -> sortedBy(id) -> last().id + 1 
  endif
```

[DR2] *Order::primary* address of an order is that of its customer.

```plaintext
category Order::primary():Address
body : self.customer.primary
```

[DR3] *Order::eMailAddress* of an order is that of its customer.

```plaintext
category Order::eMailAddress():EMail
body : self.customer.eMailAddress
```

[DR4] *Order::phone* of an order is that of its customer.

```plaintext
category Order::phone():String
body : self.customer.phone
```

[DR5] *Order::purchased* is the *DateTime* when the order was created

```plaintext
category Order::purchased():DateTime
body : Now()
```

[DR6] *Order::lastModified* is the last *DateTime* when the status order was modified

```plaintext
category Order::lastModified():DateTime
body : self.orderStatusChange -> sortedBy(added) -> last().added
```
CSTL and its application to the osCommerce case study.
Albert Tort

[DR7] **Order::status** is the current status of the order

```plaintext
context Order::status():OrderStatus
body : self.orderStatusChange -> sortedBy(added) -> last().orderStatus
```

[DR8] **Order::total** gives the total amount of an order

```plaintext
context Order::total():Money
body :
  let totalWithoutShippingCosts:Money =
    self.orderLine -> collect(finalPrice*quantity) -> sum()
  let totalWeight:Decimal =
    self.orderLine -> collect(product.weight*quantity) -> sum()
  let quantity:PositiveInteger =
    self.orderLine.quantity -> sum()
  let handlingFee:Money =
    if self.shippingMethod.oclIsKindOf(HandlingFeeMethod)
      then
        self.shippingMethod.oclAsType(HandlingFeeMethod).handlingFee
      else 0
    endif
  in
  let totalWeightIncreased:Decimal =
    if totalWeight* (ShippingAndPackaging.percentageIncreaseForLargerPackages/100) >
      ShippingAndPackaging.typicalPackageTareWeight
      then
        totalWeight * (1 +totalWeight* ShippingAndPackaging.percentageIncreaseForLargerPackages/100)
      else totalWeight + ShippingAndPackaging.typicalPackageTareWeight
    endif
  in
  totalWithoutShippingCosts +
  self.shippingMethod.shippingCosts
  (totalWeightIncreased, totalWithoutShippingCosts, quantity) + handlingFee
```

[DR9] **OrderStatusChange::added** is the DateTime when the change is done.

```plaintext
context OrderStatusChange::added():DateTime
body :  Now()
```

[10] **OrderLine::name** is that of its product in the default language

```plaintext
context OrderLine::name():String
body :
  self.product.productInLanguage
  -> select(pil | pil.language = Store.allInstances() -> any(true).defaultLanguage).name
```

[DR11] **OrderLine::model** is that of its product

```plaintext
context OrderLine::model():String
body :  self.product.model
```

[DR12] **OrderLine::basePrice** is the net price of the product without taking into account the selected attributes.

```plaintext
context OrderLine::basePrice():Money
body :
  if self.product.specialNetPrice ->notEmpty()
    then self.product.specialNetPrice
  else self.product.netPrice
  endif
```
CSTL and its application to the osCommerce case study.
Albert Tort

**[DR13]** OrderLine::price is the net price of the product with the selected attributes

```
context OrderLine::price():Money
body:
  if self.orderLineAttribute -> isEmpty() then self.basePrice
  else
    self.orderLineAttribute -> collect
    if sign = Sign::plus then increment
    else –increment
    endif -> sum() + self.basePrice
  endif
```

**[DR14]** OrderLine::finalPrice is the price of the product with the selected attributes and taking into account the taxes

```
context OrderLine::finalPrice():Money
body:
  if self.billing.zone -> notEmpty() then
    self.product.addTaxes(self.billing.zone, self.price)
  else self.price
  endif
```

**[DR15]** OrderLineAttribute::option is the option name in the default language

```
context OrderLineAttribute::option():String
body:
  self.attribute.option.hasOptionName
  -> select (hon | hon.optionLanguage = Store.allInstances() -> any(true).defaultLanguage).optionName
```

**[DR16]** OrderLineAttribute::value is the option value in the default language

```
context OrderLineAttribute::value():String
body:
  self.attribute.value.hasValueName
  -> select (hvn | hon.valueLanguage = Store.allInstances() -> any(true).defaultLanguage).valueName
```

**[DR17]** OrderLineAttribute::increment is the increment applied in the product price by the attribute

```
context OrderLineAttribute::increment():Money
body:
  self.attribute.productAttribute
  -> select (pa | pa.product = self.orderLine.product).increment
```

**[DR18]** OrderLineAttribute::sign is the sign of the increment applied in the product price by the attribute

```
context OrderLineAttribute::sign():Sign
body:
  self.attribute.productAttribute
  -> select (pa | pa.product = self.orderLine.product).sign
```

**[IC1]** A specific zone shipping method with a specific tax zone can only be applied if the delivery address zone is included in the tax zone.

```
context Order::ApplicableZoneShippingMethod: Boolean
body:
  self.shippingMethod.oclIsTypeOf(SpecificZoneMethod) and
  self.shippingMethod.oclAsType(SpecificZoneMethod).taxZone -> notEmpty implies
  self.shippingMethod.oclAsType(SpecificZoneMethod).taxZone.zone
  -> includes(self.delivery.zone)
```
Open session

Primary Actor: Customer
Precondition: None.
Trigger: A customer starts using the system.

Main Success Scenario:

1. The system creates an anonymous session:
   
   [⇒ NewSession]
**CSTL and its application to the osCommerce case study.**
Albert Tort

1. The system deletes the current session.
   
   \[ \rightarrow \text{DeleteSession} \]

**Extensions:**

1a. The customer is logged in and the session has a non empty shopping cart.
   
   1a1. The shopping cart is saved.

**Log in**

**Primary Actor:** Customer

**Precondition:** The customer is not logged in yet.

**Trigger:** A customer logs in the system.

**Main Success Scenario:**

1. The customer introduces their identification data.
2. The system validates the identification data.
3. The customer becomes the owner of the current session.
   
   \[ \rightarrow \text{LogIn} \]

**Extensions:**

3a. The customer has a shopping cart from a previous session.
   
   3a1. The previous shopping cart is restored.

   \[ \rightarrow \text{RestorePreviousShoppingCart} \]

3b. The current session has a non-empty and anonymous shopping cart
   
   3b1. The anonymous shopping cart becomes the current shopping cart of the customer.

**LogOut**

**Primary Actor:** Customer

**Precondition:** The customer is logged in.

**Trigger:** A customer logs out from the system.

**Main Success Scenario:**

1. The current session becomes anonymous.

   \[ \rightarrow \text{LogOut} \]

**Extensions:**

1a. The customer has a non empty shopping cart.
   
   1a1. The shopping cart is saved.
CSTL and its application to the osCommerce case study.
Albert Tort

**Change the current language**

**Primary Actor:** Customer  
**Precondition:** None.  
**Trigger:** A customer wants to change the current language of the session.

**Main Success Scenario:**

1. The store administrator selects the language which will become the current language.  
2. The system updates the current language.  
   
   \[ \rightarrow \text{SetCurrentLanguage} \]

**Change the current currency**

**Primary Actor:** Customer  
**Precondition:** None.  
**Trigger:** A customer wants to change the current currency of the session.

**Main Success Scenario:**

1. The store administrator selects the currency which will become the current currency.  
2. The system updates the current currency.  
   
   \[ \rightarrow \text{SetCurrentCurrency} \]

**Place and order**

**Primary Actor:** Customer  
**Precondition:** None.  
**Trigger:** A customer wants to place and order.

**Main Success Scenario:**

1. At any time before step 10 the customer logs in:  
   
   \[ \rightarrow \text{LogIn} \]
   
   The system adds the contents of the anonymous shopping cart to the customer shopping cart.  
2. The system displays the contents of the shopping cart.  
3. The customer browses the product catalog.  
   
   \[ \rightarrow \text{ReadProductInfo} \]
4. The customer selects a product to buy:  
   
   \[ \rightarrow \text{AddProductToShoppingCart} \]
5. The system adds the product to the shopping cart.  
6. The system displays the contents of the shopping cart.  
7. The customer changes the contents of the shopping cart:
CSTL and its application to the osCommerce case study.
Albert Tort

8. The system updates the shopping cart.
9. The system displays the contents of the updated shopping cart.
   The customer repeats steps 3, 4 and 7 as necessary to build his order.
10. The customer checks out the order.
11. The system shows the shipping address and the available shipping methods.
12. The customer selects the preferred shipping method.
13. The system shows the billing address and the available payment methods.
14. The customer selects the preferred payment method.
15. The system displays a summary of the order.
16. The customer confirms the order:
   [\rightarrow OrderConfirmation]
17. The system saves the order.
18. The system sends an email to the customer and to the store extra order emails with the information about the order.

Extensions:

1a. The customer is new:
   1a1. Create customer.
5a. The configurable option Display cart after adding a product is disabled
   The customer repeats steps 3 and 4 as necessary.
   5a1. The customer continues with the checkout procedure at step 9.
16a. The customer wants to change the contents of the shopping cart:
   16a1. The customer changes the contents of the shopping cart:
      [\rightarrow UpdateShoppingCart]
   16a2. The customer continues with the checkout procedure at step 11.
11a, 16a. The customer wants to change the shipping address:
   11a1. The system shows the known addresses of the customer.
   11a2. The customer selects a different shipping address.
   11a3. The customer continues with the checkout procedure at step 11.
13a, 16b. The customer wants to change the billing address:
   13a1. The system shows the known addresses of the customer.
   13a2. The customer selects a different billing address.
   13a3. The customer continues with the checkout procedure at step 13.
16c. The customer wants to change the shipping method:
   16c1. The customer selects the new shipping method.
   16c2. The customer continues with the checkout procedure at step 13.
16d. The customer wants to change the payment method:
   16d1. The customer selects the new payment method.
   16d2. The customer continues with the checkout procedure at step 15.
11a2a, 16a2a. The customer wants to define a new shipping address:
CSTL and its application to the osCommerce case study.
Albert Tort

11a2a1. The customer gives the new address:
   \[\text{NewCustomerAddress}\]
11a2a2. The system saves the address.
11a2a3. The customer continues with the checkout procedure at step 11.

13a2a1. The customer gives the new address:
   \[\text{NewCustomerAddress}\]
13a2a2. The system saves the address.
13a2a3. The customer continues with the checkout procedure at step 13.

### Cancel an order

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to cancel an order.

**Main Success Scenario:**

1. The store administrator selects the order to be cancelled.
2. The system asks for the confirmation of the store administrator.
3. The store administrator confirms that he wants to cancel the order:
   \[\text{CancelOrder}\]
4. The system sets the order status to cancelled.

### Add an order status

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to add a new order status.

**Main Success Scenario:**

1. The store administrator provides the details of the new order status:
   \[\text{NewOrderStatus}\]
2. The system validates that the data is correct.
3. The system saves the new order status.

### Edit an order status

**Primary Actor:** Store administrator  
**Precondition:** None.  
**Trigger:** The store administrator wants to edit an order status.
CSTL and its application to the osCommerce case study.
Albert Tort

Main Success Scenario:

1. The store administrator selects the order status to be edited.
2. The store administrator provides the new details of the selected order status:
   \[
   \text{Æ} \quad \text{EditOrderStatus}
   \]
3. The system validates that the data is correct.
4. The system saves the changes.

Delete an order status

Primary Actor: Store administrator
Precondition: The deleted order status is not the current status of any order.
Trigger: The store administrator wants to delete an order status.

Main Success Scenario:

1. The store administrator selects the order status to be deleted.
2. The store administrator confirms that he wants to delete the order status:
   \[
   \text{Æ} \quad \text{DeleteOrderStatus}
   \]
3. The system deletes the order status.

Extensions:

2a. The order status has been an status of an order:
   2a1. The system changes the status of the order status to disabled.
   2a2. The use case ends.

Change the status of an order

Primary Actor: Store administrator
Precondition: None.
Trigger: The store administrator wants to change the status of an order.

Main Success Scenario:

1. The system shows the orders and their status.
2. The store administrator selects the order which will be edited.
3. The system shows the applicable order status.
4. The store administrator selects the new status:
   \[
   \text{Æ} \quad \text{UpdateOrderStatus}
   \]
5. The system validates that the data is correct.
6. The system saves the changes.
Set cancelled order status

**Primary Actor:** Store administrator  
**Precondition:** The order status is not yet the cancelled status.  
**Trigger:** The store administrator wants to indicate to the system which order status is used to indicate that an order is cancelled.

**Main Success Scenario:**

1. The store administrator selects an order status.  
2. The system register that the selected order status represents cancelled orders.  

   \[\Rightarrow \text{SetCancelledOrderStatus}\]

Set default order status

**Primary Actor:** Store administrator  
**Precondition:** The order status is not yet the default status.  
**Trigger:** The store administrator wants to indicate to the system which order status is assign when an order is created.

**Main Success Scenario:**

1. The store administrator selects an order status.  
2. The system register that the selected order status is the default order status.  

   \[\Rightarrow \text{SetDefaultOrderStatus}\]

Events

**NewSession**

```plaintext
context NewSession::effect()  
post:  
s.oclIsNew() and  
s.oclIsTypeOf(Session) and  
s.currentCurrency=self.currentCurrency and  
s.currentLanguage=self.currentLanguage and  
s.sessionID=Session.allInstances->size()
```

159
CSTL and its application to the osCommerce case study.
Albert Tort

**DeleteSession**

```plaintext
context DeleteSession::effect()
post : not self.session@pre.oclIsKindOf(OclAny)
```

**LogIn**

```plaintext
context LogIn::customerIsNotLoggedIn (): Boolean
body : self.customer.session -> isEmpty()

case LogIn::effect()
post :
    self.session.customer = self.customer
post :
    self.customer.numberOfLogons = self.customer.numberOfLogons@pre + 1
post:
    if self.customer.customerShoppingCart->size()>0 then
        rpsc.oclIsNew() and
        rpsc.oclIsTypeOf(RestorePreviousShoppingCart) and
        rpsc.customer=self.customer and
        rpsc.session=self.session
    else
        if self.session.shoppingCart->notEmpty() then
            csc.oclIsNew() and
            csc.oclIsTypeOf(CustomerShoppingCart) and
            csc.shoppingCartItem = self.session.shoppingCart.shoppingCartItem and
            csc.customer=self.customer and
            self.session.shoppingCart=csc
        else true
        endif
    endif
```
CSTL and its application to the osCommerce case study.
Albert Tort

LogOut

context LogOut::customerIsLoggedIn (): Boolean
body : self.session.customer = self.customer

context LogOut::effect()
post : self.session.customer -> isEmpty()

SetCurrentLanguage

context ChangeCurrentLanguage::effect()
post : session.currentLanguage = self.newCurrentLanguage
post : Store.allInstances() -> any(true).switchToDefaultLanguageCurrency and self.newCurrentCurrentLanguage.defaultCurrency -> notEmpty()
implies ccc.oclIsNew() and ccc.oclIsTypeOf(ChangeCurrentCurrency) and ccc.session = self.session and ccc.newCurrentCurrency = self.language.defaultCurrency
CSTL and its application to the osCommerce case study.
Albert Tort

**SetCurrentCurrency**

```
context SetCurrentCurrency::effect()
post : self.session.currentCurrency = self.newCurrentCurrency
```

**RestorePreviousShoppingCart**

```
context RestorePreviousShoppingCart::CustomerHasAPreviousShoppingCart(): Boolean
body : self.customer.customerShoppingCart->notEmpty()
```

```
context RestorePreviousShoppingCart::effect()
post : self.session.shoppingCart = self.customer.customerShoppingCart
```

**SetDefaultOrderStatus**

```
context SetPendingOrderStatus::effect()
post : self.myStore.defaultStatus = self.orderStatus
```
CSTL and its application to the osCommerce case study.
Albert Tort

SetCancelledOrderStatus

context SetCancelledOrderStatus::effect()
post : self.myStore.cancelledStatus = self.orderStatus

ReadProductInfo

context ReadProductInfo::effect()
post : self.product.productInLanguage->select(pil | pil.language=self.language).viewed =
    self.product@pre.productInLanguage@pre->select(pil | pil.language=self.language).viewed + 1

AddProductToShoppingCart

context AddProductToShoppingCart::AttributesAreFromProduct(): Boolean
body : self.product.attribute -> includesAll(self.attribute)

context AddProductToShoppingCart::AttributesAreOfDifferentOptions(): Boolean
body : self.attribute -> isUnique(option)
CSTL and its application to the osCommerce case study.
Albert Tort

context AddProductToShoppingCart::effect()
  post ShoppingCartItemIsCreated :
    sci.oclIsNew and
    sci.oclIsTypeOf(ShoppingCartItem) and
    sci.quantity = self.quantity and
    sci.product = self.product and
    sci.attribute = self.attribute and
    if self.session.shoppingCart -> notEmpty() then
      --The session has a shopping cart
      self.session.shoppingCart.shoppingCartItem -> includes(sci)
    else
      --The session does not have a shopping cart
      if self.session.customer -> isEmpty() then
        --The session is Anonymous
        sc.oclIsNew() and
        sc.oclIsTypeOf(AnonymousShoppingCart) and
        self.session.shoppingCart = sc and
        sc.shoppingCartItem -> includes(sci)
      else
        --The customer has logged in
        if self.session.customer.customerShoppingCart -> notEmpty() then
          --The customer has a previous shopping cart
          self.session.shoppingCart.shoppingCartItem -> includes(sci)
        else
          --The customer does not have a previous shopping cart
          csc.oclIsNew() and
          csc.oclIsTypeOf(CustomerShoppingCart) and
          self.session.shoppingCart = csc and
          csc.shoppingCartItem -> includes(sci)
        endif
      endif
    endif

UpdateShoppingCart

context UpdateShoppingCart::complete(): Boolean
  body : self.lineChange->size() = self.session.shoppingCart.shoppingCartItem->size()

context RemoveProduct::effect()
  post : not self.shoppingCartItem@pre.oclIsKindOf(OclAny)
CSTL and its application to the osCommerce case study.
Albert Tort

context ChangeQuantity::effect()
post: self.shoppingCartItem.quantity = self.quantity

context UpdateShoppingCart::effect()
post:
self.lineChange ->forall
(lc | let cartItem: ShoppingCartItem =
 self.shoppingCart.shoppingCartItem->
at(lineChange->indexOf(lc))

 in
(lc.remove or lc.quantity <> cartItem.quantity)
implies
if lc.remove then
r.p.oclIsNew and
r.p.oclIsTypeOf(RemoveProduct) and
r.p.shoppingCartItem = cartItem
else
cq.oclIsNew() and
cq.oclIsTypeOf(ChangeQuantity) and
cq.shoppingCartItem = cartItem and
cq.quantity = quantity
endif )

CancelOrder

Order

ExistingOrderEvent

DomainEvent

CancelOrder

effect()

context CancelOrder::effect()
post:
self.order.orderStatusChange ->sortedBy(added) -> last().orderStatus =
Store.allInstances() ->any(true).cancelledStatus

NewOrderStatus

DomainEvent

Language

HasOrderStatusName

String

orderStatus
Name

OrderStatusNameEvent

NewOrderStatus

effect()
CSTL and its application to the osCommerce case study.
Albert Tort

```cstl
context NewOrderStatus::orderStatusDoesNotExist(): Boolean
body:
not OrderStatus.allInstances -> exists {os |
    Language.allInstances->exists({l |
        self.hasOrderStatusName->select(languageOfOrderStatus=l).orderStatusName =
        os.orderStatusInLanguage->select(language=l).name})
}

context NewOrderStatus::effect()
post:
    os.oclIsNew() and
    os.oclIsTypeOf(OrderStatus) and
    Language.allInstances->forAll({l |
        self.hasOrderStatusName->select(languageOfOrderStatus=l).orderStatusName.string =
        os.orderStatusInLanguage->select(language=l).name})

EditOrderStatus

context EditOrderStatus::orderStatusDoesNotExist(): Boolean
body:
Language.allInstances -> forAll ({l |
    l.orderStatusInLanguage.name ->excludes(self.hasOrderStatusName -> any(languageOfOrderStatus=l).orderStatusName)
    or
    l.orderStatusInLanguage->any(orderStatus=self.orderStatus).name =
    self.hasOrderStatusName->any(languageOfOrderStatus=l).orderStatusName)

context EditOrderStatus::effect()
post:
Language.allInstances -> forAll({l |
    self.hasOrderStatusName->select(languageOfOrderStatus=l).orderStatusName =
    self.orderStatus.orderStatusInLanguage->select(language=l).name)
```
CSTL and its application to the osCommerce case study.
Albert Tort

**DeleteOrderStatus**

- **OrderStatus**
  - 1

- **ExistingOrderStatusEvent**
  - **DomainEvent**

- **DeleteOrderStatus**
  - effect()

- **context** DeleteOrderStatus:: IsNotTheCurrentStatusOfAnyOrder(): Boolean
  - **body**:
    - Order.allInstances() -> forAll (o | o.orderStatusChange -> sortedBy(added) -> last().orderStatus <> self.orderStatus)

- **context** DeleteOrderStatus:: IsNotADefaultStatus(): Boolean
  - **body**:
    - Store.allInstances -> forAll(s | s.defaultStatus <> self.orderStatus and s.cancelledStatus <> self.orderStatus)

- **context** DeleteOrderStatus:: effect()
  - **post**:
    - if Order.allInstances.orderStatus->includes(self.orderStatus)
      - then self.orderStatus.status=Status::disabled
    - else OrderStatus.allInstances->excludes(self.orderStatus@pre)
    - endif

**UpdateOrderStatus**

- **Order**
  - 1

- **ExistingOrderEvent**
  - **DomainEvent**

- **UpdateOrderStatus**
  - comments : String [0..1]
  - newOrderStatus : OrderStatus
  - effect()

- **context** ChangeOrderStatus:: effect()
  - **post**:
    - osc.ocllsNew() and
    - osc.ocllsTypeOf(OrderStatusChange) and
    - osc.comments = self.comments and osc.order = self.order and
    - osc.orderStatus = self.newOrderStatus
CSTL and its application to the osCommerce case study.
Albert Tort

OrderConfirmation

```cstl
context OrderConfirmation::ShippingMethodIsEnabled(): Boolean
body: self.shippingMethod.status = Status::enabled

context OrderConfirmation::PaymentMethodIsEnabled(): Boolean
body: self.paymentMethod.status = Status::enabled

context OrderConfirmation::CurrencyIsEnabled(): Boolean
body: self.currency.status = Status::enabled

context OrderConfirmation::CreditCardDetailsNeeded(): Boolean
body:
self.paymentMethod.oclIsTypeOf(AuthorizeNet) or
self.paymentMethod.oclIsTypeOf(CreditCard) or
self.paymentMethod.oclIsTypeOf(IPayment) or
self.paymentMethod.oclIsTypeOf(TwoCheckOut) or
self.paymentMethod.oclIsTypeOf(PSiGate)
implies
creditCardTypenotEmpty() and
creditCardOwnernotEmpty() and
creditCardNumbernotEmpty() and
creditCardExpiresnotEmpty()

context OrderConfirmation::StockAllowsOrder(): Boolean
body:
Stock.allowCheckout or
not Stock.checkStockLevel or
self.shoppingCart.shoppingCartItem.product -> forAll (p | p.quantityOnHand > 0)

context OrderConfirmation::effect()
post theOrderIsCreated:
o.oclIsNew() and
do.oclIsTypeOf(Order) and
do.customer = self.customer @ pre and
do.billing = self.billing and
do.delivery = self.delivery and
do.shippingMethod = self.shippingMethod and
do.paymentMethod = self.paymentMethod and
do.currency = self.currency and
```
CSTL and its application to the osCommerce case study.
Albert Tort

--The initial status of the order is pending
osc.oclIsNew() and
osc.oclIsTypeOf(OrderStatusChange) and
osc.comments = self.comments and
osc.orderStatus = Store.allInstances() -> any(true).defaultStatus and
osc.order = o and

--There is an order line for each shopping cart item
shoppingCart@pre.shoppingCartItem@pre->forAll(i|OrderLine.allInstances() -> one
(ol|ol.order = o and
 ol.product = i.product@pre and
 ol.quantity = i.quantity@pre and
 i.attribute@pre->forAll(iAtt|OrderLineAttribute.allInstances() -> one
 (olAtt|olAtt.orderLine = ol and
 olAtt.attribute = iAtt))))

post theShoppingCartIsRemoved:
 not self.shoppingCart@pre.oclIsKindOf(OclAny)

post updateProductQuantities:
let productsBought:Set(Product) =
 self.shoppingCart@pre.shoppingCartItem@pre.product@pre->asSet()
in productsBought -> forAll(p)
let quantityBought:PositiveInteger =
 self.shoppingCart@pre.shoppingCartItem@pre->select
 (sc | sc.product = p).quantity -> sum()
in
 p.quantityOrdered = p.quantityOrdered@pre + quantityBought and
 Stock.substractStock implies
 p.quantityOnHand = p.quantityOnHand@pre – quantityBought)

Example test programs

testprogram SessionsManagement{ 
 co:= new Country; 
 a:= new Address(country:=co); 
 c:= new Customer(address:=a, primary:=a); 
 //Language l has no default currency
 l1:= new Language(name:='Language1', code:='L1'); 
 cu:=new Currency(title:='Currency1',code:='C1');
 cu2:=new Currency(title:='Currency2',code:='C2');
 //Language l2 has a default currency
 l2:= new Language(name:='Language2', code:='L2',defaultCurrency:=cu2); 
 //Language l3 has no default currency
 l3:= new Language(name:='Language3', code:='L3');

test OpenSession{ 
 new NewSession(currentLanguage:=l, currentCurrency:=cu) occurs; 
}

test InvalidLogIn{ 
 ns:=new NewSession(currentLanguage:=l, currentCurrency:=cu) occurs; 
 new LogIn(session:=ns.createdSession, customer:=c) occurs; 
 //A logged-in customer cannot log in
 new LogIn(session:=ns.createdSession, customer:=c) may not occur;
 //...even if the customer tries to log in another session
 delete ns;
 ns2:=new NewSession(currentLanguage:=l, currentCurrency:=cu) occurs;
 new LogIn(session:=ns.createdSession, customer:=c) may not occur;
}

test InvalidLogOut{ 
 //We cannot log out if the customer is not logged in the session
 ns:=new NewSession(currentLanguage:=l, currentCurrency:=cu) occurs;
 new LogOut(session:=ns.createdSession, customer:=c) may not occur;
}
CSTL and its application to the osCommerce case study.
Albert Tort

test LogInLogOutWithoutPreviousShoppingCart{
    ns:=new NewSession(currentLanguage:=l, currentCurrency:=cu) occurs;
    new LogIn(session:=ns.createdSession, customer:=c) occurs;
    new LogOut(session:=ns.createdSession, customer:=c) occurs;
}

test LogInLogOutWithPreviousShoppingCart{
    //The customer navigates in the store in an anonymous session
    ns:=new NewSession(currentLanguage:=l, currentCurrency:=cu) occurs;
    p:= new Product;
    assert true ns.createdSession.customer.isUndefined();
    new AddProductToShoppingCart(session:=ns.createdSession, product:=p,
                                   quantity:=1) occurs;
    assert true ns.createdSession.shoppingCart.oclIsTypeOf(AnonymousShoppingCart);
    assert equals ns.createdSession.shoppingCart.shoppingCartItem.product->asSet()
                  Set{p};

    //The customer logs in
    new LogIn(session:=ns.createdSession, customer:=c) occurs;
    assert true ns.createdSession.customer.oclIsTypeOf(CustomerShoppingCart);
    assert equals ns.createdSession.customer c;
    assert equals ns.createdSession.customerShoppingCart.shoppingCartItem.product->asSet()
                  Set{p};

    //The customer adds another product
    p2:=new Product;
    new AddProductToShoppingCart(session:=ns.createdSession, product:=p2,
                                  quantity:=2) occurs;

    //The customer logs out
    new LogOut(session:=ns.createdSession, customer:=c) occurs;

    //If the customer logs in again, the previous customer shopping cart is restored
    assert true ns.createdSession.customer.oclIsTypeOf(CustomerShoppingCart);
    assert equals ns.createdSession.customerShoppingCart.customer c;
    assert equals ns.createdSession.customerShoppingCart.shoppingCartItem.product->asSet()
                  Set{p,p2};

    //The session is finished
    new DeleteSession(session:=ns.createdSession) occurs;
}

abstract test changeCurrentLanguage(
    Boolean switch, Language newLanguage,
    Language expectedLanguage, Currency expectedCurrency)
{
    //Store Initialization
    s:=new Store(name:='FashionTShirts');
    english:=new Language(name:='English', code:='EN');
    s.defaultLanguage:=english;
    dollar:=new Currency(title:='USDollar', code:='USD', status:=$enabled);
    s.defaultCurrency:=dollar;
    s.country:=usa;
    cos:=new OrderStatus;
    cosl:=new OrderStatusInLanguage(language:=english, orderStatus:=cos);
    cosl.name:='cancelled';
    s.cancelledStatus:=cos;
    dos:=new OrderStatus;
    dosl:=new OrderStatusInLanguage(orderStatus:=dos, language:=english);
    dosl.name:='pending';
    s.defaultStatus:=dos;

    //Switch to default language currency initialization
    s.switchToDefaultLanguageCurrency:=switch;

    ns:=new NewSession(currentLanguage:=l, currentCurrency:=cu) occurs;
    new SetCurrentLanguage(session:=ns.createdSession,
                             newCurrentLanguage:=newLanguage) occurs;
    assert equals ns.createdSession.currentLanguage expectedLanguage;
    assert equals ns.createdSession.currentCurrency expectedCurrency;
}
CSTL and its application to the osCommerce case study.
Albert Tort

```java
// We test the effect of the "switch to default language" configuration value

test changeCurrentLanguage(switch:=false, newLanguage:=l, expectedLanguage:=l, expectedCurrency:=cu);

test changeCurrentLanguage(switch:=true, newLanguage:=l3, expectedLanguage:=l3, expectedCurrency:=cu);

test changeCurrentLanguage(switch:=true, newLanguage:=l2, expectedLanguage:=l2, expectedCurrency:=cu2);
}

testprogram OrderConfirmation{
  // Store initialization
  s:= new Store(name:='FashionTShirts');
  english:= new Language(name:='English', code:='EN');
  s.defaultLanguage:= english;
  dollar:= new Currency(title:='USDollar', code:='USD', status:=#enabled);
  s.defaultCurrency:= dollar;
  s.country:= usa;
  cos:= new OrderStatus;
  cosl:= new OrderStatusInLanguage(language:=english, orderStatus:=cos);
  cosl.name:='cancelled';
  s.cancelledStatus:= cos;
  dos:= new OrderStatus;
  dosl:= new OrderStatusInLanguage(orderStatus:=dos, language:=english);
  dosl.name:='pending';
  s.defaultStatus:= dos;

  // Product attributes initialization
  ssize := new Option;
  extraLarge:= new Value;
  small:= new Value;
  smallSize:= new Attribute(option:= ssize, value:=small);
  extraLargeSize:= new Attribute(option:= ssize, value:=extraLarge);

  sizeName := new StringDT(string:='size');
  new HasOptionName(option:= ssize, optionName:=sizeName, optionLanguage:=english);
  extraLargeName := new StringDT(string:='extraLarge');
  new HasValueName(value:= extraLarge, valueName:=extraLargeName, valueLanguage:=english);

  smallName := new StringDT(string:='small');
  new HasValueName(value:= small, valueName:=smallName, valueLanguage:=english);

  stock := new Stock;
  stock.checkStockLevel:= true;
  stock.subtractStock:= true;

  // Products initialization
  fashionTShirt := new Product(netPrice:=10, quantityOnHand:=50);

  smallFashionTShirt:= new ProductAttribute(product:=fashionTShirt, attribute:=smallSize);
  smallFashionTShirt.increment:=2;
  smallFashionTShirt.sign:= #minus;

  extraLargeFashionTShirt:= new ProductAttribute(product:=fashionTShirt, attribute:=extraLargeSize);
  extraLargeFashionTShirt.increment:=1;
  extraLargeFashionTShirt.sign:= #plus;

  // Customer session initialization and log in
  a:= new Address(country:=usa);
  c := new Customer(address:=a, primary:=a);
  ns:= new NewSession(currentLanguage:=english, currentCurrency:=dollar) occurs;
  new LogIn(session:=ns.createdSession, customer:=c) occurs;

  fixturecomponent addRegularSizedTShirts{
    new AddProductToShoppingCart(session:=ns.createdSession, product:=fashionTShirt, quantity:=3) occurs;
  }

  fixturecomponent addSpecialSizedTShirts{
    new AddProductToShoppingCart(session:=ns.createdSession, product:=fashionTShirt, quantity:=2, attribute:=smallSize) occurs;
  }
```
CSTL and its application to the osCommerce case study.
Albert Tort

```java
new AddProductToShoppingCart(session:=ns.createdSession,
product:=fashionTShirt, quantity:=1,
attribute:=extraLargeSize) occurs;
}
abstract test confirmedOrderTotal (Fixture itemsAddition, Real expectedTotal){
  load $itemsAddition;
  sm:= new FlatRate(status:='#enabled);
  pm:= new Nochex(status:='#enabled);
  oc := new OrderConfirmation
  (shoppingCart:=ns.createdSession.shoppingCart,
  currency:=dollar , shippingMethod:=sm, paymentMethod:=pm)
  occurs;
  assert equals oc.orderCreated.total() expectedTotal;
}
test confirmedOrderTotal
(itemsAddition:=addRegularSizedTShirts,expectedTotal:=30.0);
test confirmedOrderTotal
(itemsAddition:=addSpecialSizedTShirts,expectedTotal:=27.0);
}
testprogram CreateAndEditStatus{
    english:=new Language(name:='English', code:='EN');
    test newOrderStatus{
      pendingInEnglish:=new StringDT(string:='pending');
      nos:=new NewOrderStatus;
      newHasOrderStatusName(orderStatusName:=pendingInEnglish,
      languageOfOrderStatus:='english', orderStatusNameEvent:=nos);
      nos occurs;
      //We cannot create two order status with the same name
      nos2:=new NewOrderStatus;
      new HasOrderStatusName(orderStatusName:=pendingInEnglish,
      languageOfOrderStatus:='english', orderStatusNameEvent:=nos2);
      nos2 may not occur;
    }
    test editOrderStatus{
      pendingInEnglish:=new StringDT(string:='pending');
      nos:=new NewOrderStatus;
      new HasOrderStatusName(orderStatusName:=pendingInEnglish,
      languageOfOrderStatus:='english', orderStatusNameEvent:=this);
      nos occurs;
      cancelledInEnglish:=new StringDT(string:='cancelled');
      nos2:=new NewOrderStatus;
      new HasOrderStatusName;
      orderStatusName:=cancelledInEnglish, languageOfOrderStatus:='english',
      orderStatusNameEvent:=nos2);
      nos2 occurs;
      //VALID EDITIONS
      deliveredInEnglish:=new StringDT(string:='delivered');
      //It is possible to edit an order status without no name changes
      eos:=new EditOrderStatus(orderStatus:=nos.createdOrderStatus);
      new HasOrderStatusName(orderStatusName:=cancelledInEnglish,
      languageOfOrderStatus:='english', orderStatusNameEvent:=eos);
      eos occurs;
      eos2:=new EditOrderStatus(orderStatus:=nos.createdOrderStatus);
      new HasOrderStatusName(orderStatusName:=deliveredInEnglish,
      languageOfOrderStatus:='english', orderStatusNameEvent:=eos2);
      eos2 occurs;
      //INVALID EDITIONS
      //The edition of an order status cannot cause duplicated order status
      eos3:=new EditOrderStatus(orderStatus:=nos.createdOrderStatus);
      new HasOrderStatusName(orderStatusName:=pendingInEnglish,
      languageOfOrderStatus:='english', orderStatusNameEvent:=this);
      eos3 may not occur;
    }
}
```
CSTL and its application to the osCommerce case study.
Albert Tort

testprogram DeleteOrderStatus{

english:=new Language(name:='English', code:='EN');

//We create the order statuses
pending:=new OrderStatus;
posl:=new OrderStatusInLanguage(orderStatus:=pending, language:=english);
posl.name:=('pending');
cancelled:=new OrderStatus;
cosl:=new OrderStatusInLanguage(orderStatus:=cancelled, language:=english);
cosl.name:=('cancelled');
delivered:=new OrderStatus;
dosl:=new OrderStatusInLanguage(orderStatus:=delivered, language:=english);
dosl.name:=('delivered');
returned:=new OrderStatus;
rosl:=new OrderStatusInLanguage(orderStatus:=returned, language:=english);
rosl.name:=('returned');

//We initialize an store
euro:=new Currency(title:='Euro', code:='EUR', status:.#enabled);

//Store configuration
s:=new Store;
s.defaultLanguage:=english;
s.defaultCurrency:=euro;
s.country:=usa;
s.defaultStatus:=pending;
s.canceledStatus:=cancelled;

//Stock configuration
stock := new Stock;
stock.checkStockLevel:=true;
stock.subtractStock:=true;

//Products configuration
standardLaptop := new Product(netPrice:=949, quantityOnHand:=300);

//Payment methods configuration
pm:=new CashOnDelivery(status:=#enabled);

//Shipping configuration
sm:=new PerItem(status:=#enabled, handlingFee:=5, cost:=10);

//We create an order which, initially, has the pending status (by default)
//Customer initialization and login
a:= new Address(country:=usa);
c := new Customer(address:=a,primary:=a);
ns:=new NewSession(currentLanguage:=english, currentCurrency:=euro) occurs;
new LogIn(session:=ns.createdSession, customer:=c) occurs;
new AddProductToShoppingCart(session:=ns.createdSession,
product:=standardLaptop,quantity:=2) occurs;
oc := new OrderConfirmation(shoppingCart:=ns.createdSession.shoppingCart,
currency:=euro , shippingMethod:=sm, paymentMethod:=pm, billing:=a) occurs;
orderCreated:=oc.orderCreated;

test deleteOrderStatusIfNoOrdersUsedIt{
    //If the order status has not been used, it can be deleted at all
    new DeleteOrderStatus(orderStatus:=delivered) occurs;
    assert false OrderStatus.allInstances->exists(orderStatusInLanguage
    ->any(language=english).name='delivered');
}

test deleteStoreDefaultOrderStatus{
    //A default status of the store cannot be deleted
    new DeleteOrderStatus(orderStatus:=pending) may not occur;
    new DeleteOrderStatus(orderStatus:=cancelled) may not occur;
}

test deleteOrderStatusIfItIsTheCurrentStatusOfAnOrder{
    //If the order status is the current status of an order, the deletion
    //only changes its status to disabled
    new UpdateOrderStatus(order:=orderCreated,newOrderStatus:=delivered)
    occurs;
    new DeleteOrderStatus(orderStatus:=delivered) may not occur;
}
CSTL and its application to the osCommerce case study.
Albert Tort

```java
int test = deleteOrderStatusIfItWasTheStatusOfAnOrder{
    // If the order status was the status of an order (not the current
    // status) the system disables the order status.
    New UpdateOrderStatus(order:=orderCreated,newOrderStatus:=delivered)
    occurs;
    New UpdateOrderStatus(order:=orderCreated,newOrderStatus:=returned)
    occurs;
    new DeleteOrderStatus(orderStatus:=delivered) occurs;
    assert equals delivered.status #disabled;
}

Finally, we present a test program that test a typical scenario of the use case “Place and Order”
which is the main functionality of the system from the customers point of view.

```java
testprogram PlaceAndOrder{
    // STORE INITIALIZATION
    // Location, currencies and languages
catalonia:=new Zone(name:='Catalonia', code:='CAT', country:=spain);
    english:=new Language(name:='English', code:='EN');
euro:=new Currency(title:='Euro', code:='EUR', status:=#enabled);

    // Store configuration
    s:=new Store(name:='CustomizedComputers');
s.defaultLanguage:=english;
s.defaultCurrency:=euro;
s.country:=spain;
s.zone:=catalonia;

    // Default order status
    cancelled:=new OrderStatus;
cosl:=new OrderStatusInLanguage(language:=english,orderStatus:=cancelled);
cosl.name:='cancelled';
s.cancelledStatus:=cancelled;
pending:=new OrderStatus;
dosl:=new OrderStatusInLanguage(orderStatus:=pending, language:=english);
dosl.name:='pending';
d.s.defaultStatus:=pending;
delivered:=new OrderStatus;
deosl:=new OrderStatusInLanguage(orderStatus:=delivered, language:=english);
deosl.name:='delivered';

    // Stock configuration
    stock := new Stock;
    stock.checkStockLevel:=true;
    stock.substractStock:=true;

    // Product attributes initialization
    warranty := new Option;
    premium:=new Value;
    plus:=new Value;

    premiumWarranty:=new Attribute(option:=warranty, value:=premium);
    plusWarranty:=new Attribute(option:=warranty, value:=plus);

    warrantyName := new StringDT(string:='Warranty');
    new HasOptionName(option:=warranty,
        optionName:=warrantyName, optionLanguage:=english);

    premiumName := new StringDT(string:='Premium');
    new HasValueName(value:=premium,
        valueName:=premiumName, valueLanguage:=english);

    plusName := new StringDT(string:='Plus');
    new HasValueName(value:=plus, valueName:=plusName, valueLanguage:=english);

    // Products initialization
    standardLaptop := new Product(netPrice:=949, quantityOnHand:=300);
    plusWarrantyLaptop:= new ProductAttribute(product:=standardLaptop,
        attribute:=plusWarranty);
```
CSTL and its application to the osCommerce case study.
Albert Tort

plusWarrantyLaptop.increment:=60;
plusWarrantyLaptop.sign:=#plus;

premiumWarrantyLaptop:= new ProductAttribute(product:=standardLaptop,
attribute:=premiumWarranty);
premiumWarrantyLaptop.increment:=112;
premiumWarrantyLaptop.sign:=#plus;

illustratedStartGuide:= new Product(netPrice:=15,quantityOnHand:=50);

// Taxes configuration
spanishVAT:= new TaxZone(name:='SpanishVAT');
spanishVAT.zone:=catalonia;

// We allow two types of VAT: general VAT (16%) and super-reduced VAT (4%)
general:= new TaxClass(name:='generalVAT');
superreduced:= new TaxClass(name:='super-reducedVAT');

// For each TaxClass, there is a different tax rate applied in each zone
generalRate:= new TaxRate(taxClass:=general, taxZone:=spanishVAT);
generalRate.rate:=16;
generalRate.priority:=1;
superReducedRate:= new TaxRate(taxClass:=superreduced, taxZone:=spanishVAT);
superReducedRate.rate:=4;
superReducedRate.priority:=1;

standardLaptop.taxClass:=general;
illustratedStartGuide.taxClass:=superreduced;

// Payment methods configuration
pm:= new CashOnDelivery(status:#enabled);

// Shipping configuration
sm:= new PerItem(status:#enabled, handlingFee:=5, cost:=10);

test placeAndOrder {
  // Customer initialization
  a:= new Address(country:=spain, zone:=catalonia, state:='Catalonia');
  c := new Customer(address:=a,primary:=a);
  // The customer logs in
  ns:= new NewSession(currentLanguage:=english, currentCurrency:=euro)
  occurs;
  /*
  The customer adds to the shopping cart the following items:
  - 2 standard laptops with no warranty
  - Standard laptop with Premium warranty
  - Illustrated Start guide
  */
  new AddProductToShoppingCart(session:=ns.createdSession,
      product:=standardLaptop,quantity:=2) occurs;
  new AddProductToShoppingCart(session:=ns.createdSession,
      product:=standardLaptop,quantity:=1,
      attribute:=premiumWarranty) occurs;
  new AddProductToShoppingCart(session:=ns.createdSession,
      product:=illustratedStartGuide,quantity:=1) occurs;
  new LogIn(session:=ns.createdSession, customer:=c) occurs;
  sc:=ns.createdSession.shoppingCart;
  oc := new OrderConfirmation
      (shoppingCart:=ns.createdSession.shoppingCart, currency:=euro,
       shippingMethod:=sm, paymentMethod:=pm, billing:=a) occurs;
  orderCreated:=oc.orderCreated;

  assert equals orderCreated.orderLine.product->asSet()->size() 2;
  assert equals orderCreated.orderLine
    ->select(product=standardLaptop).quantity->sum() 3;
  assert equals orderCreated.orderLine
    ->select(product=illustratedStartGuide).quantity->sum() 1;

  assert equals standardLaptop.quantityOnHand 297;
  assert equals illustratedStartGuide.quantityOnHand 49;
  */
CSTL and its application to the osCommerce case study.
Albert Tort

Order total details
=========================
2 x standard laptop (no warranty)  x 949 = 1898,00
1 x standard laptop (premium warranty)  x 1061 = 1061,00
Subtotal ............................................ 2959,00
VAT 16% ............................................... 473,44
Total (16%) ......................................... 3432,44

1 x illustrated start guide  x 15 = 15,00
Subtotal ............................................ 15,00
VAT 4% ............................................... 0,60
Total (4%) ........................................... 15,60

---Shipping costs (Per Item)
Handling fee ........................................... 5,00
4 x Per Item Rate  x 10 = 40,00

Order Total ...................................................................... 3493,04

*/

assert equals orderCreated.total() 3493.04;

//The store administrator can change the status of the order...
new UpdateOrderStatus(order:=orderCreated,newOrderStatus:=delivered) occurs;
assert equals orderCreated.orderStatus Sequence{pending,delivered};

//...or he can cancel the order (order information cannot be deleted)
new CancelOrder(order:=orderCreated) occurs;
assert equals orderCreated.orderStatus Sequence{pending,delivered,cancelled};
CSTL and its application to the osCommerce case study.
Albert Tort

References

Appendix A: Executable Conceptual Schema of the osCommerce System

model osCommerce
  -- Enumerations
  enum SortOrder{ascending, descending}
  enum SortField{productName, expectedDate}
  enum Operator{AND, OR}
  enum TransactionMode{test, production}
  enum TransactionMethod{creditCar, eCheck}
  enum PSIGateMode{production, alwaysGood, alwaysDuplicate, alwaysDecline}
  enum PSIGateType{sale, preAuth, postAuth}
  enum PSIGateCollection{local, remote}
  enum SECPayMode{alwaysSuccessful, alwaysFall, production}
  enum Status{enabled, disabled}
  enum USPServer{test, production}
  enum ShippingTableMethod{weight, price}
  enum ProductStatus{inStock, outOfStock}
  enum Sign{plus, minus}
  enum NewsletterStatus{locked, unlocked}
  enum Gender{male, female}
  enum Rating{oneStar, twoStars, threeStars, fourStars, fiveStars}

  -- DataTypes
  class EMail
      attributes
        eMail:String
      end
  class File
      attributes
        fileName:String
      end
  class URL
      attributes
        url:String
      end
  class PostalCode
      attributes
        postalCode:String
      end
  class ShippingTableItem
      attributes
        number:Integer
        cost:Integer
      end
  class DateTime
      attributes
        dateTime:String
      end
  class Date
      attributes
        date:String
      end
-- STRUCTURAL SCHEMA
-- STORE CONFIGURATION
-- Store Data
  class Store
      attributes
        name:String
      end
CSTL and its application to the osCommerce case study.
Albert Tort

owner: String
eMailAddress: EMail
eMailFrom: EMail
expectedSortOrder: SortOrder
expectedSortField: SortField
displayCartAfterAddingProduct: Boolean
allowGuestToTellAFriend: Boolean
defaultSearchOperator: Operator
storeAddressAndPhone: String
taxDecimalPlaces: Integer
displayPricesWithTax: Boolean
switchToDefaultLanguageCurrency: Boolean
end

class NameEMail
end

association store_sendExtraOrderEMail between
  Store [*]
  NameEMail[*] role sendExtraOrderEMail
end

association store_defaultLanguage between
  Store [*]
  Language[1] role defaultLanguage
end

association store_defaultCurrency between
  Store [*]
  Currency[1] role defaultCurrency
end

association store_Country between
  Store [0..1]
  Country[1]
end

association store_zone between
  Store [0..1]
  Zone[0..1]
end

association store_cancelledStatus between
  Store [*] role storeOfCancelledStatus
  OrderStatus[1] role cancelledStatus
end

association store_defaultStatus between
  Store [*] role storeOfDefaultStatus
  OrderStatus[1] role defaultStatus
end

-- Minimum and maximum values

class MinimumValues
attributes
  firstName: Integer
  lastName: Integer
dateOfBirth: Integer
eMailAddress: Integer
streetAddress: Integer
companyName: Integer
postCode: Integer
city: Integer
state: Integer
telephoneNumber: Integer
password: Integer
creditCardOwnerName: Integer
creditCardNumber: Integer
reviewText: Integer
end
CSTL and its application to the osCommerce case study.
Albert Tort

```plaintext
class MaximumValues
attributes
  addressBookEntries:Integer
end

-- Customer details configuration
class CustomerDetails
attributes
  gender:Boolean
  dateOfBirth:Boolean
  company:Boolean
  suburb:Boolean
  state:Boolean
end

-- Shipping and Packaging configuration
class ShippingAndPackaging
attributes
  postCode:PostalCode
  maximumPackageWeight:Integer
  typicalPackageTareWeight:Integer
  percentageIncreaseForLargerPackages:Integer
end

association shippingAndPackaging_countryOfOrigin between
  ShippingAndPackaging [0..1]
  Country[1] role countryOfOrigin
end

-- Download configuration
class Download
attributes
  enableDownload:Boolean
  daysExpiryDelay:Integer
  maximumNumberOfDownloads:Integer
end

-- Stock configuration
class Stock
attributes
  checkStockLevel:Boolean
  substractStock:Boolean
  allowCheckout:Boolean
  stockReOrderLevel:Integer
end

-- Payment methods
abstract class PaymentMethod
attributes
  status:Status
end

association paymentMethod_orderStatus between
  PaymentMethod [*]
  OrderStatus[0..1]
end

association paymentMethod_taxZone between
  PaymentMethod[*]
  TaxZone[0..1]
end

class AuthorizeNet < PaymentMethod
attributes
  username:String
  key:String
  mode:TransactionMode
  method:TransactionMethod
  notification:Boolean
end
```
CSTL and its application to the osCommerce case study.
Albert Tort

class CreditCard < PaymentMethod
attributes
   splitCreditCardToMail:EMail
end

class CashOnDelivery < PaymentMethod
end

class CheckInteger < PaymentMethod
attributes
   makePayableTo: String
end

class Nochex < PaymentMethod
attributes
   eMail: EMail
end

class TwoCheckOut < PaymentMethod
attributes
   login: String
   mode: TransactionMode
   merchantNotification: Boolean
end

abstract class SpecificCurrencyPaymentMethod < PaymentMethod
end

association specificCurrencyPaymentMethod_currency between
   SpecificCurrencyPaymentMethod[*]
   Currency[*]
end

class PSiGate < SpecificCurrencyPaymentMethod
attributes
   merchantID: String
   mode: PSiGateMode
   type: PSiGateType
   creditCardCollection: PSiGateCollection
end

class SECPay < SpecificCurrencyPaymentMethod
attributes
   merchantID: String
   mode: SECPayMode
end

class IPayment < SpecificCurrencyPaymentMethod
attributes
   account: Integer
   user: String
   password: String
end

class PayPal < SpecificCurrencyPaymentMethod
attributes
   eMail: EMail
end

class CheckMoney < PaymentMethod
attributes
   makePayableTo: String
end

-- Shipping methods
class ShippingMethod
attributes
   status: Status
operations
   addTaxes(z: Zone, basePrice: Real) : Real =
      let appliedTaxRates: Set(TaxRate) =
         z.taxZone.taxRate -> select (tr | tr.taxClass = self.taxClass) -> asSet()
CSTL and its application to the osCommerce case study.
Albert Tort

```plaintext
let priorities:Set(Integer) = 
    if appliedTaxRates -> isEmpty() then oclEmpty(Set(Integer))
    else appliedTaxRates -> sortedBy(priority).priority -> asSet()
    endif
in
    if priorities -> isEmpty() then basePrice
    else priorities -> iterate (p:Integer; res:Real = 0 |
        res +
        (((appliedTaxRates -> select (tr | tr.priority = p).rate
        -> sum()) / 100)+1)*basePrice)
    endif

shippingCosts(totalWeight:Real, totalPrice:Real, quantity:Integer): Real =
    if self.oclIsTypeOf(PerItem) then
        self.oclAsType(PerItem).shippingCosts(totalWeight, totalPrice,
        quantity)
    else 0.0
    endif
end

association shippingMethod_taxClass between
    ShippingMethod[*]
    TaxClass[0..1]
end

class ZoneRates < ShippingMethod
end

association zoneRates_items between
    ZoneRates[*]
    ShippingTableItem[*] role items
end

association zoneRates_country between
    ZoneRates[*]
    Country[*]
end

abstract class SpecificZoneMethod < ShippingMethod
end

association specificZoneMethod_taxZone between
    SpecificZoneMethod[*]
    TaxZone[0..1]
end

abstract class HandlingFeeMethod < ShippingMethod
attributes
    handlingFee:Real
end

class FlatRate < SpecificZoneMethod
attributes
    cost:Real
operations
    shippingCosts3(totalWeight:Real, totalPrice:Real, quantity:Integer): Real =
        self.cost
end

class PerItem < SpecificZoneMethod,HandlingFeeMethod
attributes
    cost:Real
operations
    shippingCosts(totalWeight:Real, totalPrice:Real, quantity:Integer): Real =
        self.cost*quantity
end

class TableRate < SpecificZoneMethod,HandlingFeeMethod
attributes
    method:ShippingTableMethod
```
CSTL and its application to the osCommerce case study.
Albert Tort

operations
  shippingCosts3(totalWeight:Real, totalPrice:Real, quantity:Integer): Real =
  if self.method = #weight
    then
      self.items -> select (i | i.number <= (totalWeight*quantity)) ->
        sortedBy(number) ->last().cost
    else
      self.items -> select (i | i.number <= (totalPrice*quantity)) ->
        sortedBy(number) ->last().cost
  endif
end

association tableRate_items between
  TableRate[*]
  ShippingTableItem[*] role items
end

class USPostalService < SpecificZoneMethod,HandlingFeeMethod
attributes
  userID:String
  password:String
  server:USPSServer
operations
  shippingCosts3(totalWeight:Real, totalPrice:Real, quantity:Integer): Real =
    -- we should call USPS service to calculate the shipping costs
end

-- Languages
class Language
attributes
  name:String
  code:String
  image:File
  directory:String
  sortOrder:Integer
  _prova:Integer
End

association language_defaultCurrency between
  Language[*]
  Currency[0..1] role defaultCurrency
end

-- Currencies
class Currency
attributes
  title:String
  code:String
  symbolLeft:String
  symbolRight:String
  decimalPlaces:Integer
  value:Real
  lastUpdate:DateTime
  status:Status
end

-- Location & Taxes
class Country
attributes
  name:String
  isoCode2:String
  isoCode3:String
end

class Zone
attributes
  name:String
  code:String
end
CSTL and its application to the osCommerce case study.
Albert Tort

association country_zone between
  Country [1]
  Zone[*]
end

class TaxZone
attributes
  name:String
  description:String
end

association zone_taxZone between
  Zone[*]
  TaxZone[*]
end

class TaxClass
attributes
  name:String
  description:String
end

association taxClass_product between
  TaxClass[0..1]
  Product[*]
end

association class TaxRate between
  TaxClass[*]
  TaxZone[*]
attributes
  rate:Real
  priority:Integer
  description:String
end

-- STORE ADMINISTRATION
-- Products

class Product
attributes
  status:ProductStatus
  available:Date
  netPrice:Real
  quantityOnHand:Integer
  quantityOrdered:Integer
  modelM:String
  imagePath:String
  added:DateTime
  weight:Real
operations
  specialNetPrice():Real =
    if selfoclIsTypeOf(Special) then
      if selfoclAsType(Special).specialStatus=#enabled
        then selfoclAsType(Special).specialPrice
        else oclEmpty(Set(Real))->any(true)
      endif
    else oclEmpty(Set(Real))->any(true)
    endif

timesViewed():Integer =
  self.productInLanguage.viewed->sum()

grossPrice():Real=
  self.addTaxes(Store.allInstances -> any(true).zone, self.netPrice)

addTaxes(x:z,basePrice:Real):Real=
  let appliedTaxRates:Set(TaxRate) =
    TaxRate.allInstances->select(tr | z.taxZone->includes(tr.taxZone)) -> select (tr
    | tr.taxClass = self.taxClass)
  in
CSTL and its application to the osCommerce case study.
Albert Tort

let priorities:Set(Integer) = 
  if appliedTaxRates->isEmpty() then oclEmpty(Set(Integer))
  else appliedTaxRates -> sortedBy(priority).priority -> asSet()
endif
in
  if priorities -> isEmpty() then basePrice
  else priorities -> iterate (p:Integer; res:Real = basePrice |
    res + (((appliedTaxRates -> select (tr | tr.priority = p).rate |
      -> sum()) / 100))*res)
  endif
end

association product_manufacturer between
  Product[*]
  Manufacturer[0..1]
end

association product_category between
  Product[*]
  Category[*]
end

associationclass ProductInLanguage between
  Product[*]
  Language[*]
attributes
  name:String
  description:String
  url:URL
  viewed:Integer
end

-- Product attributes and options
class Option
end

class Value
end

associationclass Attribute between
  Option[*]
  Value[*]
end

associationclass ProductAttribute between
  Product[*]
  Attribute[*]
attributes
  increment:Real
  sign:Sign
  status:Status
end

class Downloadable < ProductAttribute
attributes
  filename:File
  expiryDays:Integer
  maximumDownloadCount:Integer
end

class StringDT
attributes
  string:String
end

associationclass HasOptionName between
  Option[0..1]
  StringDT[1] role optionName
  Language[*] role optionLanguage
end
associationclass HasValueName between
  Value[0..1] role valueName
  Language[*] role valueLanguage
end

-- Product categories
class Category
  attributes
    imagePath:String
    sortOrder:Integer
    _subcategories:Integer
    _products:Integer
  operations
    subcategories():Integer=self.child->size()
    products():Integer=Category.allInstances
      -> select(c|c.allParents()->includes(self))
      -> union(Set{self}).product->size()
    allParents():Set(Category)=if self.parent.isDefined()
      then self.parent
      -> union(self.parent.allParents())
    else Set{self}
    endif-Set{self}
end

association parent_child between
  Category[0..1] role parent
  Category[*] role child
end

associationclass HasCategoryName between
  Category[0..1] role categoryName
  Language[*]
end

-- Specials
class Special < Product
  attributes
    specialPrice:Real
    expiryDate:Date
    specialLastModified:String
    specialStatus:Status
    dateStatusChanged:DateTime
end

-- Manufacturers
class Manufacturer
  attributes
    name:String
    imagePath:String
    lastModified:DateTime
end

associationclass ManufacturerInLanguage between
  Manufacturer[*]
  Language[*]
attributes
  url:URL
  urlClicked:Integer
  lastClick:DateTime
end

-- Banners
class BannerGroup
  attributes
    name:String
end
CSTL and its application to the osCommerce case study.
Albert Tort

class Banner
attributes
  title: String
  url: URL
  imagePath: String
  html: String
  expires: Date
  scheduled: Date
  statusChanged: DateTime
  status: Status
end

association banner_bannerGroup between
  Banner[*]
  BannerGroup[*]
end

association class BannerHistory between
  Banner[*]
  Date[*]
attributes
  shown: Integer
  clicked: Integer
end

-- Newsletters

class Newsletter
attributes
  title: String
  content: String
  sent: DateTime
  status: NewsletterStatus
end

class ProductNotification < Newsletter
attributes
  global: Boolean
  _notifications: Set(Product)
operations
  notifications(): Set(Product) =
    if self.global then Product.allInstances
    else self.explicitNotifications
    endif
end

association explicitRelatedProduct_explicitNotifications between
  ProductNotification[*] role explicitRelatedProduct
  Product[*] role explicitNotifications
end

-- CUSTOMERS
-- Customers

class Customer
attributes
  gender: Gender
  firstName: String
  lastName: String
  dateOfBirth: Date
  eMailAddress: EMail
  phone: String
  fax: String
  newsletter: Boolean
  password: String
  lastModified: DateTime
  lastLogon: DateTime
  numberOfLogons: Integer
  globalNotifications: Boolean
  status: Status
CSTL and its application to the osCommerce case study.
Albert Tort

operations
  notifications():Set(Product)=
    if self.globalNotifications then Product.allInstances
    else self.explicitNotifications
  endif
end

association explicitNotificationSubscriber_explicitNotifications between
  Customer[*] role explicitNotificationSubscriber
  Product[*] role explicitNotifications
end

class Address
attributes
  gender:Gender
  firstName:String
  lastName:String
  company:String
  street:String
  suburb:String
  postCode:PostalCode
  city:String
  state:String
end

association address_zone between
  Address[*]
  Zone[0..1]
end

association address_country between
  Address[*]
  Country[1]
end

association customer_address between
  Customer[*]
  Address[1..*]
end

association primaryAddressCustomer_primary between
  Customer[*] role primaryAddressCustomer
  Address[1] role primary
end

-- ONLINE CATALOG
-- Reviews
class Review
attributes
  review:String
  rating:Rating
  lastModified:DateTime
  timesRead:Integer
end

association review_language between
  Review[*]
  Language[1]
end

association review_product between
  Review[*]
  Product[1]
end

association review_customer between
  Review[*]
  Customer[1]
end
CSTL and its application to the osCommerce case study.
Albert Tort

-- Shopping carts
class Session
attributes
    sessionID:Integer
    expiry:DateTime
    ipAddress:String
    timeEntry:DateTime
    timeLastClick:DateTime
    lastPageURL:URL
end

association session_currentLanguage between
    Session[*]
    Language[1] role currentLanguage
end

association session_currentCurrency between
    Session[*]
    Currency[1] role currentCurrency
end

association session_customer between
    Session[0..1]
    Customer[0..1]
end

class ShoppingCart
end

class AnonymousShoppingCart < ShoppingCart
end

class CustomerShoppingCart < ShoppingCart
end

association customerShoppingCart_customer between
    CustomerShoppingCart[0..1]
    Customer[1]
end

association shoppingCart_session between
    ShoppingCart[0..1]
    Session[0..1] role sessionOfShoppingCart
end

class ShoppingCartItem
attributes
    quantity:Integer
operations
    price():Real =
        let netPriceWithSpecial:Real =
            if self.product.specialNetPrice().isUndefined() then
                self.product.specialNetPrice()
            else self.product.netPrice
            endif
        in
            if self.attribute -> isEmpty() then netPriceWithSpecial
            else self.attribute.productAttribute -> select (pa | pa.product = self.product) ->
                collect
                    if sign = #plus
                        then increment
                    else (-increment)
                    endif -> sum() + netPriceWithSpecial
                endif
            endif
end

association shoppingCartItem_product between
    ShoppingCartItem[*]
    Product[1]
end
CSTL and its application to the osCommerce case study.
Albert Tort

association shoppingCartItem_attribute between
  ShoppingCartItem[*] Attribute[*]
end

association shoppingCart_shoppingCartItem between
  ShoppingCart[0..1] ShoppingCartItem[1..*] ordered
end

-- Orders
class OrderStatus
  attributes
    status:Status
  end
class Order
  attributes
    delivery:Address
    billing:Address
  operations
    id():Integer =
      if Order.allInstances -> size() = 0 then 0
      else Order.allInstances -> sortedBy(id()) -> last().id() + 1
      endif
    name():String =
      self.customer.firstName
    phone():String =
      self.customer.phone
    eMail():EMail =
      self.customer.eMailAddress
    primary():Address =
      self.customer.primary
    currencyValue():Real =
      self.currency.value
    total():Real =
      let totalWithoutShippingCosts:Real =
        self.orderLine -> collect(finalPrice()*quantity) -> sum()
      in
        let totalWeight:Real =
          self.orderLine -> collect(product.weight*quantity) -> sum()
      in
        let quantity:Integer =
          self.orderLine.quantity -> sum()
      in
        let handlingFee:Real =
          if self.shippingMethod.oclIsKindOf(HandlingFeeMethod)
            then
              self.shippingMethod.oclAsType(HandlingFeeMethod).handlingFee
            else 0.0
          endif
        in
          let totalWeightIncreased:Real =
            if totalWeight* (ShippingAndPackaging.allInstances
               ->any(true).percentageIncreaseForLargerPackages/100) >
              (ShippingAndPackaging.allInstances
                ->any(true)).typicalPackageTareWeight
              then
                totalWeight * (1 +totalWeight*
                  ((ShippingAndPackaging.allInstances
                    ->any(true)).percentageIncreaseForLargerPackages/100))
              else
                totalWeight + (ShippingAndPackaging.allInstances
                  ->any(true)).typicalPackageTareWeight
              endif
            in
              totalWithoutShippingCosts
              +
              self.shippingMethod.shippingCosts(totalWeightIncreased,
                totalWithoutShippingCosts, quantity)
              + handlingFee
          end
      end
    end
CSTL and its application to the osCommerce case study.
Albert Tort

association order_customer between
  Order[*]
  Customer[1]
end

association order_shippingMethod between
  Order[*]
  ShippingMethod[1]
end

association order_paymentMethod between
  Order[*]
  PaymentMethod[1]
end

association order_currency between
  Order[*]
  Currency[1]
end

associationclass OrderStatusChange between
  Order[*]
  OrderStatus[1..*] ordered
attributes
  comments:String
end

associationclass OrderStatusInLanguage between
  OrderStatus[*]
  Language[*]
attributes
  name:String
end

class OrderLine
attributes
  quantity:Integer
operations
  name():String=
    self.product.productInLanguage
    ->select(pil | pil.language = Store.allInstances ->
      any(true).defaultLanguage).name->any(true)
  modelM():String=
    self.product.modelM
  basePrice():Real=
    if self.product.specialNetPrice().isDefined()
      then self.product.specialNetPrice()
    else self.product.netPrice
    endif
  price():Real=
    if self.orderLineAttribute -> isEmpty() then self.basePrice()
    else
      self.orderLineAttribute -> collect
        (if sign() = + then increment()
        else (-increment())
        endif) -> sum() + self.basePrice()
    endif
  finalPrice():Real=
    if self.order.billing.zone -> notEmpty() then
      self.product.addTaxes(self.order.billing.zone, self.price())
    else self.price()
    endif
end
CSTL and its application to the osCommerce case study.
Albert Tort

association order_orderLine between
  Order[1]
  OrderLine[1..*] ordered
end

association orderLine_product between
  OrderLine[*]
  Product[1]
end

class OrderLineAttribute
  operations
    option():String=
      self.attribute.option.hasOptionName
      -> select (hon | hon.optionLanguage = Store.allInstances
      -> any(true).defaultLanguage).optionName->any(true).string
    value():String=
      self.attribute.value.hasValueName
      -> select (hvn | hvn.valueLanguage = Store.allInstances
      -> any(true).defaultLanguage).valueName->any(true).string
    increment():Real=
      self.attribute.productAttribute
      -> select (pa | pa.product = self.orderLine.product).increment->any(true)
    sign():Sign=
      self.attribute.productAttribute
      -> select (pa | pa.product = self.orderLine.product).sign->any(true)
  end

class OrderDownload < OrderLineAttribute
  attributes
    downloadCount:Integer
  end

association orderLineAttribute_attribute between
  OrderLineAttribute[*]
  Attribute[1]
end

association orderLine_orderLineAttribute between
  OrderLine[1]
  OrderLineAttribute[*] ordered
end

-- BEHAVIOURAL SCHEMA

class Time
end

abstract class Event
  attributes
    time:DateTime
  operations
    effect()
end

abstract class DomainEvent < Event
end

abstract class ActionRequest < Event
end

abstract class Query < ActionRequest
end

abstract class SessionEvent
end

association sessionEvent_session between
CSTL and its application to the osCommerce case study.
Albert Tort

```ruby
SessionEvent[*]
Session[1]
end

class AddProductToShoppingCart < SessionEvent, DomainEvent
attributes
  quantity:Integer
operations
  effect()
end

association addProductToShoppingCart_attribute between
  AddProductToShoppingCart[*]
    Attribute[*]
end

association addProductToShoppingCart_product between
  AddProductToShoppingCart[*]
    Product[1]
end

class AddressBookEntriesMaximumChange < DomainEvent
attributes
  newMaximum:Integer
operations
  effect()
end

class AllowCheckoutStockConfigurationChange < DomainEvent
attributes
  newValue:Boolean
operations
  effect()
end

abstract class StoreEvent
operations
  myStore():Store=Store.allInstances->any(true)
end

class AllowGuestToTellAFriendChange < DomainEvent, StoreEvent
attributes
  newAllowGuestToTellAFriend:Boolean
operations
  effect()
end

abstract class ExistingProductAttributeEvent
end

association existingProductAttributeEvent_productAttribute between
  ExistingProductAttributeEvent[*]
    ProductAttribute[0..1]
end

class AttributeChange < DomainEvent, ExistingProductAttributeEvent
operations
  effect()
end

association attributeChange_Value between
  AttributeChange[*]
    Value[1] role newValue
end

association attributeChange_Option between
  AttributeChange[*]
    Option[1] role newOption
end

abstract class ExistingOrderEvent
end
```
CSTL and its application to the osCommerce case study.
Albert Tort

association existingOrderEvent_Order between 
   ExistingOrderEvent[*] 
   Order[1] 
end

class CancelOrder < DomainEvent,ExistingOrderEvent 
   operations 
      effect() 
   end

class CheckLevelStockConfigurationChange < DomainEvent 
   attributes 
      newValue:Boolean 
   operations 
      effect() 
   end

class CityMinimumChange < DomainEvent 
   attributes 
      newMinimum:Integer 
   operations 
      effect() 
   end

abstract class ExistingBannerEvent 
end

association existingBannerEvent_banner between 
   ExistingBannerEvent[*] 
   Banner[0..1] 
end

class ClickBanner < DomainEvent, ExistingBannerEvent 
   operations 
      effect() 
   end

abstract class ExistingManufacturerEvent 
end

association existingManufacturerEvent_banner between 
   ExistingManufacturerEvent[*] 
   Manufacturer[0..1] 
end

class ClickManufacturer < DomainEvent, ExistingManufacturerEvent 
   operations 
      effect() 
   end

association clickManufacturer_language between 
   ClickManufacturer[*] 
   Language[1] 
end

class CompanyCustomerDetailChange < DomainEvent 
   attributes 
      newValue:Boolean 
   operations 
      effect() 
   end

class CompanyNameMinimumChange < DomainEvent 
   attributes 
      newMinimum:Integer 
   operations 
      effect() 
   end

CSTL and its application to the osCommerce case study.
Albert Tort

class CountryChange < DomainEvent, StoreEvent
operations
  effect()
end

association countryChange_country between
  CountryChange[*]
end

class CountryShippingConfigurationChange < DomainEvent
operations
  effect()
end

association countryShippingConfigurationChange_country between
  CountryShippingConfigurationChange[*]
end

class CreditCardNumberMinimumChange < DomainEvent
attributes
  newMinimum: Integer
operations
  effect()
end

class CreditCardOwnerNameMinimumChange < DomainEvent
attributes
  newMinimum: Integer
operations
  effect()
end

abstract class ExistingCurrencyEvent
end

association existingCurrencyEvent_currency between
  ExistingCurrencyEvent[*]
  Currency[0..1]
end

class CurrencyStatusChange < DomainEvent, ExistingCurrencyEvent
attributes
  newStatus: Status
operations
  effect()
end

abstract class ExistingCustomerEvent
end

association existingCustomerEvent_customer between
  ExistingCustomerEvent[*]
  Customer[0..1]
end

class CustomerStatusChange < DomainEvent, ExistingCustomerEvent
attributes
  newStatus: Status
operations
  effect()
end

class DateOfBirthCustomerDetailChange < DomainEvent
attributes
  newValue: Boolean
CSTL and its application to the osCommerce case study.
Albert Tort

operations
effect()
end
class DateOfBirthMinimumChange < DomainEvent
attributes
  newMinimum:Integer
operations
  effect()
end
class DaysExpiryDelayDownloadConfigurationChange < DomainEvent
attributes
  newValue:Integer
operations
  effect()
end
class DefaultSearchOperatorChange < DomainEvent, StoreEvent
attributes
  newDefaultSearchOperator:Operator
operations
  effect()
end
class DeleteBanner < DomainEvent, ExistingBannerEvent
operations
  effect()
end
abstract class ExistingBannerGroupEvent
end
association existingBannerGroupEvent_bannerGroup between
  ExistingBannerGroupEvent[*]
  BannerGroup[0..1]
end
class DeleteBannerGroup < DomainEvent, ExistingBannerGroupEvent
operations
  effect()
end
abstract class ExistingCategoryEvent
end
association existingCategoryEvent_category between
  ExistingCategoryEvent[*]
  Category[0..1]
end
class DeleteCategory < DomainEvent, ExistingCategoryEvent
operations
  effect()
  allChildren(cat:Category):Set(Category) = if cat.child->isEmpty()
    then oclEmpty(Set(Category))
    else cat.child->iterate(c;
      a:Set(Category)=cat.child | a->union(self.allChildren(c)))
    endif
end
abstract class ExistingCountryEvent
end
association existingCountryEvent_country between
  ExistingCountryEvent[*]
  Country[0..1]
end
CSTL and its application to the osCommerce case study.
Albert Tort

class DeleteCountry < DomainEvent, ExistingCountryEvent
  operations
effect()
end

class DeleteCurrency < DomainEvent, ExistingCurrencyEvent
  operations
effect()
end

class DeleteCustomer < DomainEvent, ExistingCustomerEvent
  operations
effect()
end

abstract class ExistingAddressEvent
end

association existingAddressEvent_address between
  ExistingAddressEvent[*]
  Address[1]
end

class DeleteCustomerAddress < DomainEvent, ExistingCustomerEvent, ExistingAddressEvent
  operations
effect()
end

abstract class ExistingLanguageEvent
end

association existingLanguageEvent_language between
  ExistingLanguageEvent[*]
  Language[0..1]
end

class DeleteLanguage < DomainEvent, ExistingLanguageEvent
  operations
effect()
end

class DeleteManufacturer < DomainEvent, ExistingManufacturerEvent
  attributes
    deleteProds:Boolean
  operations
    effect()
end

abstract class ExistingNewsletterEvent
end

association existingNewsletterEvent_newsletter between
  ExistingNewsletterEvent[*]
  Newsletter[0..1]
end

class DeleteNewsletter < DomainEvent, ExistingNewsletterEvent
  operations
effect()
end

abstract class ExistingOrderStatusEvent
end

association existingOrderStatusEvent_orderStatus between
  ExistingOrderStatusEvent[*]
  OrderStatus[0..1]
end
CSTL and its application to the osCommerce case study.
Albert Tort

class DeleteOrderStatus < DomainEvent, ExistingOrderStatusEvent
operations
effect()
end

abstract class ExistingProductEvent
end

association existingProductEvent_product between
   ExistingProductEvent[*]
   Product[0..1]
end

class DeleteProduct < DomainEvent, ExistingProductEvent
operations
effect()
end

class DeleteProductAttribute < DomainEvent, ExistingProductAttributeEvent
operations
effect()
end

class DeleteProductNotificationSubscription < DomainEvent, ExistingCustomerEvent
operations
effect()
end

association deleteProductNotificationSubscription_product between
   DeleteProductNotificationSubscription[*]
   Product[1] role deletedSubscribedProduct
end

abstract class ExistingOptionEvent
end

association existingOptionEvent_option between
   ExistingOptionEvent[*]
   Option[0..1]
end

class DeleteProductOption < DomainEvent, ExistingOptionEvent
operations
effect()
end

abstract class ExistingValueEvent
end

association existingValueEvent_option between
   ExistingValueEvent[*]
   Value[0..1]
end

class DeleteProductOptionValue < DomainEvent, ExistingValueEvent
operations
effect()
end

abstract class ExistingReviewEvent
end

association existingReviewEvent_review between
   ExistingReviewEvent[*]
   Review[0..1]
end
CSTL and its application to the osCommerce case study.
Albert Tort

class DeleteReview < DomainEvent, ExistingReviewEvent
operations
effect()
end

abstract class ExistingSessionEvent
end

association existingSessionEvent_Session between
   ExistingSessionEvent[*]
   Session[0..1]
end
class DeleteSession < DomainEvent, ExistingSessionEvent
operations
effect()
end

abstract class ExistingSpecialEvent
end

association existingSpecialEvent_special between
   ExistingSpecialEvent[*]
   Special[0..1]
end
class DeleteSpecial < DomainEvent, ExistingSpecialEvent
operations
effect()
end

abstract class ExistingTaxClassEvent
end

association existingTaxClassEvent_taxClass between
   ExistingTaxClassEvent[*]
   TaxClass[0..1]
end
class DeleteTaxClass < DomainEvent, ExistingTaxClassEvent
operations
effect()
end

abstract class ExistingTaxRateEvent
end

association existingTaxRateEvent_taxRate between
   ExistingTaxRateEvent[*]
   TaxRate[0..1]
end
class DeleteTaxRate < DomainEvent, ExistingTaxRateEvent
operations
effect()
end

abstract class ExistingTaxZoneEvent
end

association existingTaxZoneEvent_taxZone between
   ExistingTaxZoneEvent[*]
   TaxZone[0..1]
end
class DeleteTaxZone < DomainEvent, ExistingTaxZoneEvent
operations
effect()
end

abstract class ExistingZoneEvent
end
CSTL and its application to the osCommerce case study.
Albert Tort

association existingZoneEvent_zone between
   ExistingZoneEvent[*]
   Zone[0..1]
end

class DeleteZone < DomainEvent, ExistingZoneEvent
   operations
      effect()
end

class DisplayCartAfterAddingProductChange < DomainEvent, StoreEvent
   attributes
      newDisplayCartAfterAddingProduct:Boolean
   operations
      effect()
end

class DisplayPricesWithTaxChange < DomainEvent, StoreEvent
   attributes
      newDisplayPricesWithTax:Boolean
   operations
      effect()
end

abstract class EditPaymentMethodEvent
   attributes
      status:Status
end

association editPaymentMethodEvent_taxZone between
   EditPaymentMethodEvent[*]
   TaxZone[0..1]
end

association editPaymentMethodEvent_orderStatus between
   EditPaymentMethodEvent[*]
   OrderStatus[0..1]
end

class EditAuthorizeNetPaymentMethod < DomainEvent, EditPaymentMethodEvent
   attributes
      newUsername:String
      newKey:String
      newMode:TransactionMode
      newMethod:TransactionMethod
      newNotification:Boolean
   operations
      effect()
end

class EditBanner < DomainEvent, ExistingBannerEvent
   attributes
      newTitle:String
      newUrl:URL
      newImagePath:String
      newHtml:String
      newExpires:Date
      newScheduled:Date
      newStatus:Status
   operations
      effect()
end

association editBanner_bannerGroup between
   EditBanner[*] — BannerGroup[1] role newBannerGroup
end

class EditBannerGroup < DomainEvent, ExistingBannerGroupEvent
   attributes
      newName:String
CSTL and its application to the osCommerce case study.
Albert Tort

```ruby
operations
  effect()
end

class EditCashOnDeliveryPaymentMethod < DomainEvent, EditPaymentMethodEvent
  operations
    effect()
end

abstract class CategoryNameEvent
end

association class HasNewName between
  CategoryNameEvent[*]
  Language[*] role languageOfCategory
  StringDT[1] role name
end

class EditCategory < DomainEvent, ExistingCategoryEvent, CategoryNameEvent
  attributes
    imagePath:String
    sortOrder:Integer
  operations
    effect()
end

association editCategory_category between
  EditCategory[*]
  Category[0..1] role newParent
end

class EditCheckMoneyPaymentMethod < DomainEvent, EditPaymentMethodEvent
  attributes
    newMakePayableTo:String
  operations
    effect()
end

class EditCountry < DomainEvent, ExistingCountryEvent
  attributes
    newName:String
    newIsoCode2:String
    newIsoCode3:String
  operations
    effect()
end

class EditCreditCardPaymentMethod < DomainEvent, EditPaymentMethodEvent
  attributes
    newSplitCreditCardToMail:EMail
  operations
    effect()
end

class EditCurrency < DomainEvent, ExistingCurrencyEvent
  attributes
    newTitle:String
    newCode:String
    newSymbolLeft:String
    newSymbolRight:String
    newDecimalPlaces:Integer
    newValue:Real
  operations
    effect()
end

class EditCustomer < DomainEvent, ExistingCustomerEvent
  attributes
    newGender:Gender
    newFirstName:String
    newLastName:String
    newDateOfBirth:Date
```
CSTL and its application to the osCommerce case study.
Albert Tort

newEMailAddress:EMail
newPhone:PhoneNumber
newFax:PhoneNumber
newNewsletter:Boolean
newPassword:Password
newGlobalNotifications:Boolean
operations
effect() end

class EditCustomerAddress < DomainEvent, ExistingCustomerEvent, ExistingAddressEvent
attributes
newAddress:Address
operations
effect() end

class EditCustomerDetails < DomainEvent, ExistingCustomerEvent
attributes
newGender: Gender
newFirstName: String
newLastName: String
newDateOfBirth: Date
newEMailAddress: Email
newPhone: PhoneNumber
newFax: PhoneNumber
newNewsletter: Boolean
operations
effect() end

abstract class ExistingDownloadableEvent end

association existingDownloadableEvent_Downloadable between
ExistingDownloadableEvent[*]
Downloadable[1]
end

class EditDownloadableAttribute < DomainEvent, ExistingDownloadableEvent
attributes
newFilename: File
newExpiryDays: Integer
newMaximumDownloadCount: Integer
operations
effect() end

abstract class ShippingMethodEvent
attributes
status: Status
end

association ShippingMethodEvent_taxClass between
ShippingMethodEvent[*]
TaxClass[0..1]
end

abstract class SpecificZoneShippingMethodEvent < ShippingMethodEvent end

association SpecificZoneShippingMethodEvent_taxZone between
SpecificZoneShippingMethodEvent[*]
TaxZone[0..1]
end

class EditFlatRateShippingMethod < DomainEvent, SpecificZoneShippingMethodEvent
attributes
newCost: Real
operations
effect() end
CSTL and its application to the osCommerce case study.
Albert Tort

class EditGlobalNotifications < DomainEvent, ExistingCustomerEvent
attributes
  newGlobalNotifications:Boolean
operations
  effect()
end

class EditIPaymentPaymentMethod < DomainEvent, EditPaymentMethodEvent
attributes
  newAccount:Integer
  newUser:String
  newPassword:String
operations
  effect()
end

class EditLanguage < DomainEvent, ExistingLanguageEvent
attributes
  newName:String
  newCode:String
operations
  effect()
end
association editLanguage_currency between
  EditLanguage[*]
  Currency[0..1] role newDefaultCurrency
end
abstract class ManufacturerURLEvent
end
associationclass HasURL between
  ManufacturerURLEvent[*]
  Language[*] role languageOfURL
  URL[1] role url
end

class EditManufacturer < DomainEvent, ExistingManufacturerEvent, ManufacturerURLEvent
attributes
  imagePath:String
  name:String
operations
  effect()
end

class EditNewsletter < DomainEvent, ExistingNewsletterEvent
attributes
  newTitle:String
  newContent:String
operations
  effect()
end

class EditNochexPaymentMethod < DomainEvent, EditPaymentMethodEvent
attributes
  newEMail:EMail
operations
  effect()
end
abstract class OrderStatusNameEvent
end
associationclass HasOrderStatusName between
  OrderStatusNameEvent[*]
  Language[*] role languageOfOrderStatus
  StringDT[1] role orderStatusName
end

class EditOrderStatus < DomainEvent, ExistingOrderStatusEvent, OrderStatusNameEvent
CSTL and its application to the osCommerce case study.
Albert Tort

```ruby
operations
effect()
end

class EditPayPalPaymentMethod < DomainEvent, EditPaymentMethodEvent
attributes
newEMail:EMail
operations
effect()
end

association editPayPalMethod_currency between
   EditPayPalPaymentMethod[*]
   Currency[0..1]
end

abstract class HandlingFeeMethodEvent
attributes
   handlingFee:Real
end

class EditPerItemShippingMethod < DomainEvent, SpecificZoneShippingMethodEvent, HandlingFeeMethodEvent
attributes
   newCost:Real
operations
effect()
end

abstract class ProductNameEvent
end

association class HasNewProductName between
   ProductNameEvent[*]
   Language[*] role languageOfProduct
   StringDT[*] role nameOfProduct
end

class EditProduct < DomainEvent, ExistingProductEvent, ProductNameEvent
attributes
   status:ProductStatus
   available:Date
   netPrice:Real
   quantityOnHand:Integer
   modelM:String
   imagePath:String
   weight:Real
operations
effect()
end

association editProduct_manufacturer between
   EditProduct[*]
   Manufacturer[0..1]
end

association editProduct_category between
   EditProduct[*]
   Category[*]
end

association editProduct_taxClass between
   EditProduct[*]
   TaxClass[0..1]
end

class EditProductNotification < DomainEvent
attributes
   newGlobal:Boolean
operations
effect()
end
```
CSTL and its application to the osCommerce case study.
Albert Tort

association editProductNotification_product between
  EditProductNotification[*]
  Product[*] role newExplicitNotifications
end

association editProductNotification_productNotification between
  EditProductNotification[*]
  ProductNotification[1]
end

abstract class ProductOptionNameEvent
end

associationclass HasNewOptionName between
  ProductOptionNameEvent[*]
  Language[*] role languageOfOption
  StringDT[1] role nameOfOption
end

class EditProductOption < DomainEvent, ExistingOptionEvent, ProductOptionNameEvent
operations
  effect()
end

abstract class ProductValueNameEvent
end

associationclass HasNewValueName between
  ProductValueNameEvent[*]
  Language[*] role languageOfValue
  StringDT[1] role nameOfValue
end

class EditProductOptionValue < DomainEvent, ExistingValueEvent, ProductValueNameEvent
operations
  effect()
end

association editProductOptionValue_Option between
  EditProductOptionValue[*]
  Option[1..*]
end

class EditPSiGatePaymentMethod < DomainEvent, EditPaymentMethodEvent
attributes
  newMerchantID:String
  newMode:PSiGateMode
  newType:PSiGateType
  newCreditCardCollection:PSiGateCollection
operations
  effect()
end

association editPSiGatePaymentMethod_currency between
  EditPSiGatePaymentMethod[*]
  Currency[0..1]
end

class EditReview < DomainEvent, ExistingReviewEvent
attributes
  newReview:String
  newRating:Rating
operations
  effect()
end

association editReview_Language between
  EditReview[*]
  Language[1] role newLanguage
end
CSTL and its application to the osCommerce case study.
Albert Tort

association editReview_Product between
    EditReview[*]
    Product[1] role newProduct
end

association editReview_Customer between
    EditReview[*]
    Customer[1] role newCustomer
end

class EditSECPaymentMethod < DomainEvent, EditPaymentMethodEvent
    attributes
        newMerchantID:String
        newMode:SECPayMode
    operations
        effect()
end

association editSECPaymentMethod_currency between
    EditSECPaymentMethod[*]
    Currency[0..1]
end

class EditSpecial < DomainEvent, ExistingSpecialEvent
    attributes
        newSpecialPrice:Real
        newExpiryDate:Date
        newStatus:Status
    operations
        effect()
end

class EditTableRateShippingMethod < DomainEvent, SpecificZoneShippingMethodEvent, HandlingFeeMethodEvent
    attributes
        newMethod:ShippingTableMethod
    operations
        effect()
end

association editTableRateShippingMethod_newItems between
    EditTableRateShippingMethod[*]
    ShippingTableItem[*] role newItems
end

class EditTaxClass < DomainEvent, ExistingTaxClassEvent
    attributes
        newName:String
        newDescription:String
    operations
        effect()
end

class EditTaxRate < DomainEvent, ExistingTaxRateEvent
    attributes
        newRate:Integer
        newPriority:Integer
        newDescription:String
    operations
        effect()
end

association editTaxRate_taxZone between
    EditTaxRate[*]
    TaxZone[1] role newTaxZone
end

association editTaxRate_taxClass between
    EditTaxRate[*]
    TaxClass[1] role newTaxClass
end
CSTL and its application to the osCommerce case study.
Albert Tort

```ruby
class EditTaxZone < DomainEvent, ExistingTaxZoneEvent
  attributes
    newName:String
    newDescription:String
  operations
    effect()
end

association editTaxZone_mewZones between
  EditTaxZone[*]
  Zone[*] role newZones
end

class EditTwoCheckOutPaymentMethod < DomainEvent, EditPaymentMethodEvent
  attributes
    newLogin:String
    newMode:TransactionMode
    newMerchantNotification:Boolean
  operations
    effect()
end

class EditUSPostalServiceShippingMethod < DomainEvent, SpecificZoneShippingMethodEvent, HandlingFeeMethodEvent
  attributes
    newUserID:String
    newPassword:String
    newServer:USPSServer
  operations
    effect()
end

class EditZone < DomainEvent, ExistingZoneEvent
  attributes
    newName:String
    newCode:String
  operations
    effect()
end

class EditZoneRatesShippingMethod < DomainEvent, ShippingMethodEvent
  operations
    effect()
end

association editZoneRatesShippingMethod_country between
  EditZoneRatesShippingMethod[*]
  Country[*]
end

association editZoneRatesShippingMethod_mewItems between
  EditZoneRatesShippingMethod[*]
  ShippingTableItem[*] role newItems
end

class EMailAddressChange < DomainEvent, StoreEvent
  attributes
    newEmailAddress:EMail
  operations
    effect()
end

class EMailAddressMinimumChange < DomainEvent
  attributes
    newMinimum:Integer
  operations
    effect()
end
```
CSTL and its application to the osCommerce case study.
Albert Tort

class EMailFromChange < DomainEvent, StoreEvent
attributes
  newEmailFrom: EMail
operations
  effect()
end

class EnableDownloadConfigurationChange < DomainEvent
attributes
  newValue: Boolean
operations
  effect()
end

class ExpectedSortFieldChange < DomainEvent, StoreEvent
attributes
  newExpectedSortField: SortField
operations
  effect()
end

class ExpectedSortOrderChange < DomainEvent, StoreEvent
attributes
  newExpectedSortOrder: SortOrder
operations
  effect()
end

class FirstNameMinimumChange < DomainEvent
attributes
  newMinimum: Integer
operations
  effect()
end

class GenderCustomerDetailChange < DomainEvent
attributes
  newValue: Boolean
operations
  effect()
end

class IncrementAndSignAttributeChange < DomainEvent, ExistingProductAttributeEvent
attributes
  newIncrement: Real
  newSign: Sign
operations
  effect()
end

class InstallAuthorizeNetPaymentMethod < DomainEvent
operations
  effect()
end

class InstallCashOnDeliveryPaymentMethod < DomainEvent
operations
  effect()
end

class InstallCheckMoneyPaymentMethod < DomainEvent
operations
  effect()
end

class InstallCreditCardPaymentMethod < DomainEvent
operations
  effect()
end
CSTL and its application to the osCommerce case study.
Albert Tort

class InstallFlatRateShippingMethod < DomainEvent
operations
  effect()
end

class InstallIPaymentPaymentMethod < DomainEvent
operations
  effect()
end

class InstallNochexPaymentMethod < DomainEvent
operations
  effect()
end

class InstallPayPalPaymentMethod < DomainEvent
operations
  effect()
end

class InstallPerItemShippingMethod < DomainEvent
operations
  effect()
end

class InstallPSiGatePaymentMethod < DomainEvent
operations
  effect()
end

class InstallSECPaymentMethod < DomainEvent
operations
  effect()
end

class InstallTableRateShippingMethod < DomainEvent
operations
  effect()
end

class InstallTwoCheckOutPaymentMethod < DomainEvent
operations
  effect()
end

class InstallUSPostalServiceShippingMethod < DomainEvent
operations
  effect()
end

class InstallZoneRatesShippingMethod < DomainEvent
operations
  effect()
end

class LastNameMinimumChange < DomainEvent
attributes
  newMinimum:Integer
operations
  effect()
end

class LinkProduct < DomainEvent, ExistingProductEvent
operations
  effect()
end

association linkProduct_category between
  LinkProduct[*] Category[1] role newCategory
end
CSTL and its application to the osCommerce case study.
Albert Tort

class LockNewsletter < DomainEvent, ExistingNewsletterEvent
operations
  effect()
end

class LogIn < DomainEvent, ExistingCustomerEvent
operations
  effect()
end

association logIn_session between
  LogIn[*]
  Session[0..1]
end

class LogOut < DomainEvent, ExistingCustomerEvent, ExistingSessionEvent
operations
  effect()
end

class NameChange < DomainEvent, StoreEvent
attributes
  newName:String
operations
  effect()
end

class MaximumNumberDownloadConfigurationChange < DomainEvent
attributes
  newMaximum:Integer
operations
  effect()
end

class MaximumPackageWeightShippingConfigurationChange < DomainEvent
attributes
  newMaximum:Integer
operations
  effect()
end

class MoveCategory < DomainEvent, ExistingCategoryEvent
operations
  effect()
end

association moveCategory_newParent between
  MoveCategory[*]
  Category[0..1] role newParent
end

class MoveProduct < DomainEvent, ExistingProductEvent
operations
  effect()
end

association moveProduct_oldCategory between
  MoveProduct[*]
  Category[1] role oldCategory
association moveProduct_newCategory between
  MoveProduct[*]
  Category[1] role newCategory
end

class NewBanner < DomainEvent
attributes
  title:String
  url:URL
  imagePath:String
  html:

CSTL and its application to the osCommerce case study.
Albert Tort

```ruby
class NewBannerGroup < DomainEvent
  attributes
    name:String
  operations
    effect()
end

association newBanner_bannerGroup between
  NewBanner[*]
  BannerGroup[1];
end

class NewCategory < DomainEvent, CategoryNameEvent
  attributes
    imagePath:String
    sortOrder:Integer
  operations
    effect()
end

association newCategory_category between
  NewCategory[*]
  Category[0..1] role parent
end

class NewCountry < DomainEvent
  attributes
    name:String
    isoCode2:String
    isoCode3:String
  operations
    effect()
end

class NewCurrency < DomainEvent
  attributes
    title:String
    code:String
    symbolLeft:String
    symbolRight:String
    decimalPlaces:Integer
    value:Real
  operations
    effect()
end

class NewCustomer < DomainEvent
  attributes
    dateOfBirth:Date
    eMailAddress:EMail
    phone:String
    fax:String
    newsletter:Boolean
    password:String
    passwordConfirmation:String
    primary:Address
    customerCreated:Customer
  operations
    effect()
end

class NewCustomerAddress < DomainEvent, ExistingCustomerEvent
  attributes
    gender:Gender
    firstName:String
    lastName:String
```
CSTL and its application to the osCommerce case study.
Albert Tort

```java
class NewDownloadableProductAttribute < DomainEvent, ExistingProductEvent
attributes
    increment:Real
    sign:Sign
    filename:File
    expiryDays:Integer
    maximumDownloadCount:Integer
operations
effect()
end

association newDownloadableProductAttribute_option between
    NewDownloadableProductAttribute[*]
        Option[1]
end

association newDownloadableProductAttribute_value between
    NewDownloadableProductAttribute[*]
        Value[1]
end

class NewManufacturer < DomainEvent, ManufacturerURLEvent
attributes
    imagePath:String
    name:String
operations
effect()
end

class NewLanguage < DomainEvent
attributes
    newName:String
    newCode:String
operations
effect()
end

association NewLanguage_currency between
    NewLanguage[*]
        Currency[0..1] role defaultCurrency
end
```
CSTL and its application to the osCommerce case study.
Albert Tort

class NewOrderStatus < DomainEvent, OrderStatusNameEvent
attributes
  name:String
  createdOrderStatus:OrderStatus;
operations
  effect();
end

class NewProduct < DomainEvent, ProductNameEvent
attributes
  status:ProductStatus
  available:Date
  netPrice:Real
  quantityOnHand:Integer
  modelM:String
  imagePath:String
  weight:Real
operations
  effect();
end
association newProduct_manufacturer between
  NewProduct[*]
  Manufacturer[0..1]
end
association newProduct_category between
  NewProduct[*]
  Category[*]
end
association newProduct_taxClass between
  NewProduct[*]
  TaxClass[0..1]
end

class NewProductAttribute < DomainEvent, ExistingProductEvent
attributes
  increment:Real
  sign:Sign
operations
  effect();
end
association newProductAttribute_option between
  NewProductAttribute[*]
  Option[1]
end
association newProductAttribute_value between
  NewProductAttribute[*]
  Value[1]
end

class NewProductNotification < DomainEvent
attributes
  title:String
  content:String
  global:Boolean
operations
  effect();
end
association newProductNotification_product between
  NewProductNotification[*]
  Product[*] role explicitNotifications
end

class NewProductNotificationSubscription < DomainEvent, ExistingCustomerEvent
operations
  effect();
end

213
CSTL and its application to the osCommerce case study.
Albert Tort

association newProductNotificationSubscription_product between
    NewProductNotificationSubscription[*]
    Product[1] role newSubscribedProduct
end

class NewProductOption < DomainEvent, ProductOptionNameEvent
operations
effect()
end

class NewProductOptionValue < DomainEvent, ProductValueNameEvent
operations
effect()
end

association newProductOptionValue_option between
    NewProductOptionValue[*]
    Option[1..*] role option
end

class NewReview < DomainEvent
attributes
    review:String
    rating:Rating
    createdReview:Review
operations
effect()
end

association newReview_language between
    NewReview[*]
    Language[1] role language
end

association newReview_product between
    NewReview[*]
    Product[1] role product
end

association newReview_customer between
    NewReview[*]
    Customer[1] role customer
end

class NewSession < DomainEvent
attributes
    createdSession:Session
operations
effect()
end

association newSession_currentCurrency between
    NewSession[*]
    Currency[1] role currentCurrency
end

association newSession_currentLanguage between
    NewSession[*]
    Language[1] role currentLanguage
end

class NewSpecial < DomainEvent
attributes
    specialPrice:Real
    expiryDate:Date
    status:Status
operations
effect()
end
CSTL and its application to the osCommerce case study.
Albert Tort

association newSpecial_product between
    NewSpecial[*]
    Product[0..1]
end

class NewTaxClass < DomainEvent
attributes
    name:String
    description:String
operations
    effect()
end
class NewTaxRate < DomainEvent
attributes
    rate:Integer
    priority:Integer
    description:String
operations
    effect()
end
association newTaxRate_taxZone between
    NewTaxRate[*]
    TaxZone[1]
end
association newTaxRate_taxClass between
    NewTaxRate[*]
    TaxClass[1]
end

class NewTaxZone < DomainEvent
attributes
    name:String
    description:String
operations
    effect()
end
association newTaxZone_mewZones between
    NewTaxZone[*]
    Zone[*]
end
class NewZone < DomainEvent
attributes
    name:String
    code:String
operations
    effect()
end
association newZone_country between
    NewZone[*]
    Country[0..1]
end
class OrderConfirmation < DomainEvent
attributes
    delivery:Address
    billing:Address
    creditCardType:String
    creditCardOwner:String
    creditCardNumber:String
    creditCardExpires:Date
    comments:String
operations
    effect()
end
CSTL and its application to the osCommerce case study.
Albert Tort

association orderConfirmation_customerShoppingCart between
OrderConfirmation[*]
   CustomerShoppingCart[0..1] role shoppingCart
end

association orderConfirmation_shippingMethod between
OrderConfirmation[*]
   ShippingMethod[1]
end

association orderConfirmation_paymentMethod between
OrderConfirmation[*]
   PaymentMethod[1]
end

association orderConfirmation_currency between
OrderConfirmation[*]
   Currency[1]
end

class OwnerChange < DomainEvent, StoreEvent
attributes
   newOwner:String
operations
   effect()
end

class PasswordChange < DomainEvent, ExistingCustomerEvent
attributes
   oldPassword:String
   newPassword:String
operations
   effect()
end

class PasswordMinimumChange < DomainEvent
attributes
   newMinimum:Integer
operations
   effect()
end

class PercentageIncreaseForLargerPackagesShippingConfigurationChange < DomainEvent
attributes
   newPercentage:Real
operations
   effect()
end

class PostCodeMinimumChange < DomainEvent
attributes
   newMinimum:Integer
operations
   effect()
end

class PostCodeShippingConfigurationChange < DomainEvent
attributes
   newPostCode:PostalCode
operations
   effect()
end

class PrimaryCustomerAddressChange < DomainEvent, ExistingAddressEvent, ExistingCustomerEvent
operations
   effect()
end
CSTL and its application to the osCommerce case study.
Albert Tort

class ProductAttributeStatusChange < DomainEvent, ExistingProductAttributeEvent
  attributes
    newStatus:Status
  operations
    effect()
end

class ProductDownload < DomainEvent, ExistingCustomerEvent, ExistingProductEvent
  operations
    effect()
end

association productDownload_downloadable between
  ProductDownload[*]
  Downloadable[1]
end

class ProductOptionAttributeChange < DomainEvent, ExistingProductAttributeEvent
  operations
    effect()
end

association productOptionAttributeChange_option between
  ProductOptionAttributeChange[*]
  Option[1]
end

class ProductValueAttributeChange < DomainEvent, ExistingProductAttributeEvent
  operations
    effect()
end

association productValueAttributeChange_value between
  ProductValueAttributeChange[*]
  Value[1]
end

class ProductStatusChange < DomainEvent, ExistingProductEvent
  attributes
    newStatus:ProductStatus
  operations
    effect()
end

class ReadProductInfo < DomainEvent, ExistingProductEvent
  operations
    effect()
end

association readProductInfo_language between
  ReadProductInfo[*]
  Language[1]
end

class ReadReview < DomainEvent, ExistingReviewEvent
  operations
    effect()
end

class ReorderLevelStockConfigurationChange < DomainEvent
  attributes
    newValue:Integer
  operations
    effect()
end

class RestorePreviousShoppingCart < DomainEvent, ExistingCustomerEvent
  operations
    effect()
end
CSTL and its application to the osCommerce case study.
Albert Tort

association restorePreviousShoppingCart_session between
   RestorePreviousShoppingCart[*]
   Session[0..1]
end

class ReviewTextMinimumChange < DomainEvent
attributes
   newMinimum:Integer
operations
   effect()
end

class SendExtraOrderEmailChange < DomainEvent, StoreEvent
operations
   effect()
end

association sendExtraOrderEmailChange_newSendExtraOrderEmail between
   SendExtraOrderEmailChange[*]
   NameEMail[*] role newSendExtraOrderEMail
end

class SendNewsletter < DomainEvent, ExistingNewsletterEvent
operations
   effect()
end

class SetCancelledOrderStatus < DomainEvent, StoreEvent
operations
   effect()
end

association setCancelledOrderStatus_orderStatus between
   SetCancelledOrderStatus[*]
   OrderStatus[1]
end

class SetCurrentCurrency < DomainEvent, ExistingSessionEvent
operations
   effect()
end

association setCurrentCurrency_currency between
   SetCurrentCurrency[*]
   Currency[1] role newCurrentCurrency
end

class SetCurrentLanguage < DomainEvent, ExistingSessionEvent
operations
   effect()
end

association setCurrentLanguage_language between
   SetCurrentLanguage[*]
   Language[1] role newCurrentLanguage
end

class SetDefaultCurrency < DomainEvent, ExistingCurrencyEvent
operations
   effect()
end

class SetDefaultLanguage < DomainEvent, ExistingLanguageEvent
operations
   effect()
end

class SetDefaultOrderStatus < DomainEvent, StoreEvent
operations
   effect()
end
CSTL and its application to the osCommerce case study.
Albert Tort

association setDefaultOrderStatus_orderStatus between
  SetDefaultOrderStatus[*]
  OrderStatus[1]
end

class ShowBanner < DomainEvent, ExistingBannerEvent
  operations
effect()
end

class ShowBestPurchasedProducts < Query
  operations
    answer():Set(Tuple(product:String, quantity:Integer)) =
      Product.allInstances
      -> sortedBy(quantityOrdered)
      -> collect (p | Tuple {product : ProductInLanguage.allInstances ->select
        (pil | pil.product = p and pil.language=language)->any(true).name,
        quantity : p.quantityOrdered})-asSet()
end
association showBestPurchasedProducts_language between
  ShowBestPurchasedProducts[*]
  Language[1]
end

class ShowBestViewedProducts < Query
  operations
    answer():Set(Tuple(product:String, timesViewed:Integer)) =
      Product.allInstances
      -> sortedBy(timesViewed())
      -> collect (p | Tuple {product : ProductInLanguage.allInstances ->select
        (pil | pil.product = p and pil.language=language)->any(true).name,
        timesViewed : p.timesViewed()})-asSet()
end
association showBestViewedProducts_language between
  ShowBestViewedProducts[*]
  Language[1]
end

class ShowCustomersOrdersTotal < Query
  operations
    answer():Set(Tuple(name:String, total:Real))=
      Customer.allInstances
      -> collect (c | Tuple {name : c.firstName.concat(c.lastName),
        total : c.order.total() -> sum()}) -> asSet()
end

class ShowExpectedProducts < Query
  operations
    answer(): Set(Tuple(product:String, dateAvailable:Date))=
      Product.allInstances -> select(p|p.available.isDefined())
      -> sortedBy(available.date)
      -> collect (p | Tuple {product : ProductInLanguage.allInstances ->select
        (pil | pil.product = p and pil.language=language)->any(true).name,
        dateAvailable : p.available}) -asSet()
end
association showExpectedProducts_language between
  ShowExpectedProducts[*]
  Language[1]
End
CSTL and its application to the osCommerce case study.
Albert Tort

class ShowNewProducts < Query
operations
  answer(): Set(Tuple(product:String, added:DateTime))=
    Product.allInstances
    -> sortedBy(added.dateTime)
    -> collect (p | Tuple {product : ProductInLanguage.allInstances
      -> select
        (pil | pil.product = p and
        pil.language=language) -> any(true).name,
        added : p.added}) -> asSet()
end

association showNewProducts_language between
  ShowNewProducts[*]
  Language[1]
end

class ShowOnlineCustomers < Query
operations
  answer(): Set(String)=
    Session.allInstances.customer
    -> collect (c | c.firstName.concat(c.lastName)) -> asSet()
end

class ShowOrdersOfCustomer < Query, ExistingCustomerEvent
operations
  answer(): Set(Tuple(id:Integer, total:Real, status:OrderStatus))=
    self.customer.order
    -> collect (o | Tuple {id : o.id(),
      total : o.total(),
      status : o.orderStatusChange-> last().orderStatus})
    -> asSet()
end

association showOrdersOfCustomer_language between
  ShowOrdersOfCustomer[*]
  Language[1]
end

class ShowProductsOfCategory < Query, ExistingCategoryEvent
operations
  answer(): Set(String)=
    Product.allInstances -> select(p | p.category -> includes(self.category))
    -> collect (p | ProductInLanguage.allInstances
      -> select
        (pil | pil.product = p and
        pil.language=language) -> any(true).name) -> asSet()
end

association showProductsOfCategory_language between
  ShowProductsOfCategory[*]
  Language[1]
end

class ShowProductsOfManufacturer < Query, ExistingManufacturerEvent
operations
  answer(): Set(String)=
    Product.allInstances -> select(p | p.manufacturer=self.manufacturer)
    -> collect (p | ProductInLanguage.allInstances
      -> select
        (pil | pil.product = p and
        pil.language=language) -> any(true).name) -> asSet()
end

association showProductsOfManufacturer_language between
  ShowProductsOfManufacturer[*]
  Language[1]
end

class ShowReviewsOfProduct < Query, ExistingProductEvent
operations
  answer(): Set(Tuple(review:String, rating:Rating))=
    self.product.review -> select (r | r.language = self.language)
    -> collect (r | Tuple {review : r.review,
      rating : r.rating}) -> asSet()
end
CSTL and its application to the osCommerce case study.
Albert Tort

association showReviewsOfProduct_language between
  ShowReviewsOfProduct[*]
  Language[1]
end

class ShowSpecials < Query
operations
  answer(): Set(Tuple(product:String,oldPrice:Real, newPrice:Real))=
    Special.allInstances
    -> collect (s | Tuple {product : ProductInLanguage.allInstances -> select
      pil | pil.product = s and
      pil.language=language} -> any(true).name,
      oldPrice : s.netPrice,
      newPrice : s.specialPrice)) -> asSet()
end

association showSpecials_language between
  ShowSpecials[*]
  Language[1]
end

class ShowUnderStockProducts < Query
operations
  answer(): Set(Tuple(product:String,quantity:Integer))=
    Product.allInstances -> select(p | p.quantityOnHand < Stock.allInstances
    -> any(true).stockReOrderLevel)
    -> collect (p | Tuple {product : ProductInLanguage.allInstances -> select
      pil | pil.product = p and pil.language=language} -> any(true).name,
      quantity : p.quantityOnHand}) -> asSet()
end

association showUnderStockProducts_language between
  ShowUnderStockProducts[*]
  Language[1]
end

class StateCustomerDetailChange < DomainEvent
attributes
  newValue:Boolean
operations
  effect()
end

class StateMinimumChange < DomainEvent
attributes
  newMinimum:Integer
operations
  effect()
end

class StatusPaymentMethodChange < DomainEvent, ExistingPaymentMethodEvent
attributes
  newStatus:Status
operations
  effect()
end

class StatusShippingMethodChange < DomainEvent, ExistingShippingMethodEvent
attributes
  newStatus:Status
operations
  effect()
end

abstract class ExistingPaymentMethodEvent
end

association existingPaymentMethodEvent_paymentMethod between
  ExistingPaymentMethodEvent[*]
  PaymentMethod[1]
end
abstract class ExistingShippingMethodEvent
end

association existingShippingMethodEvent_shippingMethod between
  ExistingShippingMethodEvent[*]
  ShippingMethod[1]
end

class StoreAddressAndPhoneChange < DomainEvent, StoreEvent
attributes
  newStoreAddressAndPhone:String
operations
  effect()
end

class StreetAddressMinimumChange < DomainEvent
attributes
  newMinimum:Integer
operations
  effect()
end

class SubstractStockConfigurationChange < DomainEvent
attributes
  newValue:Boolean
operations
  effect()
end

class SuburbCustomerDetailChange < DomainEvent
attributes
  newValue:Boolean
operations
  effect()
end

class SwitchToDefaultLanguageCurrencyChange < DomainEvent, StoreEvent
attributes
  newSwitchToDefaultLanguageCurrency:Boolean
operations
  effect()
end

class TaxDecimalPlacesChange < DomainEvent, StoreEvent
attributes
  newTaxDecimalPlaces:Integer
operations
  effect()
end

class TelephoneMinimumChange < DomainEvent
attributes
  newMinimum:Integer
operations
  effect()
end

class TypicalPackageTareWeightShippingConfigurationChange < DomainEvent
attributes
  newValue:Integer
operations
  effect()
end

class UninstallAuthorizeNetPaymentMethod < DomainEvent
operations
  effect()
end
CSTL and its application to the osCommerce case study.
Albert Tort

class UninstallCashOnDeliveryPaymentMethod < DomainEvent
  operations
  end

class UninstallCheckMoneyPaymentMethod < DomainEvent
  operations
    effect()
  end

class UninstallCreditCardPaymentMethod < DomainEvent
  operations
    effect()
  end

class UninstallFlatRateShippingMethod < DomainEvent
  operations
    effect()
  end

class UninstallIPaymentPaymentMethod < DomainEvent
  operations
    effect()
  end

class UninstallNochexPaymentMethod < DomainEvent
  operations
    effect()
  end

class UninstallPayPalPaymentMethod < DomainEvent
  operations
    effect()
  end

class UninstallPerItemShippingMethod < DomainEvent
  operations
    effect()
  end

class UninstallPSiGatePaymentMethod < DomainEvent
  operations
    effect()
  end

class UninstallSECPaymentMethod < DomainEvent
  operations
    effect()
  end

class UninstallTableRateShippingMethod < DomainEvent
  operations
    effect()
  end

class UninstallTwoCheckOutPaymentMethod < DomainEvent
  operations
    effect()
  end

class UninstallUSPostalServiceShippingMethod < DomainEvent
  operations
    effect()
  end

class UninstallZoneRatesShippingMethod < DomainEvent
  operations
    effect()
  end
CSTL and its application to the osCommerce case study.
Albert Tort

class UnlockNewsletter < DomainEvent, ExistingNewsletterEvent
operations
effect()
end

class UpdateCurrencyValueChange < DomainEvent, ExistingCurrencyEvent
attributes
newValue:Real
operations
effect()
end

class UpdateOrderStatus < DomainEvent, ExistingOrderEvent
attributes
comments:String
operations
effect()
end

association updateOrderStatus_zone between
UpdateOrderStatus[*]
OrderStatus[1] role newOrderStatus
end

class ZoneChange < DomainEvent, StoreEvent
operations
effect()
end

association zoneChange_zone between
ZoneChange[*]
Zone[1] role newZone
end

class UpdateShoppingCart < SessionEvent, ActionRequest
operations
effect()
end

abstract class ExistingShoppingCartItemEvent
end

association existingShoppingCartItemEvent_shoppingCartItem between
ExistingShoppingCartItemEvent[*]
ShoppingCartItem[1]
end

class LineChange
attributes
index:Integer
remove:Boolean
quantity:Integer
end

association updateShoppingCart_lineChange between
UpdateShoppingCart[*]
LineChange[1..*] ordered
end

class RemoveProduct < ExistingShoppingCartItemEvent
operations
effect()
end

class ChangeQuantity < ExistingShoppingCartItemEvent
attributes
quantity:Integer
operations
effect()
end
CSTL and its application to the osCommerce case study.
Albert Tort

-- CONSTRAINTS
constraints

category Store inv alwaysOneInstance:
    Store.allInstances->size()=1

category Store inv zoneIsPartOfCountry:
    self.zone->notEmpty() implies self.country.zone->includes(self.zone)

category ShippingAndPackaging inv tareIsLessThanMaximumWeight:
    self.typicalPackageTareWeight < self.maximumPackageWeight

category PaymentMethod inv atLeastOneEnabled:
    PaymentMethod.allInstances
        -> select(pm | pm.status=#enabled)->size() >= 1

category ShippingMethod inv atLeastOneEnabled:
    ShippingMethod.allInstances
        -> select(sm | sm.status=#enabled) -> size() >= 1

category Language inv codeAndNameAreUnique:
    Language.allInstances->isUnique(name) and Language.allInstances->isUnique(code)

category Currency inv codeAndTitleAreUnique:
    Currency.allInstances->isUnique(title) and Currency.allInstances->isUnique(code)

category Country inv nameAndCodesAreUnique:
    Country.allInstances->isUnique(name) and
    Country.allInstances->isUnique(isoCode2) and
    Country.allInstances->isUnique(isoCode3)

category Zone inv nameAndCountryAndCodeAndCountryAreUnique:
    Zone.allInstances->isUnique(Tuple{n:name, c:country}) and
    Zone.allInstances->isUnique(Tuple{n:code, c:country})

category TaxZone inv nameIsUnique:
    TaxZone.allInstances->isUnique(name)

category TaxClass inv nameIsUnique:
    TaxClass.allInstances->isUnique(name)

category Language inv optionNameIsUnique:
    self.hasOptionName->isUnique(optionName.string)

category Language inv valueNameIsUnique:
    self.hasValueName->isUnique(valueName.string)

category Language inv categoryNameIsUnique:
    self.hasCategoryName->isUnique(categoryName.string)

category Category inv isAHierarchy:
    not self.allParents() -> includes(self)

category Manufacturer inv nameIsUnique:
    Manufacturer.allInstances->isUnique(name)

category Manufacturer inv aURLInEachLanguage:
    self.language->size()=Language.allInstances->size()

category Banner inv titleIsUnique:
    Banner.allInstances->isUnique(title)

category BannerGroup inv nameIsUnique:
    BannerGroup.allInstances->isUnique(name)

category Newsletter inv titleIsUnique:
    Newsletter.allInstances->isUnique(title)
CSTL and its application to the osCommerce case study.
Albert Tort

context Customer inv eMailIsUnique:
    Customer.allInstances->isUnique(eMailAddress)

context Country inv addressesHaveZoneIfNeeded:
    self.zone->size()>0 implies self.address->forAll
        (a | a.state=a.zone.name and self=a.zone.country)

context CustomerShoppingCart inv sameCustomer:
    self.sessionOfShoppingCart.customer->notEmpty() implies
    self.sessionOfShoppingCart.customer=self.customer

context ShoppingCartItem inv productHasTheAttributes:
    self.product.attribute->includesAll(self.attribute)

context ShoppingCartItem inv onlyOneAttributePerOption:
    self.attribute->isUnique(option)

context Session inv sessionIDIsUnique:
    Session.allInstances->isUnique(sessionID)

context Order inv ApplicableZoneShippingMethod:
    self.shippingMethod.oclIsTypeOf(SpecificZoneMethod) and
    self.shippingMethod.oclAsType(SpecificZoneMethod).taxZone -> notEmpty implies
    self.shippingMethod.oclAsType(SpecificZoneMethod).taxZone.zone
    -> includes(self.delivery.zone)

context Order inv ApplicableZoneRatesShippingMethod:
    self.shippingMethod.oclIsTypeOf(ZoneRates) implies
    self.shippingMethod.oclAsType(ZoneRates).country -> includes(self.delivery.country)

context Order inv ApplicableZonesPaymentMethod:
    self.paymentMethod.taxZone -> notEmpty() implies
    self.paymentMethod.taxZone.zone -> includes(self.billing.zone)

context Order inv IDIsUnique:
    Order.allInstances -> isUnique(id())

context OrderStatus inv NameIsUnique:
    Language.allInstances->forAll(
        l | l.orderStatus->isUnique(orderStatusInLanguage.name)
    )

-- EVENT CONSTRAINTS

context TypicalPackageTareWeightShippingConfigurationChange inv
    _iniIC_valueDoesNotExceedMaxWeight:
        self.newValue < ShippingAndPackaging.allInstances->any(true).maximumPackageWeight

context MaximumPackageWeightShippingConfigurationChange inv
    _iniIC_maxIsGreaterThanTypicalWeight:
        self.newMaximum > ShippingAndPackaging.allInstances
        ->any(true).typicalPackageTareWeight

context EditCreditCardPaymentMethod inv _iniIC_DoNotImpliesAllPaymentMethodsDisabled:
    PaymentMethod.allInstances -> select(pm | not(pm.oclIsTypeOf(CreditCard)))
    -> exists(pm | pm.status=#enabled)

context EditManufacturer inv _iniIC_manufacturerDoesNotExist:
    (Manufacturer.allInstances - Set{self.manufacturer}).name->excludes(self.name)

context NewCategory inv _iniIC_categoryDoesNotExist:
    Language.allInstances->forAll(
        l | l.hasCategoryName.categoryName.string->excludes(self.hasNewName
            ->select(languageOfCategory=l)->any(true).name.string)
    )

context EditCountry inv _iniIC_countryDoesNotExist:
    (Country.allInstances - Set{self.country}).name->excludes(self.newName)
    and
    (Country.allInstances - Set{self.country}).isoCode2->excludes(self.newIsoCode2)
    and
    (Country.allInstances - Set{self.country}).isoCode3->excludes(self.newIsoCode3)
CSTL and its application to the osCommerce case study.

Albert Tort

context EditZone inv _iniIC_zoneDoesNotExist:
    (Zone.allInstances - Set{self.zone}).name->excludes(self.newName) and
    (Zone.allInstances - Set{self.zone}).code->excludes(self.newCode)

context EditTaxClass inv _iniIC_taxClassDoesNotExist:
    (TaxClass.allInstances - Set{self.taxClass}).name->excludes(self.newName)

context EditTaxZone inv _iniIC_taxZoneDoesNotExist:
    (TaxZone.allInstances - Set{self.taxZone}).name->excludes(self.newName)

context EditBannerGroup inv _iniIC_bannerGroupDoesNotExist:
    (BannerGroup.allInstances - Set{self.bannerGroup}).name->excludes(self.newName)

context EditBanner inv _iniIC_bannerDoesNotExist:
    (Banner.allInstances - Set{self.banner}).title->excludes(self.newTitle)

context LockNewsletter inv _iniIC_newsletterIsNotLocked:
    self.newsletter.status <> #locked

context UnlockNewsletter inv _iniIC_newsletterIsNotUnlocked:
    self.newsletter.status <> #unlocked

context EditNewsletter inv _iniIC_newsletterIsUnlocked:
    self.newsletter.status = #unlocked

context EditNewsletter inv _iniIC_newsletterDoesNotExist:
    (Newsletter.allInstances - Set{self.newsletter}).title->excludes(self.newTitle)

context DeleteNewsletter inv _iniIC_newsletterIsUnlocked:
    self.newsletter.status = #unlocked

context EditTaxRate inv _iniIC_taxRateDoesNotExist:
    (TaxRate.allInstances - Set{self.taxRate})->select(tr | tr.taxClass = self.newTaxClass and
    tr.taxZone = self.newTaxZone)->size()=0

context EditPerItemShippingMethod inv _iniIC_DoNotImpliesAllShippingMethodsDisabled:
    ShippingMethod.allInstances -> select(sm | not(sm.oclIsTypeOf(PerItem)))
    -> exists(sm | sm.status=#enabled)

context AttributeChange inv _iniIC_OptionAndValueAreAValidAttribute:
    Attribute.allInstances->exists(a| a.option=self.newOption and
    a.value=self.newValue)

context MoveProduct inv _iniIC_oldCategoryIsValid:
    product.category->includes(self.oldCategory)

context AddProductToShoppingCart inv _iniIC_AttributesAreFromProduct:
    self.product.attribute -> includesAll(self.attribute)

context AddProductToShoppingCart inv _iniIC_AttributesAreOfDifferentOptions:
    self.attribute -> isUnique(option)

context DeleteBannerGroup inv _iniIC_BannerGroupIsEmpty:
    self.bannerGroup.banner -> isEmpty()

context DeleteCountry inv _iniIC_CountryIsNotALocation:
    Store.allInstances -> any(true).country <> self.country and
    Address.allInstances.country -> excludes(self.country)

context DeleteCurrency inv _iniIC_ExistsAnotherCurrencyEnabled:
    Currency.allInstances -> select (c| c<>self.currency) -> exists(c|c.status=#enabled)

context DeleteCustomerAddress inv _iniIC_AddressOfCustomer:
    self.customer.address -> includes(self.address)

context DeleteCustomerAddress inv _iniIC_AtLeastTwoAddresses:
    self.customer.address->size() >= 2

context DeleteCustomerAddress inv _iniIC_PrimaryAddressCannotBeDeleted:
    self.address <> self.customer.primary
CSTL and its application to the osCommerce case study.

Albert Tort

context DeleteLanguage inv _iniIC_AtLeastTwoLanguages:
Language.allInstances -> size() >= 2

context DeleteOrderStatus inv _iniIC_IsNotTheCurrentStatusOfAnyOrder:
Order.allInstances ->forall (o | o.orderStatusChange -> last().orderStatus <> self.orderStatus)

context DeleteOrderStatus inv _iniIC_IsNotADefaultStatus:
Store.allInstances->forall(s | s.defaultStatus <> self.orderStatus and s.cancelledStatus <> self.orderStatus)

context DeleteProductOption inv _iniIC_HasNotProducts:
self.option.attribute.product -> isEmpty()

context DeleteProductOptionValue inv _iniIC_HasNotProducts:
self.value.attribute.product -> isEmpty() and self.value.attribute.orderLineAttribute->isEmpty()

context DeleteZone inv _iniIC_ZoneIsNotALocation:
Store.allInstances -> any(true).zone <> self.zone and Address.allInstances.zone -> excludes(self.zone)

context EditAuthorizeNetPaymentMethod inv _iniIC_PaymentMethodIsInstalled:
AuthorizeNet.allInstances -> notEmpty()

context EditCashOnDeliveryPaymentMethod inv _iniIC_PaymentMethodIsInstalled:
CashOnDelivery.allInstances -> notEmpty()

context EditCheckMoneyPaymentMethod inv _iniIC_PaymentMethodIsInstalled:
CheckMoney.allInstances -> notEmpty()

context EditCreditCardPaymentMethod inv _iniIC_PaymentMethodIsInstalled:
CreditCard.allInstances -> notEmpty()

context EditPerItemShippingMethod inv _iniIC_atLeastOneEnabled:
self.status=#disabled implies (ShippingMethod.allInstances-Set{PerItem.allInstances->any(true)})->exists(pm | pm.status=#enabled)

context EditCustomer inv _iniIC_firstNameRight:
self.newFirstName.size() >= MinimumValues.allInstances->any(true).firstName

context EditCustomer inv _iniIC_lastNameRight:
self.newLastName.size() >= MinimumValues.allInstances->any(true).lastName

context EditCustomer inv _iniIC_dateOfBirthRight:
CustomerDetails.allInstances->any(true).dateOfBirth implies
self.newDateOfBirth.isDefined and self.newDateOfBirth.date.size() >= MinimumValues.allInstances
->any(true).dateOfBirth

context EditCustomer inv _iniIC_genderRight:
CustomerDetails.allInstances->any(true).gender implies self.newGender.isDefined()

context EditCustomer inv _iniIC_eMailRight:
self.newEMailAddress.eMail.size() >= MinimumValues.allInstances
->any(true).eMailAddress

context EditCustomer inv _iniIC_telephoneRight:
self.newPhoneNumber.size() >= MinimumValues.allInstances->any(true).telephoneNumber

context EditLanguage inv _iniIC_languageDoesNotExist:
not ((Language.allInstances-Set{self.language})->exists(name=self.newName or code=self.newCode))

context EditCurrency inv _iniIC_currencyDoesNotExist:
not ((Currency.allInstances-Set{self.currency})->exists(title=self.newTitle or code=self.newCode))
CSTL and its application to the osCommerce case study.
Albert Tort

class CurrencyStatusChange

context CurrencyStatusChange inv _iniIC_atLeastOneCurrencyEnabled:
  self.newStatus=#disabled
  implies
  (Currency.allInstances-Set{self.currency})-exists(c|c.status=#enabled)

context EditCustomerAddress inv _iniIC_AddressOfCustomer:
  self.customer.address -> includes(self.address)

context EditCustomerAddress inv _iniIC_firstNameRight:
  self.newAddress.firstName.size() >= MinimumValues.allInstances->any(true).firstName

context EditCustomerAddress inv _iniIC_lastNameRight:
  self.newAddress.lastName.size() >= MinimumValues.allInstances->any(true).lastName

context EditCustomerAddress inv _iniIC_genderRight:
  CustomerDetails.allInstances->any(true).gender implies self.
  newAddress.gender.isDefined()

context EditCustomerAddress inv _iniIC_suburbRight:
  CustomerDetails.allInstances->any(true).suburb implies self.
  newAddress.suburb.isDefined()

context EditCustomerAddress inv _iniIC_streetAddressRight:
  self.newAddress.street.size() >= MinimumValues.allInstances->any(true).streetAddress

context EditCustomerAddress inv _iniIC_companyRight:
  CustomerDetails.allInstances->any(true).company implies
  self.newAddress.company.isDefined() and
  self.newAddress.company.size() >= MinimumValues.allInstances
  ->any(true).companyName

context EditCustomerAddress inv _iniIC_postCodeRight:
  self.newAddress.postCode.postalCode.size() >= MinimumValues.allInstances
  ->any(true).postCode

context EditCustomerAddress inv _iniIC_cityRight:
  self.newAddress.city.size() >= MinimumValues.allInstances->any(true).city

context EditCustomerAddress inv _iniIC_stateRight:
  CustomerDetails.allInstances->any(true).state implies
  self.newAddress.state.isDefined() and
  self.newAddress.state.size() >= MinimumValues.allInstances->any(true).state

context EditCustomerAddress inv _iniIC_addressesHaveZoneIfNeeded:
  self.newAddress.zone->size()>0 implies
  self.newAddress.state = self.newAddress.zone.name and
  self.newAddress.country = self.newAddress.zone.country

context EditCustomerDetails inv _iniIC_firstNameRight:
  self.newFirstName.size() >= MinimumValues.allInstances->any(true).firstName

context EditCustomerDetails inv _iniIC_lastNameRight:
  self.newLastName.size() >= MinimumValues.allInstances->any(true).lastName

context EditCustomerDetails inv _iniIC_dateOfBirthRight:
  CustomerDetails.allInstances->any(true).dateOfBirth implies
  self.newDateOfBirth.isDefined() and
  self.newDateOfBirth.date.size() >= MinimumValues.allInstances
  ->any(true).dateOfBirth

context EditCustomerDetails inv _iniIC_genderRight:
  CustomerDetails.allInstances->any(true).gender implies self.newGender.isDefined()

context EditCustomerDetails inv _iniIC_eMailRight:
  self.newEMailAddress.eMail.size() >= MinimumValues.allInstances
  ->any(true).eMailAddress

context EditCustomerDetails inv _iniIC_telephoneRight:
  self.newPhone.size() >= MinimumValues.allInstances->any(true).telephoneNumber
CSTL and its application to the osCommerce case study.
Albert Tort

context EditFlatRateShippingMethod inv _iniIC_PaymentMethodIsInstalled:
    FlatRate.allInstances -> notEmpty()

context EditIPaymentPaymentMethod inv _iniIC_PaymentMethodIsInstalled:
    IPayment.allInstances -> notEmpty()

context EditPerItemShippingMethod inv _iniIC_PaymentMethodIsInstalled:
    PerItem.allInstances -> notEmpty()

context EditPSiGatePaymentMethod inv _iniIC_PaymentMethodIsInstalled:
    PSiGate.allInstances -> notEmpty()

context EditSECPaymentMethod inv _iniIC_PaymentMethodIsInstalled:
    SICP.allInstances -> notEmpty()

context EditTableRateShippingMethod inv _iniIC_PaymentMethodIsInstalled:
    TableRate.allInstances -> notEmpty()

context EditTwoCheckOutPaymentMethod inv _iniIC_PaymentMethodIsInstalled:
    TwoCheckOut.allInstances -> notEmpty()

context EditUSPostalServiceShippingMethod inv _iniIC_PaymentMethodIsInstalled:
    USPostalService.allInstances -> notEmpty()

context EditZoneRatesShippingMethod inv _iniIC_PaymentMethodIsInstalled:
    ZoneRates.allInstances -> notEmpty()

context InstallAuthorizeNetPaymentMethod inv _iniIC_PaymentMethodIsNotInstalled:
    AuthorizeNet.allInstances -> isEmpty()

context InstallCashOnDeliveryPaymentMethod inv _iniIC_PaymentMethodIsNotInstalled:
    CashOnDelivery.allInstances -> isEmpty()

context InstallCheckMoneyPaymentMethod inv _iniIC_PaymentMethodIsNotInstalled:
    AuthorizeNet.allInstances -> isEmpty()

context InstallCreditCardPaymentMethod inv _iniIC_PaymentMethodIsNotInstalled:
    CreditCard.allInstances -> isEmpty()

context InstallFlatRateShippingMethod inv _iniIC_PaymentMethodIsNotInstalled:
    FlatRate.allInstances -> isEmpty()

context InstallIPaymentPaymentMethod inv _iniIC_PaymentMethodIsNotInstalled:
    IPayment.allInstances -> isEmpty()

context InstallNochexPaymentMethod inv _iniIC_PaymentMethodIsNotInstalled:
    Nochex.allInstances -> isEmpty()

context InstallPayPalPaymentMethod inv _iniIC_PaymentMethodIsNotInstalled:
    PayPal.allInstances -> isEmpty()

context InstallPerItemShippingMethod inv _iniIC_PaymentMethodIsNotInstalled:
    PerItem.allInstances -> isEmpty()

context InstallPSiGatePaymentMethod inv _iniIC_PaymentMethodIsNotInstalled:
    PSiGate.allInstances -> isEmpty()

context InstallSECPaymentMethod inv _iniIC_PaymentMethodIsNotInstalled:
    SECPay.allInstances -> isEmpty()

context InstallTableRateShippingMethod inv _iniIC_PaymentMethodIsNotInstalled:
    TableRate.allInstances -> isEmpty()
CSTL and its application to the osCommerce case study.
Albert Tort

context InstallTwoCheckoutPaymentMethod inv _iniIC_PaymentMethodIsNotInstalled:
   TwoCheckout.allInstances -> isEmpty()

countext InstallUSPostalServiceShippingMethod inv _iniIC_ShippingMethodIsNotInstalled:
   USPostalService.allInstances -> isEmpty()

countext InstallZoneRatesShippingMethod inv _iniIC_ShippingMethodIsNotInstalled:
   ZoneRates.allInstances -> isEmpty()

countext LogIn inv _iniIC_CustomerIsNotLoggedIn:
   self.customer.session -> isEmpty()

countext LogOut inv _iniIC_CustomerIsLoggedIn:
   self.session.customer = self.customer

countext NewBanner inv _iniIC_bannerDoesNotExist:
   not Banner.allInstances -> exists (b | b.title = self.title)

countext NewBannerGroup inv _iniIC_bannerGroupDoesNotExist:
   not BannerGroup.allInstances -> exists (bg | bg.name = self.name)

countext NewCountry inv _iniIC_countryDoesNotExist:
   not Country.allInstances -> exists (c | c.name = self.name and
   c.isoCode2 = self.isoCode2 and
   c.isoCode3 = self.isoCode3)

countext NewCurrency inv _iniIC_currencyDoesNotExist:
   not (Currency.allInstances -> exists (c | c.title = self.title and
   c.code = self.code))

countext NewCustomer inv _iniIC_passwordCorrect:
   password = passwordConfirmation

   self.primary.firstName.size() >= MinimumValues.allInstances->any(true).firstName

countext NewCustomer inv _iniIC_firstNameRight:
   self.primary.firstName.size() >= MinimumValues.allInstances->any(true).firstName

   self.primary.lastName.size() >= MinimumValues.allInstances->any(true).lastName

   countext NewCustomer inv _iniIC_dateOfBirthRight:
   CustomerDetails.allInstances->any(true).dateOfBirth implies
   self.dateOfBirth.isDefined() and
   self.dateOfBirth.date.size() >= MinimumValues.allInstances->any(true).dateOfBirth

   countext NewCustomer inv _iniIC_genderRight:
   CustomerDetails.allInstances->any(true).gender implies
   self.primary.gender.isDefined() and

   countext NewCustomer inv _iniIC_suburbRight:
   CustomerDetails.allInstances->any(true).suburb implies
   self.primary.suburb.isDefined() and

   countext NewCustomer inv _iniIC_eMailRight:
   self.eMailAddress.eMail.size() >= MinimumValues.allInstances->any(true).eMailAddress

   countext NewCustomer inv _iniIC_streetAddressRight:
   self.primary.street.size() >= MinimumValues.allInstances->any(true).streetAddress

   countext NewCustomer inv _iniIC_companyRight:
   self.primary.company.isDefined() and
   self.primary.company.size() >= MinimumValues.allInstances->any(true).companyName

   countext NewCustomer inv _iniIC_postCodeRight:
   self.primary.postCode.postalCode.size() >= MinimumValues.allInstances
   ->any(true).postCode

   countext NewCustomer inv _iniIC_cityRight:
   self.primary.city.size() >= MinimumValues.allInstances->any(true).city
CSTL and its application to the osCommerce case study.
Albert Tort

context NewCustomer inv _iniIC_stateRight:
    CustomerDetails.allInstances->any(true).state implies
    self.primary.state.isDefined() and
    self.primary.state.size() >= MinimumValues.allInstances->any(true).state

context NewCustomer inv _iniIC_telephoneRight:
    self.phone.size() >= MinimumValues.allInstances->any(true).telephoneNumber

context NewCustomer inv _iniIC_passwordRight:
    self.password.size() >= MinimumValues.allInstances->any(true).password

context NewCustomerAddress inv _iniIC_firstNameRight:
    self.firstName.size() >= MinimumValues.allInstances->any(true).firstName

context NewCustomerAddress inv _iniIC_lastNameRight:
    self.lastName.size() >= MinimumValues.allInstances->any(true).lastName

context NewCustomerAddress inv _iniIC_genderRight:
    CustomerDetails.allInstances->any(true).gender implies self.gender.isDefined()

context NewCustomerAddress inv _iniIC_suburbRight:
    CustomerDetails.allInstances->any(true).suburb implies self.suburb.isDefined()

context NewCustomerAddress inv _iniIC_streetAddressRight:
    self.street.size() >= MinimumValues.allInstances->any(true).streetAddress

context NewCustomerAddress inv _iniIC_companyRight:
    CustomerDetails.allInstances->any(true).company implies
    self.company.isDefined() and
    self.company.size() >= MinimumValues.allInstances->any(true).companyName

context NewCustomerAddress inv _iniIC_postCodeRight:

context NewCustomerAddress inv _iniIC_cityRight:
    self.city.size() >= MinimumValues.allInstances->any(true).city

context NewCustomerAddress inv _iniIC_stateRight:
    CustomerDetails.allInstances->any(true).state implies
    self.state.isDefined() and
    self.state.size() >= MinimumValues.allInstances->any(true).state

context NewCustomerAddress inv _iniIC_addressesHaveZoneIfNeeded:
    self.country.zone->size()>0 implies
    (self.state = self.zone.name and
    self.country = self.zone.country)

context NewCustomerAddress inv _iniIC_numberOfAddressesRight:
    self.customer.address -> size() < MaximumValues.allInstances
    ->any(true).addressBookEntries

context NewDownloadableProductAttribute inv _iniIC_productAttributeDoesNotExist:
    not ProductAttribute.allInstances -> exists (pa | pa.attribute.option =
    self.option and pa.attribute.value = self.value and
    pa.product = self.product)

context NewLanguage inv _iniIC_languageDoesNotExist:
    not (Language.allInstances -> exists (l | l.name=self.newName and l.code =
    self.newCode))

context NewManufacturer inv _iniIC_manufacturerDoesNotExist:
    not Manufacturer.allInstances -> exists (m | m.name=self.name)

context NewNewsletter inv _iniIC_newsletterDoesNotExist:
    not Newsletter.allInstances -> exists (n | n.title=self.title)

context NewOrderStatus inv _iniIC_orderStatusDoesNotExist:
    not OrderStatus.allInstances -> exists (os | Language.allInstances->
    exists(l|
CSTL and its application to the osCommerce case study.
Albert Tort

self.hasOrderStatusName
  ->select((languageOfOrderStatus=1).orderStatusName.string= os.orderStatusInLanguage)
  select((language=1).name)

context NewProduct inv _iniIC_productDoesNotExist:
  Language.allInstances -> forAll ( l |
  1.productInLanguage.name
  -> excludes(self.hasNewProductName
  -> any((languageOfProduct=l).nameOfProduct.string))

context EditProduct inv _iniIC_productDoesNotExist:
  Language.allInstances -> forAll ( l |
  1.productInLanguage.name
  -> excludes(self.hasNewProductName
  -> any((languageOfProduct=l).nameOfProduct.string) or
  (self.hasNewProductName->any((languageOfProduct=l).nameOfProduct.string =
  self.product.productInLanguage->any((language=1).name))

context EditProductOption inv _iniIC_productOptionDoesNotExist:
  Language.allInstances -> forAll ( l |
  1.hasOptionName.optionName
  -> excludes(self.hasNewOptionName -> any((languageOfOption=l).nameOfOption) or
  (self.hasNewOptionName->any((languageOfOption=l).nameOfOption =
  self.option.hasOptionName->any((optionLanguage=l).optionName))

context EditCategory inv _iniIC_categoryDoesNotExist:
  Language.allInstances -> forAll ( l |
  1.hasCategoryName.categoryName.string
  -> excludes(self.hasNewName -> any((languageOfCategory=l).name.string) or
  (self.hasNewName->any((languageOfCategory=l).name.string =
  self.category.hasCategoryName->any((language=l).categoryName.string))

context EditOrderStatus inv _iniIC_orderStatusDoesNotExist:
  Language.allInstances -> forAll ( l |
  1.orderStatusInLanguage.name
  ->excludes(self.hasOrderStatusName
  -> any((languageOfOrderStatus=l).orderStatusName.string)
  or
  1.orderStatusInLanguage->any(orderStatus=self.orderStatus).name =
  self.hasOrderStatusName->any((languageOfOrderStatus=l).orderStatusName.string)
  )

context EditCategory inv _iniIC_cyclesDoNotAppear:
  self.category.allParents()->union(Set{self.newParent})->excludes(self.category)

context MoveCategory inv _iniIC_cyclesDoNotAppear:
  self.newParent.allParents()->excludes(self.category)

context EditProductOptionValue inv _iniIC_productOptionValueDoesNotExist:
  Language.allInstances -> forAll ( l |
  1.hasValueName.valueName
  -> excludes(self.hasNewValueName -> any((languageOfValue=l).nameOfValue) or
  (self.hasNewValueName->any((languageOfValue=l).nameOfValue =
  self.value.hasValueName->any((valueLanguage=l).valueName))

context NewProductAttribute inv _iniIC_productAttributeDoesNotExist:
  not self.product.productAttribute ->
  exists(attribute.value=self.value and
  attribute.option = self.option)

context NewProductAttribute inv _iniIC_optionValueIsValid:
  self.option.value -> includes(self.value)

context NewProductNotification inv _iniIC_ProductNotificationDoesNotExist:
  not Newsletter.allInstances -> exists (n | n.title = self.title)

context NewProductNotificationSubscription inv _iniIC_ProductIsUnsubscribed:
  not self.customer.globalNotifications and
  self.customer.explicitNotifications -> excludes(self.newSubscribedProduct)
CSTL and its application to the osCommerce case study.
Albert Tort

context NewProductOption inv _iniIC_productOptionDoesNotExist:
Language.allInstances -> forAll ( l | l.hasOptionName.optionName -> excludes(self.hasNewOptionName -> select(languageOfOption=l).nameOfOption->any(true)))

context NewProductOptionValue inv _iniIC_optionValueDoesNotExist:
Language.allInstances -> forAll ( l | l.hasValueName.valueName.string
-> excludes(self.hasNewValueName -> select(languageOfValue=l).nameOfValue
->any(true).string))

context NewReview inv _iniIC_reviewRight:
self.review.size() >= MinimumValues.allInstances->any(true).reviewText

context NewTaxClass inv _iniIC_TaxClassDoesNotExist:
not TaxClass.allInstances -> exists (tc | tc.name = self.name)

context NewTaxRate inv _iniIC_TaxRateDoesNotExist:
not TaxRate.allInstances -> exists (tr | tr.taxClass = self.taxClass and tr.taxZone = self.taxZone)

context NewTaxZone inv _iniIC_TaxZoneDoesNotExist:
not TaxZone.allInstances -> exists (tz | tz.name = self.name)

context NewZone inv _iniIC_ZoneDoesNotExist:
not Zone.allInstances -> exists (z | z.name = self.name and z.country = self.country or z.code = self.code and z.country = self.country)

context OrderConfirmation inv _iniIC_ShippingMethodIsEnabled:
self.shippingMethod.status= #enabled

context OrderConfirmation inv _iniIC_PaymentMethodIsEnabled:
self.paymentMethod.status= #enabled

context OrderConfirmation inv _iniIC_CurrencyIsEnabled:
self.currency.status = #enabled

context OrderConfirmation inv _iniIC_CreditCardDetailsNeeded:
self.paymentMethod.oclIsTypeOf(AuthorizeNet) or
self.paymentMethod.oclIsTypeOf(CreditCard) or
self.paymentMethod.oclIsTypeOf(IPayment) or
self.paymentMethod.oclIsTypeOf(TwoCheckOut) or
self.paymentMethod.oclIsTypeOf(PSiGate)
implies
creditCardType.isDefined() and
creditCardOwner.isDefined() and
creditCardNumber.isDefined() and
creditCardExpires.isDefined()

context OrderConfirmation inv _iniIC_StockAllowsOrder:
Stock.allInstances->any(true).allowCheckout or
not Stock.allInstances->any(true).checkStockLevel or
(self.shoppingCart.shoppingCartItem.product -> forAll (p | p.quantityOnHand > 0))

context PasswordChange inv _iniIC_passwordRight:
self.newPassword.size() => MinimumValues.allInstances->any(true).password

context PasswordChange inv _iniIC_oldPasswordIsCorrect:
self.customer.password = self.OLDPassword

context PrimaryCustomerAddressChange inv _iniIC_AddressOfCustomer:
self.customer.address -> includes(self.address)

context ProductDownload inv _iniIC_DownloadEnabled:
Download.allInstances->any(true).enableDownload

context ProductDownload inv _iniIC_ProductWasPurchasedByCustomer:
self.customer.order.orderLine.product -> includes (self.product)
CSTL and its application to the osCommerce case study.
Albert Tort

context ProductDownload inv _iniIC_DownloadableIsFromProduct:
  self.product.productAttribute -> select(pa | pa.oclIsTypeOf(Downloadable))
  -> includes (self.downloadable)

context ProductDownload inv _iniIC_DownloadsCountNotExceeded:
  let DownloadCountFromProduct:Integer =
  self.customer.order.orderLine.orderLineAttribute
  -> select (ola | ola.oclIsTypeOf(OrderDownload) and ola.orderLine.product = self.product)
  ->asSequence()->last().oclAsType(OrderDownload).downloadCount
  in
  DownloadCountFromProduct < self.downloadable.maximumDownloadCount

context RestorePreviousShoppingCart inv _iniIC_CustomerHasAPreviousShoppingCart:
  self.customer.customerShoppingCart->notEmpty()

context UninstallAuthorizeNetPaymentMethod inv _iniIC_PaymentMethodIsNotUninstalled:
  AuthorizeNet.allInstances -> notEmpty()

context UninstallCashOnDeliveryPaymentMethod inv _iniIC_PaymentMethodIsNotUninstalled:
  CashOnDelivery.allInstances -> notEmpty()

context UninstallCheckMoneyPaymentMethod inv _iniIC_PaymentMethodIsNotUninstalled:
  AuthorizeNet.allInstances -> notEmpty()

context UninstallCreditCardPaymentMethod inv _iniIC_PaymentMethodCanBeUninstalled:
  CreditCard.allInstances -> notEmpty() and
  {PaymentMethod.allInstances->Set{CreditCard.allInstances->any(true)}}
  ->exists(pm|pm.status=#enabled)

context UninstallFlatRateShippingMethod inv _iniIC_ShippingMethodIsNotUninstalled:
  FlatRate.allInstances -> notEmpty()

context UninstallIPaymentPaymentMethod inv _iniIC_PaymentMethodIsNotUninstalled:
  IPayment.allInstances -> notEmpty()

context UninstallNochexPaymentMethod inv _iniIC_PaymentMethodIsNotUninstalled:
  Nochex.allInstances -> notEmpty()

context UninstallPayPalPaymentMethod inv _iniIC_PaymentMethodIsNotUninstalled:
  PayPal.allInstances -> notEmpty()

context UninstallPerItemShippingMethod inv _iniIC_ShippingMethodCanBeUninstalled:
  PerItem.allInstances -> notEmpty() and
  {ShippingMethod.allInstances->Set{PerItem.allInstances->any(true)}}
  ->exists(sm|sm.status=#enabled)

context UninstallPSiGatePaymentMethod inv _iniIC_PaymentMethodIsNotUninstalled:
  PSiGate.allInstances -> notEmpty()

context UninstallSECPaymentMethod inv _iniIC_PaymentMethodIsNotUninstalled:
  SECPay.allInstances -> notEmpty()

context UninstallTableRateShippingMethod inv _iniIC_ShippingMethodIsNotUninstalled:
  TableRate.allInstances -> notEmpty()

context UninstallTwoCheckOutPaymentMethod inv _iniIC_PaymentMethodIsNotUninstalled:
  TwoCheckOut.allInstances -> notEmpty()

context UninstallUSPostalServiceShippingMethod inv _iniIC_ShippingMethodIsNotUninstalled:
  USPostalService.allInstances -> notEmpty()
context UninstallZoneRatesShippingMethod inv _iniIC_ShippingMethodIsNotUninstalled:
ZoneRates.allInstances -> notEmpty()

context UpdateShoppingCart inv _iniIC_complete:
self.lineChange->size() = self.session.shoppingCart.shoppingCartItem->size()

-- EFFECT OPERATIONS
context AddProductToShoppingCart::effect()
post ShoppingCartItemsIsCreated:
(ShoppingCartItem.allInstances - ShoppingCartItem.allInstances@pre) -> forAll(sci:ShoppingCartItem | sci.oclIsNew and sci.oclIsTypeOf(ShoppingCartItem) and sci.quantity = self.quantity and sci.product = self.product and sci.attribute = self.attribute and if self.session.shoppingCart -> notEmpty() then --The session has a shopping cart
  self.session.shoppingCart.shoppingCartItem -> includes(sci)
else --The session does not have a shopping cart
  if self.session.customer -> isEmpty() then --The session is Anonymous
    (AnonymousShoppingCart.allInstances - AnonymousShoppingCart.allInstances@pre) -> forAll(sc:AnonymousShoppingCart | sc.oclIsNew() and sc.oclIsTypeOf(AnonymousShoppingCart) and self.session.shoppingCart = sc and sc.shoppingCartItem -> includes(sci))
  else --The customer has logged in
    if self.session.customer.customerShoppingCart -> notEmpty() then --The customer has a previous shopping cart
      self.session.shoppingCart = self.session.customer.customerShoppingCart and self.session.shoppingCart.shoppingCartItem -> includes(sci)
    else --The customer does not have a previous shopping cart
      (CustomerShoppingCart.allInstances - CustomerShoppingCart.allInstances@pre) -> forAll(csc:CustomerShoppingCart | csc.oclIsNew() and csc.oclIsTypeOf(CustomerShoppingCart) and self.session.shoppingCart = csc and csc.shoppingCartItem -> includes(sci))
    endif
  endif
endif

context AddressBookEntriesMaximumChange::effect()
post : MaximumValues.allInstances->any(true).addressBookEntries = self.newMaximum

context AllowCheckoutStockConfigurationChange::effect()
post : Stock.allInstances->any(true).allowCheckout = self.newValue

context AllowGuestToTellAFriendChange::effect()
post : myStore().allowGuestToTellAFriend = self.newAllowGuestToTellAFriend

context AttributeChange::effect()
post :
self.productAttribute.attribute.value = self.newValue and self.productAttribute.attribute.option = self.newOption

context CancelOrder::effect()
post:
self.order.orderStatusChange -> last().orderStatus = Store.allInstances -> any(true).cancelledStatus
CSTL and its application to the osCommerce case study.
Albert Tort

context CheckLevelStockConfigurationChange::effect()
post : Stock.allInstances->any(true).checkStockLevel= self.newValue

context CityMinimumChange::effect()
post : MinimumValues.allInstances->any(true).city = self.newMinimum

context ClickBanner::effect()
post :
BannerHistory.allInstances -> one
{bh | bh.banner = self.banner and
bh.clicked = bh@pre.clicked + 1}

context ClickManufacturer::effect()
post :
let manufacturerLanguageRead:ManufacturerInLanguage =
ManufacturerInLanguage.allInstances -> select
{mil | mil.manufacturer = self.manufacturer and
mil.language = self.language})->any(true)
in
manufacturerLanguageRead.urlClicked =
manufacturerLanguageRead@pre.urlClicked + 1

context CompanyCustomerDetailChange::effect()
post : CustomerDetails.allInstances->any(true).company = self.newValue

context CompanyNameMinimumChange::effect()
post : MinimumValues.allInstances->any(true).companyName = self.newMinimum

context CountryChange::effect()
post : myStore().country = self.newCountry

context CountryShippingConfigurationChange::effect()
post : ShippingAndPackaging.allInstances->any(true).countryOfOrigin =
self.newCountryOfOrigin

context CreditCardNumberMinimumChange::effect()
post : MinimumValues.allInstances->any(true).creditCardNumber = self.newMinimum

context CreditCardOwnerNameMinimumChange::effect()
post : MinimumValues.allInstances->any(true).creditCardOwnerName = self.newMinimum

context CurrencyStatusChange::effect()
post : self.currency.status = self.newStatus

context CustomerStatusChange::effect()
post : self.customer.status = self.newStatus

context DateOfBirthCustomerDetailChange::effect()
post : CustomerDetails.allInstances->any(true).dateOfBirth = self.newValue

context DateOfBirthMinimumChange::effect()
post : MinimumValues.allInstances->any(true).dateOfBirth = self.newMinimum

context DaysExpiryDelayDownloadConfigurationChange::effect()
post : Download.allInstances->any(true).daysExpiryDelay= self.newValue

context DefaultSearchOperatorChange::effect()
post : myStore().defaultSearchOperator = self.newDefaultSearchOperator

context DeleteBanner::effect()
post : Banner.allInstances->excludes(self.banner@pre)

context DeleteBannerGroup::effect()
post : BannerGroup.allInstances->excludes(self.bannerGroup@pre)

context DeleteCategory::effect()
post deleteCategoryAndSubcategories:
Category.allInstances->excludes(self.category@pre) and
self.allChilds(category@pre) -> forAll(c | Category.allInstances
->excludes(c))
CSTL and its application to the osCommerce case study.
Albert Tort

post deleteProductsOfCategory:
    self.category@pre.product -> forall(p |
        if p.orderLine->notEmpty() then p.status=#outOfStock
        else Product.allInstances->excludes(p@pre)
    )

post deleteProductsOfChildCategories:
    self.allChildren(category@pre).product -> forall(p |
        if p.orderLine->notEmpty() then p.status=#outOfStock
        else Product.allInstances->excludes(p@pre)
    )

context DeleteCountry::effect()
    post : Country.allInstances->excludes(self.country@pre)
    post : self.country@pre.zone -> forall(z | Zone.allInstances->excludes(z))

context DeleteCurrency::effect()
    post : Currency.allInstances->excludes(self.currency@pre)

context DeleteCustomer::effect()
    post deleteCustomer:
        if customer@pre.order->size()=0 then
            Customer.allInstances->excludes(customer@pre)
        else
            customer.status=#disabled
        endif
    post deleteReviews:
        Review.allInstances->excludesAll(customer@pre.review@pre)
    post deleteShoppingCartIfNeeded:
        customer@pre.customerShoppingCart->size()>0
        implies
        ShoppingCart.allInstances->excludes(customer@pre.customerShoppingCart@pre)

context DeleteCustomerAddress::effect()
    post : self.customer.address -> excludes(self.address)

context DeleteLanguage::effect()
    post : not Language.allInstances->exists(l | l=self.language@pre)

context DeleteManufacturer::effect()
    post deleteManufacturer:
        Manufacturer.allInstances->excludes(self.manufacturer@pre)
    post changeProductsToOutOfStock:
        deleteProds implies
        manufacturer@pre.product@pre ->
        forall(status = #outOfStock)

context DeleteNewsletter::effect()
    post : Newsletter.allInstances->excludes(self.newsletter@pre)

context DeleteOrderStatus::effect()
    post : if Order.allInstances.orderStatus->includes(self.orderStatus)
        then self.orderStatus.status=#disabled
        else OrderStatus.allInstances->excludes(self.orderStatus@pre)
    endif

context DeleteProduct::effect()
    post : if product@pre.orderLine -> size()=0
    then Product.allInstances->excludes(product@pre)
    else
        (ProductStatusChange.allInstances - ProductStatusChange.allInstances@pre)
        -> forall(psc:ProductStatusChange | psc.oclIsNew() and
        psc.oclIsTypeOf(ProductStatusChange) and
        psc.newStatus = #outOfStock and
        psc.product = self.product@pre)
    endif
CSTL and its application to the osCommerce case study.
Albert Tort

context DeleteProductAttribute::effect()
  post: if OrderLineAttribute.allInstances -> exists(o | 
    o.attribute=productAttribute.attribute and 
    o.orderLine.product=productAttribute.product) 
  then 
    productAttribute.status=#disabled 
  else 
    ProductAttribute.allInstances->excludes(productAttribute@pre) 
  endif

context DeleteProductOption::effect()
  post: Option.allInstances->excludes(self.option@pre)
  post: self.option@pre.value->select(v|
    (v.option=Set{self.option@pre})->isEmpty() or 
    v.attribute.orderLineAttribute->isEmpty())

context DeleteProductOptionValue::effect()
  post: Value.allInstances->excludes(self.value@pre)

context DeleteReview::effect()
  post: Review.allInstances->excludes(self.review@pre)

context DeleteSession::effect()
  post: Session.allInstances->excludes(self.session@pre)

context DeleteSpecial::effect()
  post: Special.allInstances->excludes(special@pre)
  post: (Product.allInstances - Product.allInstances@pre) -> forAll(p:Product | 
    p.status = special@pre.status@pre and 
    p.available = special@pre.available@pre and 
    p.netPrice = special@pre.netPrice@pre and 
    p.quantityOnHand = special@pre.quantityOnHand@pre and 
    p.modelM = special@pre.modelM@pre and 
    p.imagePath = special@pre.imagePath@pre and 
    p.weight = special@pre.weight@pre and 
    p.category = special@pre.category@pre and 
    p.manufacturer = special@pre.manufacturer@pre and 
    p.taxClass = special@pre.taxClass@pre and 
    Language.allInstances ->
      forAll (l| 
        special@pre.productInLanguage->select(language=l).name = 
        p.productInLanguage->select(language=l).name)

context DeleteTaxClass::effect()
  post deleteTaxClass:
    TaxClass.allInstances->excludes(self.taxClass@pre)
  post deleteAssociatedTaxRates:
    self.taxClass@pre.taxRate@pre -> forAll(tr | TaxRate.allInstances->excludes(tr))

context DeleteTaxRate::effect()
  post: TaxRate.allInstances->excludes(self.taxRate@pre)

context DeleteTaxZone::effect()
  post deleteTaxZone:
    TaxZone.allInstances->excludes(self.taxZone@pre)
  post deleteAssociatedTaxRates:
    self.taxZone@pre.taxRate@pre -> forAll(tr | TaxRate.allInstances->excludes(tr))

context DeleteZone::effect()
  post: Zone.allInstances->excludes(self.zone@pre)

context DisplayCartAfterAddingProductChange::effect()
  post: myStore().displayCartAfterAddingProduct = 
    self.newDisplayCartAfterAddingProduct

context DisplayPricesWithTaxChange::effect()
  post: myStore().displayPricesWithTax = self.newDisplayPricesWithTax
CSTL and its application to the osCommerce case study.
Albert Tort

context  EditAuthorizeNetPaymentMethod::effect()
post : 
  let pm:AuthorizeNet = AuthorizeNet.allInstances -> any(true) in
  pm.username=self.newUsername and
  pm.key=self.newKey and
  pm.mode=self.newMode and
  pm.method=self.newMethod and
  pm.notification=self.newNotification and
  pm.orderStatus=self.orderStatus and
  pm.status=self.status and
  pm.taxZone=self.taxZone

context  EditBanner::effect()
post :
  self.banner.title = self.newTitle and
  self.banner.url = self.newUrl and
  self.banner.imagePath = self.newImagePath and
  self.banner.html = self.newHtml and
  self.banner.expires = self.newExpires and
  self.banner.scheduled = self.newScheduled and
  self.banner.status = self.newStatus and
  self.banner.bannerGroup=self.newBannerGroup

context  EditBannerGroup::effect()
post :  self.bannerGroup.name = self.newName

context  EditCashOnDeliveryPaymentMethod::effect()
post : 
  let pm:CashOnDelivery = CashOnDelivery.allInstances -> any(true) in
  pm.orderStatus=self.orderStatus and
  pm.status=self.status and
  pm.taxZone=self.taxZone

context  EditCategory::effect()
post :
  self.category.imagePath = self.imagePath and
  self.category.sortOrder = self.sortOrder and
  self.category.parent = self.newParent and
  Language.allInstances
  -> forAll (l| self.hasNewName -> select(languageOfCategory=l)->any(true).name.string =
  self.category.hasCategoryName->select(language=l).categoryName
  ->any(true).string
  )

context  EditCheckMoneyPaymentMethod::effect()
post : 
  let pm:CheckMoney = CheckMoney.allInstances -> any(true) in
  pm.makePayableTo=self.newMakePayableTo and
  pm.orderStatus=self.orderStatus and
  pm.status=self.status and
  pm.taxZone=self.taxZone

context  EditCountry::effect()
post :
  country.name = self.newName and
  country.isoCode2 = self.newIsoCode2 and
  country.isoCode3 = self.newIsoCode3

context  EditCreditCardPaymentMethod::effect()
post : 
  let pm:CreditCard = CreditCard.allInstances -> any(true) in
  pm.splitCreditCardToMail=self.newSplitCreditCardToMail and
  pm.orderStatus=self.orderStatus and
  pm.status=self.status and
  pm.taxZone=self.taxZone

context  EditCurrency::effect()
post :
  currency.title = self.newTitle

240
CSTL and its application to the osCommerce case study.
Albert Tort

currency.code = self.newCode and
currency.symbolLeft = self.newSymbolLeft and
currency.symbolRight = self.newSymbolRight and
currency.decimalPlaces = self.newDecimalPlaces and
currency.value = self.newValue

context EditCustomer::effect()
post :
customer.gender = self.newGender and
customer.firstName = self.newFirstName and
customer.lastName = self.newLastName and
customer.dateOfBirth = self.newDateOfBirth and
customer.eMailAddress = self.newEMailAddress and
customer.phone = self.newPhone and
customer.fax = self.newFax and
customer.newsletter = self.newNewsletter and
customer.password = self.newPassword and
customer.globalNotifications = self.newGlobalNotifications

context EditCustomerAddress::effect()
post :
self.customer.address -> excludes(self.address) and
self.customer.address -> includes(self.newAddress)

context EditCustomerDetails::effect()
post :
customer.gender = self.newGender and
customer.firstName = self.newFirstName and
customer.lastName = self.newLastName and
customer.dateOfBirth = self.newDateOfBirth and
customer.eMailAddress = self.newEMailAddress and
customer.phone = self.newPhone and
customer.fax = self.newFax and
customer.newsletter = self.newNewsletter

context EditDownloadableAttribute::effect()
post :
s.downloadable.filename = self.newFilename and
downloadable.expiryDays = self.newExpiryDays and
downloadable.maximumDownloadCount = self.newMaximumDownloadCount

context EditFlatRateShippingMethod::effect()
post :
let sm: FlatRate= FlatRate.allInstances -> any(true) in
sm.cost = self.newCost and
sm.taxZone = self.taxZone and
sm.taxClass = self.taxClass and
sm.status = self.status

context EditGlobalNotifications::effect()
post : self.customer.globalNotifications = self.newGlobalNotifications

context EditIPaymentPaymentMethod::effect()
post :
let pm: IPayment = IPayment.allInstances -> any(true) in
pm.account = self.newAccount and
pm.user = self.newUser and
pm.password = self.newPassword and
pm.status = self.status and
pm.orderStatus = self.orderStatus and
pm.taxZone = self.taxZone

context EditLanguage::effect()
post :
s.language.name = self.newName and
s.language.code = self.newCode and
language.defaultCurrency = self.newDefaultCurrency

context EditManufacturer::effect()
post :
s.manufacturer.name = self.name and
s.manufacturer.imagePath = self.imagePath and
CSTL and its application to the osCommerce case study.
Albert Tort

Language.allInstances ->
  forAll(l|
    self.hasURL->select(languageOfURL=l).url=
    self.manufacturer.manufacturerInLanguage->
      select(language=l).url)

context EditNewsletter::effect()
  post :
    newsletter.title = self.newTitle and
    newsletter.content = self.newContent

def
context EditNochexPaymentMethod::effect()
  post :
    let pm: Nochex = Nochex.allInstances -> any(true) in
    pm.eMail= self.newEMail and
    pm.status= self.status and
    pm.orderStatus= self.orderStatus and
    pm.taxZone= self.taxZone

context EditOrderStatus::effect()
  post :
    Language.allInstances->
      forAll(l|
        self.hasOrderStatusName
        -> select(languageOfOrderStatus=l).orderStatusName.string=
        self.orderStatus.orderStatusInLanguage->
          select(language=l).name)

context EditPayPalPaymentMethod::effect()
  post :
    let pm: PayPal = PayPal.allInstances -> any(true) in
    pm.eMail= self.newEMail and
    pm.status= self.status and
    pm.orderStatus= self.orderStatus and
    pm.taxZone= self.taxZone

context EditPerPageShippingMethod::effect()
  post :
    let sm: PerItem= PerItem.allInstances -> any(true) in
    sm.cost= self.newCost and
    sm.handlingFee= self.handlingFee and
    sm.taxZone= self.taxZone and
    sm.taxClass= self.taxClass and
    sm.status = self.status

context EditProduct::effect()
  post :
    self.product.status = self.status and
    self.product.available = self.available and
    self.product.netPrice = self.netPrice and
    self.product.quantityOnHand = self.quantityOnHand and
    self.product.modelM = self.modelM and
    self.product.imagePath = self.imagePath and
    self.product.weight = self.weight and
    self.product.manufacturer = self.manufacturer and
    self.product.category = self.category and
    self.product.taxClass = self.taxClass and
    Language.allInstances
      -> forAll (l|
      self.hasNewProductName -> select(languageOfProduct=l).nameOfProduct
        ->any(true).string =
      self.product.productInLanguage->select(language=l).name->any(true)

context EditProductNotification::effect()
  post :
    self.productNotification.global = self.newGlobal and
    self.productNotification.explicitNotifications = self.newExplicitNotifications
CSTL and its application to the osCommerce case study.
Albert Tort

context EditProductOption::effect()
  post:
  Language.allInstances ->
  forAll (l| self.hasNewOptionName -> select(languageOfOption=l).nameOfOption =
  option.hasOptionName->select(optionLanguage=l).optionName)

context EditProductOptionValue::effect()
  post:
  Language.allInstances ->
  forAll (l| self.hasNewValueName -> select(languageOfValue=l).nameOfValue =
  value.hasValueName->select(valueLanguage=l).valueName) and
  self.value.option = self.option

context EditPSiGatePaymentMethod::effect()
  post:
  let pm: PSIgate = PSIgate.allInstances -> any(true) in
  pm.merchantID=self.newMerchantID and
  pm.mode=self.newMode and
  pm.type=self.newType and
  pm.creditCardCollection=self.newCreditCardCollection and
  pm.status=self.status and
  pm.orderStatus=self.orderStatus and
  pm.taxZone=self.taxZone

context EditReview::effect()
  post:
  self.review.review = self.newReview and
  self.review.rating = self.newRating and
  self.review.language = self.newLanguage and
  self.review.product = self.newProduct and
  self.review.customer = self.newCustomer

context EditSECPaymentMethod::effect()
  post:
  let pm: SECPay = SECPay.allInstances -> any(true) in
  pm.merchantID=self.newMerchantID and
  pm.mode=self.newMode and
  pm.status=self.status and
  pm.orderStatus=self.orderStatus and
  pm.taxZone=self.taxZone

context EditSpecial::effect()
  post:
  self.special.specialPrice = self.newSpecialPrice and
  self.special.expiryDate = self.newExpiryDate and
  self.special.specialStatus = self.newStatus

context EditTableRateShippingMethod::effect()
  post:
  let sm: TableRate = TableRate.allInstances -> any(true) in
  sm.items=self.newItems and
  sm.method=self.newMethod and
  sm.handlingFee=self.handlingFee and
  sm.taxZone=self.taxZone and
  sm.taxClass=self.taxClass and
  sm.status = self.status

context EditTaxClass::effect()
  post:
  self.taxClass.name = self.newName and
  self.taxClass.description = self.newDescription

context EditTaxRate::effect()
  post:
  self.taxRate.rate = self.newRate and
  self.taxRate.priority = self.newPriority and
  self.taxRate.description = self.newDescription and
  self.taxRate.taxClass = self.newTaxClass and
  self.taxRate.taxZone = self.newTaxZone

context EditTaxZone::effect()
  post:
CSTL and its application to the osCommerce case study.
Albert Tort

self.taxZone.name = self.newName and
self.taxZone.description = self.newDescription and
self.taxZone.zone = self.newZones

class EditTwoCheckOutPaymentMethod::effect()
  post:
    let pm: TwoCheckOut = TwoCheckOut.allInstances -> any(true)
in
      pm.login = self.newLogin and
      pm.mode = self.newMode and
      pm.merchantNotification = self.newMerchantNotification and
      pm.status = self.status and
      pm.orderStatus = self.orderStatus and
      pm.taxZone = self.taxZone

class EditUSPostalServiceShippingMethod::effect()
  post:
    let sm: USPostalService = USPostalService.allInstances -> any(true) in
    sm.userID = self.newUserID and
    sm.password = self.newPassword and
    sm.server = self.newServer and
    sm.handlingFee = self.handlingFee and
    sm.taxZone = self.taxZone and
    sm.taxClass = self.taxClass and
    sm.status = self.status

class EditZone::effect()
  post:
    self.zone.name = self.newName and
    self.zone.code = self.newCode

class EditZoneRatesShippingMethod::effect()
  post:
    let sm: ZoneRates = ZoneRates.allInstances -> any(true) in
    sm.items = self.newItems and
    sm.country = self.country and
    sm.taxClass = self.taxClass and
    sm.status = self.status

class EMailAddressChange::effect()
  post:
    myStore().eMailAddress = self.newEmailAddress

class EMailAddressMinimumChange::effect()
  post:
    MinimumValues.allInstances -> any(true).eMailAddress = self.newMinimum

class EMailFromChange::effect()
  post:
    myStore().eMailFrom = self.newEmailFrom

class EnableDownloadConfigurationChange::effect()
  post:
    Download.allInstances -> any(true).enableDownload = self.newValue

class ExpectedSortFieldChange::effect()
  post:
    myStore().expectedSortField = self.newExpectedSortField

class ExpectedSortOrderChange::effect()
  post:
    myStore().expectedSortOrder = self.newExpectedSortOrder

class FirstNameMinimumChange::effect()
  post:
    MinimumValues.allInstances -> any(true).firstName = self.newMinimum

class GenderCustomerDetailChange::effect()
  post:
    CustomerDetails.allInstances -> any(true).gender = self.newValue

class IncrementAndSignAttributeChange::effect()
  post:
    self.productAttribute.increment = self.newIncrement and
    self.productAttribute.sign = self.newSign

class InstallAuthorizeNetPaymentMethod::effect()
  post:
    (AuthorizeNet.allInstances - AuthorizeNet.allInstances@pre)
CSTL and its application to the osCommerce case study.
Albert Tort

-> forAll(pm:AuthorizeNet | pm.oclIsNew() and pm.oclIsTypeOf(AuthorizeNet) and pm.status=#enabled)

context InstallCashOnDeliveryPaymentMethod::effect()
post :
(CashOnDelivery.allInstances - CashOnDelivery.allInstances@pre) -> forAll(pm:CashOnDelivery | pm.oclIsNew() and pm.oclIsTypeOf(CashOnDelivery) and pm.status=#enabled)

context InstallCheckMoneyPaymentMethod::effect()
post :
(CheckMoney.allInstances - CheckMoney.allInstances@pre) -> forAll(pm:CheckMoney | pm.oclIsNew() and pm.oclIsTypeOf(CheckMoney) and pm.status=#enabled)

context InstallCreditCardPaymentMethod::effect()
post :
(CreditCard.allInstances - CreditCard.allInstances@pre) -> forAll(pm:CreditCard | pm.oclIsNew() and pm.oclIsTypeOf(CreditCard) and pm.status=#enabled)

context InstallFlatRateShippingMethod::effect()
post :
(FlatRate.allInstances - FlatRate.allInstances@pre) -> forAll(sm:FlatRate | sm.oclIsNew() and sm.oclIsTypeOf(FlatRate) and sm.status=#enabled)

context InstallIPaymentPaymentMethod::effect()
post :
(IPayment.allInstances - IPayment.allInstances@pre) -> forAll(pm:IPayment | pm.oclIsNew() and pm.oclIsTypeOf(IPayment) and pm.status=#enabled)

context InstallNochexPaymentMethod::effect()
post :
(Nochex.allInstances - Nochex.allInstances@pre) -> forAll(pm:Nochex | pm.oclIsNew() and pm.oclIsTypeOf(Nochex) and pm.status=#enabled)

context InstallPayPalPaymentMethod::effect()
post :
(PayPal.allInstances - PayPal.allInstances@pre) -> forAll(pm:PayPal | pm.oclIsNew() and pm.oclIsTypeOf(PayPal) and pm.status=#enabled)

context InstallPerItemShippingMethod::effect()
post :
(PerItem.allInstances - PerItem.allInstances@pre) -> forAll(sm:PerItem | sm.oclIsNew() and sm.oclIsTypeOf(PerItem) and sm.status=#enabled)

context InstallPSiGatePaymentMethod::effect()
post :
(PSiGate.allInstances - PSiGate.allInstances@pre) -> forAll(pm:PSiGate | pm.oclIsNew() and pm.oclIsTypeOf(PSiGate) and pm.status=#enabled)

context InstallSECPaymentMethod::effect()
post :
(SECPay.allInstances - SECPay.allInstances@pre) -> forAll(pm:SECPay | pm.oclIsNew() and pm.oclIsTypeOf(SECPay) and pm.status=#enabled)

context InstallTableRateShippingMethod::effect()
post :
(TableRate.allInstances - TableRate.allInstances@pre) -> forAll(sm:TableRate | sm.oclIsNew() and sm.oclIsTypeOf(TableRate) and sm.status=#enabled)

context InstallTwoCheckOutPaymentMethod::effect()
post :
(TwoCheckOut.allInstances - TwoCheckOut.allInstances@pre) -> forAll(pm:TwoCheckOut | pm.oclIsNew() and pm.oclIsTypeOf(TwoCheckOut) and pm.status=#enabled)

context InstallUSPostalServiceShippingMethod::effect()
post :
(USPostalService.allInstances - USPostalService.allInstances@pre) -> forAll(sm:USPostalService | sm.oclIsNew() and sm.oclIsTypeOf(USPostalService) and sm.status=#enabled)
CSTL and its application to the osCommerce case study.
Albert Tort

context InstallZoneRatesShippingMethod::effect()
  post : (ZoneRates.allInstances - ZoneRates.allInstances@pre) -> forAll(sm:ZoneRates | sm.ocIsNew() and sm.ocIsTypeOf(ZoneRates) and sm.status=#enabled)

context LastNameMinimumChange::effect()
  post : MinimumValues.allInstances->any(true).lastName = self.newMinimum

context LinkProduct::effect()
  post: self.product.category -> includes(self.newCategory)

context LockNewsletter::effect()
  post : self.newsletter.status = #locked

context LogIn::effect()
  post : self.session.customer = self.customer
  post : self.customer.numberOfLogons = self.customer.numberOfLogons@pre + 1
  post: if self.customer.customerShoppingCart->size()>0 then
    (RestorePreviousShoppingCart.allInstances - RestorePreviousShoppingCart.allInstances@pre) -> forAll(rpsc:RestorePreviousShoppingCart | rpsc.ocIsNew() and rpsc.ocIsTypeOf(RestorePreviousShoppingCart) and rpsc.customer=self.customer and rpsc.session=self.session)
  else
    if self.session.shoppingCart->notEmpty() then
      (CustomerShoppingCart.allInstances - CustomerShoppingCart.allInstances@pre) -> one(csc:CustomerShoppingCart | csc.ocIsNew() and csc.ocIsTypeOf(CustomerShoppingCart) and csc.shoppingCartItem = self.session.shoppingCart.shoppingCartItem and csc.customer=self.customer and self.session.shoppingCart=csc)
    else true
  endif
endif

context LogOut::effect()
  post : self.session.customer -> isEmpty()

context NameChange::effect()
  post : self.myStore().name = self.newName

context MaximumNumberDownloadConfigurationChange::effect()
  post : Download.allInstances->any(true).maximumNumberOfDownloads= self.newMaximum

context MaximumPackageWeightShippingConfigurationChange::effect()
  post : ShippingAndPackaging.allInstances->any(true).maximumPackageWeight = self.newMaximum

context MoveCategory::effect()
  post : self.category.parent = self.newParent

context MoveProduct::effect()
  post: self.product.category -> includes(self.newCategory) and self.product.category -> excludes(self.oldCategory)

context NewBanner::effect()
  post : (Banner.allInstances - Banner.allInstances@pre) -> forAll(b:Banner | b.ocIsNew() and b.ocIsTypeOf(Banner) and b.title = self.title and b.url = self.url and b.imagePath = self.imagePath and b.html = self.html and b.locale = self.locale and b.translated = self.translated and b.image = self.image and b.isPublished = self.isPublished)
CSTL and its application to the osCommerce case study.
Albert Tort

b.expires = self.expires and
b.scheduled = self.scheduled and
b.status = #enabled)

context NewBannerGroup::effect()
post :
(BannerGroup.allInstances - BannerGroup.allInstances@pre) -> forAll(bg:BannerGroup |
b.oclIsNew() and
bg.oclIsTypeOf(BannerGroup) and
bg.name = self.name)

context NewCategory::effect()
post :
(Category.allInstances - Category.allInstances@pre) -> forAll(c:Category |
c.oclIsNew() and
c.oclIsTypeOf(Category) and
c.imagePath = self.imagePath and
c.sortOrder = self.sortOrder and
c.parent = self.parent and
Language.allInstances ->
forall (l:Language) self.hasNewName -> select{languageOfCategory=l}->any(true).name =
c.hasCategoryName->select{language=l} ->any(true).categoryName)

context NewCountry::effect()
post :
(Country.allInstances - Country.allInstances@pre) -> forAll(c:Country |
c.oclIsNew() and
c.oclIsTypeOf(Country) and
c.name = self.name and
c.isoCode2 = self.isoCode2 and
c.isoCode3 = self.isoCode3)

context NewCurrency::effect()
post :
(Currency.allInstances - Currency.allInstances@pre) -> forAll(c:Currency |
c.oclIsNew() and
c.oclIsTypeOf(Currency) and
c.title = self.title and
c.code = self.code and
c.symbolLeft = self.symbolLeft and
c.symbolRight = self.symbolRight and
c.decimalPlaces = self.decimalPlaces and
c.value = self.value and
c.status = #enabled)

context NewCustomer::effect()
pre: not Customer.allInstances -> exists (c | c.eMailAddress = self.eMailAddress)
post :
(Customer.allInstances - Customer.allInstances@pre) -> forAll(c:Customer |
c.oclIsNew() and
c.oclIsTypeOf(Customer) and
c.gender = self.primary.gender and
c.firstName = self.primary.firstName and
c.lastName = self.primary.lastName and
c.dateOfBirth = self.dateOfBirth and
c.eMailAddress = self.eMailAddress and
c.phone = self.phone and
c.fax = self.fax and
c.newsletter = self.newsletter and
c.password = self.password and
c.numberOfLogons = 0 and
c.address = Set{primary} and
c.primary = primary)

context NewCustomerAddress::effect()
post :
Address.allInstances ->exists (a | a.gender = self.gender and
a.firstName = self.firstName and
da.lastName = self.lastName and
da.company = self.company and
CSTL and its application to the osCommerce case study.
Albert Tort

a.street = self.street and
a.street = self.suburb and
a.street = self.postCode and
a.street = self.city and
a.street = self.state and
a.street = self.country and
self.customer.address -> includes(a))

context NewDownloadableProductAttribute::effect()
post :
(Downloadable.allInstances - Downloadable.allInstances@pre) -> forAll(dpa:Downloadable | dpa.oclIsNew() and dpa.oclIsTypeOf(Downloadable) and dpa.increment = self.increment and dpa.sign = self.sign and dpa.filename = self.filename and dpa.product = self.product and dpa.attribute.option = self.option and dpa.attribute.value = self.value and if self.expiryDays.isDefined() then dpa.expiryDays = self.expiryDays else self.expiryDays = Download.allInstances->any(true).daysExpiryDelay endif and if self.maximumDownloadCount.isDefined() then dpa.maximumDownloadCount = self.maximumDownloadCount else self.maximumDownloadCount = Download.allInstances->any(true).maximumNumberOfDownloads endif)

context NewLanguage::effect()
post :
(Language.allInstances - Language.allInstances@pre) -> forAll(l:Language | l.oclIsNew() and l.oclIsTypeOf(Language) and l.name = self.newName and l.code = self.newCode and l.defaultCurrency = self.defaultCurrency)

context NewManufacturer::effect()
post :
(Manufacturer.allInstances - Manufacturer.allInstances@pre) -> forAll(m:Manufacturer | m.oclIsNew() and m.oclIsTypeOf(Manufacturer) and m.name = self.name and m.imagePath = self.imagePath and Language.allInstances -> forAll (l| self.hasURL -> select(languageOfURL=l).url = m.manufacturerInLanguage->select(language=l).url))

context NewNewsletter::effect()
post :
(Newsletter.allInstances - Newsletter.allInstances@pre) -> forAll(n:Newsletter | n.oclIsNew() and n.oclIsTypeOf(Newsletter) and n.title = self.title and n.content = self.content and n.status = #unlocked)

context NewOrderStatus::effect()
post :
(OrderStatus.allInstances - OrderStatus.allInstances@pre) -> forAll(os:OrderStatus | os.oclIsNew() and os.oclIsTypeOf(OrderStatus) and Language.allInstances-> forAll (l| self.hasOrderStatusName ->select(languageOfOrderStatus=l).orderStatusName.string = os.orderStatusInLanguage-> select(language=l).name) )
CSTL and its application to the osCommerce case study.
Albert Tort

context NewProduct::effect()
post : (Product.allInstances - Product.allInstances@pre) -> forAll(p:Product |
p.oclIsNew() and
p.oclIsTypeOf(Product) and
p.status = self.status and
p.available = self.available and
p.netPrice = self.netPrice and
p.quantityOnHand = self.quantityOnHand and
p.modelM = self.modelM and
p.imagePath = self.imagePath and
p.weight = self.weight and
p.category = self.category and
p.manufacturer = self.manufacturer and
p.taxClass = self.taxClass and
Language.allInstances -> forAll (l|
  self.hasNewProductName -> select(languageOfProduct=l).nameOfProduct.string =
  p.productInLanguage->select(language=l).name))

context NewProductAttribute::effect()
post : (ProductAttribute.allInstances - ProductAttribute.allInstances@pre) -> forAll(pa:ProductAttribute |
  pa.oclIsNew() and
  pa.oclIsTypeOf(ProductAttribute) and
  pa.increment = self.increment and
  pa.sign = self.sign and
  pa.product = self.product and
  pa.attribute.option = self.option and
  pa.attribute.value = self.value)

context NewProductNotification::effect()
post : (ProductNotification.allInstances - ProductNotification.allInstances@pre) -> forAll(n:ProductNotification |
  n.oclIsNew() and
  n.oclIsTypeOf(ProductNotification) and
  n.title = self.title and
  n.content = self.content and
  n.global = self.global and
  n.explicitNotifications = self.explicitNotifications and
  n.status = #unlocked)

context NewProductSubscription::effect()
post : self.customer.explicitNotifications -> includes(self.newSubscribedProduct)

context NewProductOption::effect()
post : (Option.allInstances - Option.allInstances@pre) -> forAll(po:Option |
  po.oclIsNew() and
  po.oclIsTypeOf(Option) and
  Language.allInstances -> forAll (l| self.hasNewOptionName -> select(languageOfOption=l).nameOfOption =
  po.hasOptionName->select(optionLanguage=l).optionName))

context NewProductOptionValue::effect()
post : (Value.allInstances - Value.allInstances@pre) -> forAll(pov:Value |
  pov.oclIsNew() and
  pov.oclIsTypeOf(Value) and
  Language.allInstances -> forAll (l| self.hasNewValueName -> select(languageOfValue=l).nameOfValue =
  pov.hasValueName->select(valueLanguage=l).valueName) and
  pov.option = self.option)

context NewReview::effect()
post : (Review.allInstances - Review.allInstances@pre) -> forAll(r:Review |
  r.oclIsNew() and
  r.oclIsTypeOf(Review) and
CSTL and its application to the osCommerce case study.
Albert Tort

r.review = self.review and
r.rating = self.rating and
r.customer = self.customer and
r.product = self.product and
r.language = self.language)

context NewSession::effect()
post :
(Session.allInstances - Session.allInstances@pre) -> forAll(s:Session |
s.oclIsNew() and
s.oclIsTypeOf(Session) and
s.currentCurrency = self.currentCurrency and
s.currentLanguage = self.currentLanguage and
s.sessionID = Session.allInstances->size())

context NewSpecial::effect()
post :
self.product.oclIsTypeOf(Special) and
self.product.oclAsType(Special).specialPrice = self.specialPrice and
self.product.oclAsType(Special).expiryDate = self.expiryDate and
self.product.oclAsType(Special).specialStatus = self.status

context NewTaxZone::effect()
post :
(TaxZone.allInstances - TaxZone.allInstances@pre) -> forAll(tz:TaxZone |
tz.oclIsNew() and
tz.oclIsTypeOf(TaxZone) and
tz.name = self.name and
tz.description = self.description and
tz.zone = self.zone)

context NewTaxRate::effect()
post :
(TaxRate.allInstances - TaxRate.allInstances@pre) -> forAll(tr:TaxRate |
tr.oclIsNew() and
tr.oclIsTypeOf(TaxRate) and
tr.rate = self.rate and
tr.priority = self.priority and
tr.description = self.description and
tr.taxClass = self.taxClass and
tr.taxZone = self.taxZone)

context NewTaxClass::effect()
post :
(TaxClass.allInstances - TaxClass.allInstances@pre) -> forAll(tc:TaxClass |
tc.oclIsNew() and
tc.oclIsTypeOf(TaxClass) and
tc.name = self.name and
tc.description = self.description)

context NewZone::effect()
post :
(Zone.allInstances - Zone.allInstances@pre) -> forAll(z:Zone |
z.oclIsNew() and
z.oclIsTypeOf(Zone) and
z.name = self.name and
z.code = self.code and
z.country = self.country)

context OrderConfirmation::effect()
post theOrderIsCreated:
(Order.allInstances - Order.allInstances@pre) -> forAll(o:Order |
o.oclIsNew() and
o.oclIsTypeOf(Order) and
self.orderCreated = o and
o.customer = self.shoppingCart@pre.customer@pre and
o.billing = self.billing and
o.delivery = self.delivery and
o.shippingMethod = self.shippingMethod and
CSTL and its application to the osCommerce case study.
Albert Tort

```cstl

o.paymentMethod = self.paymentMethod and
o.currency = self.currency and
--The initial status of the order is pending
(OrderStatusChange.allInstances - OrderStatusChange.allInstances@pre)
-> forAll(osc:OrderStatusChange |
osc.oclIsNew() and
osc.oclIsTypeOf(OrderStatusChange) and
osc.comments = self.comments and
osc.orderStatus = Store.allInstances -> any(true).defaultStatus and
osc.order = o and
--There is an order line for each shopping cart item
shoppingCart@pre.shoppingCartItem@pre->forall
  {i|OrderLine.allInstances -> one
   (ol|ol.order = i and
    ol.product = i.product@pre and
    ol.quantity = i.quantity@pre and
    i.attribute@pre->forall
     (iAtt|OrderLineAttribute.allInstances -> exists
      {olAtt|olAtt.orderLine = ol and
       olAtt.attribute = iAtt})))
post theShoppingCartIsRemoved:
  ShoppingCart.allInstances->excludes(self.shoppingCart@pre)
post updateProductQuantities:
  let productsBought:Set(Product) =
    self.shoppingCart@pre.shoppingCartItem@pre.product@pre->asSet()
in productsBought ->forall (p |
  let quantityBought:Integer =
    self.shoppingCart@pre.shoppingCartItem@pre->forall
     (sc | sc.product = p).quantity -> sum()
in
  p.quantityOrdered = p.quantityOrdered@pre + quantityBought and
  Stock.allInstances->any(true).subtractStock implies
  p.quantityOnHand = p.quantityOnHand@pre - quantityBought)

context OwnerChange::effect()
post : myStore().owner = self.newOwner

context PasswordChange::effect()
post : self.customer.password = self.newPassword

context PasswordMinimumChange::effect()
post : MinimumValues.allInstances->any(true).password = self.newMinimum

context PercentageIncreaseForLargerPackagesShippingConfigurationChange::effect()
post : ShippingAndPackaging.allInstances
  ->any(true).percentageIncreaseForLargerPackages= self.newPercentage

context PostCodeMinimumChange::effect()
post : MinimumValues.allInstances->any(true).postCode = self.newMinimum

context PostCodeShippingConfigurationChange::effect()
post : ShippingAndPackaging.allInstances->any(true).postCode = self.newPostCode

context PrimaryCustomerAddressChange::effect()
post : self.customer.primary = self.address

context ProductAttributeStatusChange::effect()
post : self.productAttribute.status = self.newStatus

context ProductDownload::effect()
post :
  let OrderDownloadFromProduct:OrderDownload =
    self.customer.order.orderLine.orderLineAttribute
    ->forall (ola | ola.oclIsTypeOf(OrderDownload) and
    ola.orderLine.product = self.product)
  asSequence() -> last() .oclAsType(OrderDownload)
  in
  let OldOrderDownloadCount:Integer =
    self.customer.order.orderLine.orderLineAttribute@pre
```

251
CSTL and its application to the osCommerce case study.
Albert Tort

-> select (ola | ola.oclIsTypeOf(OrderDownload) and
  ola.orderLine.product = self.product) -> asSequence() -> last()
  .oclAsType(OrderDownload).downloadCount

in
  OrderDownloadFromProduct.downloadCount = OldOrderDownloadCount + 1

context  ProductOptionAttributeChange::effect()
  post : productAttribute.attribute.option = self.option

context  ProductValueAttributeChange::effect()
  post : productAttribute.attribute.value = self.value

context  ProductStatusChange::effect()
  post : self.product.status = self.newStatus

context  ReadProductInfo::effect()
  post : self.product.productInLanguage->select(pil | pil.language=self.language)
    ->any(true).viewed =
    self.product@pre.productInLanguage@pre->select(pil | pil.language=self.language)->any(true).viewed + 1

context  ReadReview::effect()
  post : self.review.timesRead = self.review@pre.timesRead + 1

context  ReorderLevelStockConfigurationChange::effect()
  post : Stock.allInstances->any(true).stockReOrderLevel = self.newValue

context  RestorePreviousShoppingCart::effect()
  post : self.session.shoppingCart = self.customer.customerShoppingCart

context  ReviewTextMinimumChange::effect()
  post : MinimumValues.allInstances->any(true).reviewText = self.newMinimum

context  SendExtraOrderEmailChange::effect()
  post : myStore().sendExtraOrderEMail->includesAll(self.newSendExtraOrderEMail)

context  SendNewsletter::effect()
  post : true

context  SetCancelledOrderStatus::effect()
  post : self.myStore().cancelledStatus = self.orderStatus

context  SetCurrentCurrency::effect()
  post : self.session.currentCurrency = self.newCurrentCurrency

context  SetCurrentLanguage::effect()
  post : self.session.currentLanguage = self.newCurrentLanguage
  post :
    Store.allInstances -> any(true).switchToDefaultLanguageCurrency and
    self.newCurrentLanguage.defaultCurrency -> notEmpty()
  implies (
    (SetCurrentCurrency.allInstances = SetCurrentCurrency.allInstances@pre)
    -> forAll(ccc: SetCurrentCurrency | ccc.oclIsNew() and
      ccc.oclIsTypeOf(SetCurrentCurrency) and
      ccc.session = self.session and
      ccc.newCurrentCurrency = self.newCurrentLanguage.defaultCurrency)

context  SetDefaultCurrency::effect()
  post : Store.allInstances -> any(true).defaultCurrency = self.currency

context  SetDefaultLanguage::effect()
  post : Store.allInstances -> any(true).defaultLanguage = self.language

context  SetDefaultOrderStatus::effect()
  post : self.myStore().defaultStatus = self.orderStatus
CSTL and its application to the osCommerce case study.
Albert Tort

class ShowBanner::effect()
post : 
   BannerHistory.allInstances -> one
   (bh | bh.banner = self.banner and
    bh.shown = bh@pre.shown + 1)

class StateCustomerDetailChange::effect()
post : StateCustomerDetails.allInstances->any(true).state = self.newValue

class StateMinimumChange::effect()
post : StateMinimumValues.allInstances->any(true).state = self.newMinimum

class StatusPaymentMethodChange::effect()
post : self.paymentMethod.status = self.newStatus

class StatusShippingMethodChange::effect()
post : self.shippingMethod.status = self.newStatus

class StoreAddressAndPhoneChange::effect()
post : myStore().storeAddressAndPhone = self.newStoreAddressAndPhone

class StreetAddressMinimumChange::effect()
post : MinimumValues.allInstances->any(true).streetAddress = self.newMinimum

class SubstractStockConfigurationChange::effect()
post : Stock.allInstances->any(true).substractStock = self.newValue

class SuburbCustomerDetailChange::effect()
post : CustomerDetails.allInstances->any(true).suburb = self.newValue

class SwitchToDefaultLanguageCurrencyChange::effect()
post : myStore().switchToDefaultLanguageCurrency = self.switchToDefaultLanguageCurrency

class TaxDecimalPlacesChange::effect()
post : myStore().taxDecimalPlaces = self.newTaxDecimalPlaces

class TelephoneMinimumChange::effect()
post : MinimumValues.allInstances->any(true).telephoneNumber = self.newMinimum

class TypicalPackageTareWeightShippingConfigurationChange::effect()
post : ShippingAndPackaging.allInstances->any(true).typicalPackageTareWeight = self.newValue

class UninstallAuthorizeNetPaymentMethod::effect()
post : AuthorizeNet.allInstances@pre->any(true).oclIsKindOf(OclAny)

class UninstallCashOnDeliveryPaymentMethod::effect()
post : CashOnDelivery.allInstances@pre->any(true).oclIsKindOf(OclAny)

class UninstallCheckMoneyPaymentMethod::effect()
post : CheckMoney.allInstances@pre->any(true).oclIsKindOf(OclAny)

class UninstallCreditCardPaymentMethod::effect()
post : CreditCard.allInstances@pre->any(true).oclIsKindOf(OclAny)

class UninstallFlatRateShippingMethod::effect()
post : FlatRate.allInstances@pre->any(true).oclIsKindOf(OclAny)

class UninstallIPaymentPaymentMethod::effect()
post : IPayment.allInstances@pre->any(true).oclIsKindOf(OclAny)

class UninstallNochexPaymentMethod::effect()
post : Nochex.allInstances@pre->any(true).oclIsKindOf(OclAny)
CSTL and its application to the osCommerce case study.
Albert Tort

context UninstallPayPalPaymentMethod::effect()
post :
PayPal.allInstances@pre->any(true).oclIsKindOf(OclAny)

context UninstallPerItemShippingMethod::effect()
post :
PerItem.allInstances@pre->any(true).oclIsKindOf(OclAny)

context UninstallPSiGatePaymentMethod::effect()
post :
PSiGate.allInstances@pre->any(true).oclIsKindOf(OclAny)

context UninstallSECPaymentMethod::effect()
post :
SECPay.allInstances@pre->any(true).oclIsKindOf(OclAny)

context UninstallTableRateShippingMethod::effect()
post :
TableRate.allInstances@pre->any(true).oclIsKindOf(OclAny)

context UninstallTwoCheckOutPaymentMethod::effect()
post :
TwoCheckOut.allInstances@pre->any(true).oclIsKindOf(OclAny)

context UninstallUSPostalServiceShippingMethod::effect()
post :
USPostalService.allInstances@pre->any(true).oclIsKindOf(OclAny)

context UninstallZoneRatesShippingMethod::effect()
post :
ZoneRates.allInstances@pre->any(true).oclIsKindOf(OclAny)

context UnlockNewsletter::effect()
post : self.newsletter.status = #unlocked

context UpdateCurrencyValueChange::effect()
post : self.currency.value = self.newValue

context UpdateOrderStatus::effect()
post :
(OrderStatusChange.allInstances - OrderStatusChange.allInstances@pre)
-> forAll(osc:OrderStatusChange |
osc.oclIsNew() and
osc.oclIsTypeOf(OrderStatusChange) and
osc.comments = self.comments and
osc.order = self.order and
osc.orderStatus = self.newOrderStatus)

context ZoneChange::effect()
post : myStore().zone = self.newZone

context RemoveProduct::effect()
post : not self.shoppingCartItem@pre.oclIsKindOf(OclAny)

context ChangeQuantity::effect()
post : self.shoppingCartItem.quantity = self.quantity

context UpdateShoppingCart::effect()
post :
self.lineChange ->forAll
{lc|let cartItem:ShoppingCartItem = 
self.session.shoppingCart.shoppingCartItem->at(lc.index)

in
{lc.remove or lc.quantity <> cartItem.quantity}
implies
if lc.remove then
(RemoveProduct.allInstances
- RemoveProduct.allInstances@pre)
-> forAll(rp:RemoveProduct |
rp.oclIsNew and

254
CSTL and its application to the osCommerce case study.
Albert Tort

rp.oclIsTypeOf(RemoveProduct) and
rp.shoppingCartItem = cartItem)
else
   (ChangeQuantity.allInstances -
   ChangeQuantity.allInstances@pre)
   -> forAll(cq:ChangeQuantity |
   cq.oclIsNew() and
   cq.oclIsTypeOf(ChangeQuantity) and
   cq.shoppingCartItem = cartItem and
   cq.quantity = lc.quantity)
endif}
Appendix B: Example CSTL methods of the case study

```cstl
method NameChange{
    self.myStore().name := self.newName;
}

method CountryChange{
    self.myStore().country := self.newCountry;
}

method InstallCreditCardPaymentMethod{
    cc:=new CreditCard;
    cc.status:=#enabled;
}

method InstallCashOnDeliveryPaymentMethod{
    cd:=new CashOnDelivery;
    cd.status:=#enabled;
}

method UninstallCreditCardPaymentMethod{
    delete CreditCard.allInstances->any(true);
}

method InstallPerItemShippingMethod{
    pi:=new PerItem;
    pi.status:=#enabled;
}

method InstallFlatRateShippingMethod{
    fr:=new FlatRate;
    fr.status:=#enabled;
}

method UninstallPerItemShippingMethod{
    delete PerItem.allInstances->any(true);
}

method NewLanguage{
    l:=new Language;
    l.name:=self.newName;
    l.code:=self.newCode;
    l.defaultCurrency:=self.defaultCurrency;
}

method EditLanguage{
    self.language.name:=self.newName;
    self.language.code:=self.newCode;
    self.language.defaultCurrency:=self.newDefaultCurrency;
}

method OrderConfirmation{
    //The order is created
    o:=new Order;
    o.customer := self.shoppingCart.customer;
    o.billing:=self.billing;
    o.delivery:=self.delivery;
    o.shippingMethod := self.shippingMethod;
    o.paymentMethod := self.paymentMethod;
    o.currency := self.currency;

    //The initial status of an order is pending
    OrderStatus os:=Store.allInstances->any(true).defaultStatus;
    osc:=new OrderStatusChange(order:=o, orderStatus:=os);
    osc.comments := self.comments;

    //There is an order line for each shopping cart item
    Integer index:=0;
```
CSTL and its application to the osCommerce case study.
Albert Tort

Integer indexat:=0;
while self.shoppingCart.shoppingCartItem->size()>index do
  i := self.shoppingCart.shoppingCartItem->at(index+1);
  ol:=new OrderLine;
  ol.order:=o;
  ol.product:=i.product;
  ol.quantity:=i.quantity;
  while i.attribute->size()>indexat do
    attr:=i.attribute->asSequence()]->at(indexat+1);
    ola:=new OrderLineAttribute;
    ola.orderLine:=ol;
    ola.attribute:=attr;
    indexat:=indexat+1;
  endwhile
  index:=index+1;
  indexat:=0;
endwhile

//update product quantities
products:=o.orderLine.product->asSet();
Integer i:=0;
while products->size()>i do
  p:=products->asSequence()]->at(i+1);
  substract:= Stock.allInstances->any(true).subtractStock;
  if substract then
    var:=o.orderLine->select(product=p).quantity->sum();
  endif
  i:=i+1;
endwhile

//The shopping cart is removed
delete self.shoppingCart;

} method PasswordMinimumChange{
    MinimumValues.allInstances->any(true).password := self.newMinimum;
}

} method CreditCardNumberMinimumChange{
    MinimumValues.allInstances->any(true).creditCardNumber := self.newMinimum;
}

} method AddressBookEntriesMaximumChange{
    MaximumValues.allInstances->any(true).addressBookEntries := self.newMaximum;
}

} method GenderCustomerDetailChange{
    CustomerDetails.allInstances->any(true).gender := self.newValue;
}

} method MaximumNumberDownloadConfigurationChange{
    Download.allInstances->any(true).maximumNumberOfDownloads := self.newMaximum;
}

} method CheckLevelStockConfigurationChange{
    Stock.allInstances->any(true).checkStockLevel := self.newValue;
}

} method TypicalPackageTareWeightShippingConfigurationChange{
    ShippingAndPackaging.allInstances->any(true).typicalPackageTareWeight := self.newValue;
}

} method MaximumPackageWeightShippingConfigurationChange{
    ShippingAndPackaging.allInstances->any(true).maximumPackageWeight := self.newMaximum;
}
CSTL and its application to the osCommerce case study.
Albert Tort

```plaintext
method StatusPaymentMethodChange{
    self.paymentMethod.status:=self.newStatus;
}

method EditCreditCardPaymentMethod{
    CreditCard.allInstances->any(true).splitCreditCardToMail :=
        self.newSplitCreditCardToMail;
    CreditCard.allInstances->any(true).status := self.status;
    CreditCard.allInstances->any(true).orderStatus := self.orderStatus;
    CreditCard.allInstances->any(true).taxZone := self.taxZone;
}

method EditPerItemShippingMethod{
    PerItem.allInstances->any(true).cost := self.newCost;
    PerItem.allInstances->any(true).handlingFee := self.handlingFee;
    PerItem.allInstances->any(true).taxZone := self.taxZone;
    PerItem.allInstances->any(true).taxClass := self.taxClass;
    PerItem.allInstances->any(true).status := self.status;
}

method SetDefaultLanguage{
    Store.allInstances->any(true).defaultLanguage := self.language;
}

method DeleteLanguage{
    delete self.language;
}

method NewCurrency{
    c:= new Currency;
    c.title:=self.title;
    c.code:=self.code;
    c.symbolLeft:=self.symbolLeft;
    c.symbolRight:=self.symbolRight;
    c.decimalPlaces:=self.decimalPlaces;
    c.value:=self.value;
    c.status:=#enabled;
}

method EditCurrency{
    self.currency.title:=self.newTitle;
    self.currency.code:=self.newCode;
    self.currency.symbolLeft:=self.newSymbolLeft;
    self.currency.symbolRight:=self.newSymbolRight;
    self.currency.decimalPlaces:=self.newDecimalPlaces;
    self.currency.value:=self.newValue;
}

method DeleteCurrency{
    delete self.currency;
}

method SetDefaultCurrency{
    Store.allInstances->any(true).defaultCurrency := self.currency;
}

method CurrencyStatusChange{
    self.currency.status := self.newStatus;
}

method NewCountry{
    c:=new Country;
    c.name:=self.name;
    c.isoCode2:=self.isoCode2;
    c.isoCode3:=self.isoCode3;
}

method EditCountry{
    self.country.name:=self.newName;
    self.country.isoCode2:=self.newIsoCode2;
    self.country.isoCode3:=self.newIsoCode3;
}
```
CSTL and its application to the osCommerce case study.
Albert Tort

```plaintext
method DeleteCountry{
    Integer i:=0;
    while self.country.zone->size()>i do
        z:=self.country.zone->asSequence()->at(i+1);
        delete z;
        endwhile
    delete self.country;
}

method NewZone{
    z:=new Zone;
    z.name:=self.name;
    z.code:=self.code;
    z.country:=self.country;
}

method EditZone{
    self.zone.name:=self.newName;
    self.zone.code:=self.newCode;
}

method DeleteZone{
    delete self.zone;
}

method NewTaxZone{
    tz := new TaxZone;
    tz.name := self.name;
    tz.description := self.description;
    tz.zone := self.zone;
}

method EditTaxZone{
    self.taxZone.name := self.newName;
    self.taxZone.description := self.newDescription;
    self.taxZone.zone := self.newZones;
}

method DeleteTaxZone{
    delete self.taxZone;
}

method NewTaxClass{
    tc := new TaxClass;
    tc.name := self.name;
    tc.description := tc.description;
}

method EditTaxClass{
    self.taxClass.name := self.newName;
    self.taxClass.description := self.newDescription;
}

method DeleteTaxClass{
    delete self.taxClass;
}

method NewTaxRate{
    tc:=self.taxClass;
    tz:=self.taxZone;
    tr := new TaxRate(taxClass:=tc, taxZone:=tz);
    tr.rate:=self.rate;
    tr.priority:=self.priority;
    tr.description:=self.description;
}

method EditTaxRate{
    tc:=self.newTaxClass;
    tz:=self.newTaxZone;
    tr := new TaxRate(taxClass:=tc, taxZone:=tz);
    tr.rate:=self.newRate;
}
```
CSTL and its application to the osCommerce case study.
Albert Tort

```plaintext
tr.priority := self.newPriority;
tr.description := self.newDescription;
self.taxRate := tr;
}

method DeleteTaxRate{
delete self.taxRate;
}

method NewProduct{
p:= new Product;
p.status := self.status;
p.available := self.available;
p.netPrice := self.netPrice;
p.modelM := self.modelM;
p.imagePath := self.imagePath;
p.weight := self.weight;
p.category := self.category;
p.manufacturer := self.manufacturer;
p.taxClass := self.taxClass;
Integer index := 0;
while Language.allInstances->size()>index do
  l := Language.allInstances->asSequence()->at(index+1);
  hnpn := HasNewProductName.allInstances->select(languageOfProduct=l)
    ->select(productNameEvent=self)->any(true);
  pil := new ProductInLanguage(product:=p, language=l);
  pil.name := hnpn.nameOfProduct.string;
  index := index+1;
endwhile
}

method EditProduct{
self.product.status := self.status;
self.product.available := self.available;
self.product.netPrice := self.netPrice;
self.product.modelM := self.modelM;
self.product.imagePath := self.imagePath;
self.product.weight := self.weight;
self.product.category := self.category;
self.product.manufacturer := self.manufacturer;
self.product.taxClass := self.taxClass;
Integer i := 0;
while Language.allInstances->size()>i do
  l := Language.allInstances->asSequence()->at(i+1);
  hnpn := HasNewProductName.allInstances->select(languageOfProduct=l)
    ->select(productNameEvent=self)->any(true);
  pil := self.product.productInLanguage->any(language=l);
  pil.name := hnpn.nameOfProduct.string;
i := i+1;
endwhile
}

method DeleteProduct{
  if self.product.orderLine->size()=0
    then delete self.product;
  else
    new ProductStatusChange(product:=self.product, newStatus:=#outOfStock);
  endif
}

method ProductStatusChange{
self.product.status := self.newStatus;
}

method NewProductOption{
o := new Option;
Integer i := 0;
while Language.allInstances->size()>i do
```
```
CSTL and its application to the osCommerce case study.
Albert Tort

```java
1:=Language.allInstances->asSequence()->at(i+1);
hnnon:=HasNewOptionName.allInstances->select(languageOfOption=l)
  ->select(productOptionNameEvent=self)->any(true);
oname:=hnnon.nameOfOption;
pil:=new HasOptionName(option:=o,optionLanguage:=l,optionName:=oname);
i:=i+1;
endwhile

method EditProductOption{
  o:=self.option;
  Integer i:=0;
  while Language.allInstances->size()>i do
    l:=Language.allInstances->asSequence()->at(i+1);
    hnnon:=HasNewOptionName.allInstances->select(languageOfOption=l)
      ->select(productOptionNameEvent=self)->any(true);
    oname:=hnnon.nameOfOption;
    pil:=new HasOptionName(option:=o,optionLanguage:=l,optionName:=oname);
    hon:=o.hasOptionName->any(optionLanguage=l);
    delete hon;
    i:=i+1;
  endwhile
}

method DeleteProductOption{
  Integer i:=0;
  valuesNotUsedSize:=self.option.value->select(option->size()==1)
    ->select(attribute.orderLineAttribute->isEmpty())->size();
  while valuesNotUsedSize>i do
    v:=self.option.value->select(option->size()==1)
      ->select(attribute.orderLineAttribute->isEmpty())->asSequence()->at(i+1);
    delete v;
    i:=i+1;
  endwhile
  delete self.option;
}

method NewProductOptionValue{
  v:=new Value;
  Integer i:=0;
  while Language.allInstances->size()>i do
    l:=Language.allInstances->asSequence()->at(i+1);
    hnvn:=HasNewValueName.allInstances->select(languageOfValue=l)
      ->select(productValueNameEvent=self)->any(true);
    vname:=hnvn.nameOfValue;
    new HasValueName(value:=v,valueLanguage:=l,valueName:=vname);
    i:=i+1;
  endwhile
  v.option:=self.option;
}

method EditProductOptionValue{
  v:=self.value;
  Integer i:=0;
  while Language.allInstances->size()>i do
    l:=Language.allInstances->asSequence()->at(i+1);
    hnvn:=HasNewValueName.allInstances->select(languageOfValue=l)
      ->select(productValueNameEvent=self)->any(true);
    vname:=hnvn.nameOfValue;
    hon:=v.hasValueName->any(valueLanguage=l);
    pil:=new HasValueName(value:=v,valueLanguage:=l,valueName:=vname);
    delete hon;
    i:=i+1;
  endwhile
  v.option:=self.option;
}

method DeleteProductOptionValue{
  delete self.value;
}
```
CSTL and its application to the osCommerce case study.
Albert Tort

method NewProductAttribute{
  o:=self.option;
  v:=self.value;
  attr:=Attribute.allInstances->select(value=v)->any(option=o);
  pa:=new ProductAttribute(product:=self.product, attribute:=attr);
  pa.sign:=self.sign;
  pa.increment:=self.increment;
}

method AttributeChange{
  o:=self.newOption;
  v:=self.newValue;
  pa:=self.productAttribute;
  attr:=Attribute.allInstances->select(value=v)->any(option=o);
  increment:=pa.increment;
  sign:=pa.sign;
  status:=pa.status;
  product:=pa.product;
  npa:=new ProductAttribute(product:=product, attribute:=attr);
  self.productAttribute:=npa;
  delete pa;
}

method IncrementAndSignAttributeChange{
  pa:=self.productAttribute;
  pa.increment:=self.newIncrement;
  pa.sign:=self.newSign;
}

method DeleteProductAttribute{
  participantOrdersSize:=OrderLineAttribute.allInstances->select(attribute=self.productAttribute.attribute)->select(orderLine.product=self.productAttribute.product)->size();
  if participantOrdersSize=0 then
    delete self.productAttribute;
  else
    new ProductAttributeStatusChange(productAttribute:=self.productAttribute,
                                    newStatus:=#disabled);
  endif
}

method ProductAttributeStatusChange{
  self.productAttribute.status:=#disabled;
}

method NewSpecial{
  p:=self.product;
  s:=new Special;
  s.specialPrice:=self.specialPrice;
  s.expiryDate:=self.expiryDate;
  s.specialStatus:=self.status;
  s.status := p.status;
  s.available := p.available;
  s.netPrice:= p.netPrice;
  s.quantityOnHand := p.quantityOnHand;
  s.modelM:=p.modelM;
  s.imagePath:=p.imagePath;
  s.weight:=p.weight;
  s.category := p.category;
  s.manufacturer:=p.manufacturer;
  s.taxClass:=p.taxClass;
  Integer i:=0;
  while Language.allInstances->size()>=i do
    l:=Language.allInstances->asSequence()->at(i+1);
    hnpn:=HasProductName.allInstances->select(languageOfProduct=l)
    ->select(product=p)->any(true);
    pil:=new ProductInLanguage(product:=s,language:=l);
    pil.name:=hnpn.nameOfProduct.string;
    i:=i+1;
  endwhile
  self.product:=s;
}
CSTL and its application to the osCommerce case study.
Albert Tort

method EditSpecial{
   self.special.specialPrice:=self.newSpecialPrice;
   self.special.expiryDate:=self.newExpiryDate;
   self.special.specialStatus:=self.newStatus;
}

method DeleteSpecial{
   s:=self.special;
   p:=new Product;
   //We save the self.product information
   p.status := s.status;
   p.available := s.available;
   p.netPrice:= s.netPrice;
   p.quantityOnHand := s.quantityOnHand;
   p.modelM:=s.modelM;
   p.imagePath:=s.imagePath;
   p.weight:=s.weight;
   p.category := s.category;
   p.manufacturer:=s.manufacturer;
   p.taxClass:=s.taxClass;
   i:=0;
   while Language.allInstances->size()>i do
      l:=Language.allInstances->asSequence()->at(i+1);
      hnn:=HasProductName.allInstances->select(languageOfProduct=l)
               ->select (product=s)->any(true);
      pil:=new ProductInLanguage(product:=p,language:=l);
      pil.name:=hnn.nameOfProduct.string;
      i:=i+1;
   endwhile
   delete s;
}

method NewCategory{
   c:=new Category;
   c.imagePath:=self.imagePath;
   c.sortOrder:=self.sortOrder;
   c.parent:=self.parent;
   Integer i:=0;
   while Language.allInstances->size()>i do
      l:=Language.allInstances->asSequence()->at(i+1);
      hnn:=HasNewName.allInstances->select(languageOfCategory=l)
              ->select(categoryNameEvent=self)->any(true);
      cname:=hnn.name;
      ctn:=new HasCategoryName(category:=c,language:=l,categoryName:=cname);
      hcn:=c.hasCategoryName->any(language=l);
      delete hcn;
      i:=i+1;
   endwhile
}

method EditCategory{
   c:=self.category;
   c.imagePath:=self.imagePath;
   c.sortOrder:=self.sortOrder;
   c.parent:=self.newParent;
   Integer i:=0;
   while Language.allInstances->size()>i do
      l:=Language.allInstances->asSequence()->at(i+1);
      hnn:=HasNewName.allInstances->select(languageOfCategory=l)
              ->select(categoryNameEvent=self)->any(true);
      cname:=hnn.name;
      ctn:=new HasCategoryName(category:=c,language:=l,categoryName:=cname);
      hcn:=c.hasCategoryName->any(language=l);
      delete hcn;
      i:=i+1;
   endwhile
}

method MoveCategory{
   self.category.parent:=self.newParent;
}

method DeleteCategory{
   topCategory:=self.category;
   Integer i:=0;
CSTL and its application to the osCommerce case study.
Albert Tort

while self.allChilds(topCategory)->size()>i do
c:=self.allChilds(topCategory)->asSequence()->at(i+1);
delete c;
i:=i+1;
endwhile
delete self.category;
}

method MoveProduct{
   newCat:=self.newCategory;
   oldCat:=self.oldCategory;
categories:=self.product.category->union(Set{newCat})-Set{oldCat};
self.product.category:=categories;
}

method LinkProduct{
   newCat:=self.newCategory;
categories:=self.product.category->union(Set{newCat});
self.product.category:=categories;
}

method NewCustomer{
   c:=new Customer;
c.gender:=self.primary.gender;
c.firstName:=self.primary.firstName;
c.lastName:=self.primary.lastName;
c.dateOfBirth:=self.dateOfBirth;
c.eMailAddress:=self.eMailAddress;
c.phone:=self.phone;
c.fax:=self.fax;
c.newsletter:=self.newsletter;
c.password:=self.password;
c.numberOfLogons:=0;
primaryAddress:=self.primary;
c.address:=Set{primaryAddress};
c.primary:=primaryAddress;
}

method PasswordChange{
   self.customer.password:=self.newPassword;
}

method NewManufacturer{
   m:=new Manufacturer;
m.name := self.name;
m.imagePath := self.imagePath;
   Integer i:=0;
while Language.allInstances->size()>i do
   l:=Language.allInstances->asSequence()->at(i+1);
hurl:=HasURL.allInstances->select(languageOfURL=l)
   -->select(manufacturerURLEvent=self)->any(true);
mil:=new ManufacturerInLanguage(manufacturer:=m,language:=l);
mil.url:=hurl.url;
i:=i+1;
endwhile
}

method EditManufacturer{
   m:=self.manufacturer;
m.name := self.name;
m.imagePath := self.imagePath;
   Integer index:=0;
while Language.allInstances->size()>index do
   l:=Language.allInstances->asSequence()->at(index+1);
hurl:=HasURL.allInstances->select(languageOfURL=l)
   -->select(manufacturerURLEvent=self)->any(true);
mil:=m.manufacturerInLanguage->any(languages=1);
mil.url:=hurl.url;
   index:=index+1;
endwhile
}
CSTL and its application to the osCommerce case study.
Albert Tort

```vbnet
method DeleteManufacturer
  m:=self.manufacturer;
  deleteProducts:=self.deleteProds;
  Integer ip:=0;
  if deleteProducts then
    while m.product->size()>ip do
      p:=m.product->asSequence()->at(ip+1);
      p.status:=#outOfStock;
      ip:=ip+1;
    endwhile
  endif
  //Delete the manufacturer
  delete m;
}

method NewBannerGroup{
  bg:=new BannerGroup;
  bg.name:=self.name;
}

method EditBannerGroup{
  self.bannerGroup.name:=self.newName;
}

method NewBanner{
  b:=new Banner;
  b.title:=self.title;
  b.url:=self.url;
  b.imagePath:=self.imagePath;
  b.html:=self.html;
  b.expires:=self.expires;
  b.scheduled:=self.scheduled;
  b.status:=#enabled;
  b.bannerGroup:=self.bannerGroup;
}

method EditBanner{
  b:=self.banner;
  b.title:=self.newTitle;
  b.url:=self.newUrl;
  b.imagePath:=self.newImagePath;
  b.html:=self.newHtml;
  b.expires:=self.newExpires;
  b.scheduled:=self.newScheduled;
  b.status:=self.newStatus;
  b.bannerGroup:=self.newBannerGroup;
}

method DeleteBanner{
  delete self.banner;
}

method DeleteBannerGroup{
  delete self.bannerGroup;
}

method NewNewsletter{
  n:=new Newsletter;
  n.title:=self.title;
  n.content:=self.content;
  n.status:=#unlocked;
}

method NewProductNotification{
  n:=new ProductNotification;
  n.title:=self.title;
  n.content:=self.content;
  n.status:=#unlocked;
  n.global:=self.global;
  n.explicitNotifications:=self.explicitNotifications;
}
```
method EditNewsletter{
    n:=self.newsletter;
    n.title:=self.newTitle;
    n.content:=self.newContent;
}

method DeleteNewsletter{
    delete self.newsletter;
}

method LockNewsletter{
    self.newsletter.status:=#locked;
}

method UnlockNewsletter{
    self.newsletter.status:=#unlocked;
}

method NewSession{
    s:=new Session;
    self.createdSession:=s;
    s.currentCurrency:=self.currentCurrency;
    s.currentLanguage:=self.currentLanguage;
    s.sessionID:=Session.allInstances->size();
}

method DeleteSession{
    delete self.session;
}

method LogIn{
    s:=self.session;
    s.customer := self.customer;
    self.customer.numberOfLogons:=self.customer.numberOfLogons+1;
    if c.customerShoppingCart->size()=0 then
        new RestorePreviousShoppingCart(customer:=self.customer,session:=s) occurs;
    else
        if self.session.shoppingCart->size()=1 then
            csc:=new CustomerShoppingCart;
            csc.customer:=self.customer;
            csc.shoppingCartItem:=self.session.shoppingCart.shoppingCartItem;
            self.session.shoppingCart.shoppingCartItem:=oclEmpty(Set(ShoppingCartItem));
            self.session.shoppingCart:=oclEmpty(ShoppingCart);
        endif
    endif
}

method AddProductToShoppingCart{
    //Shopping cart item is created
    sci:=new ShoppingCartItem;
    sci.quantity:=self.quantity;
    sci.product:=self.product;
    sci.attribute:=self.attribute;

    if self.session.shoppingCart->size()=0 then
        //The session has a shopping cart
        self.session.shoppingCart.shoppingCartItem :=
            self.session.shoppingCartItem->asSet() ->union(Set{sci})->asSequence();
    else
        //The session does not have a shopping cart
        if self.session.customer.isUndefined() then
            //The session is anonymous
            asc := new AnonymousShoppingCart;
            self.session.shoppingCart:=asc;
            asc.shoppingCartItem:=sci;
        else
            //The session is authenticated
            csc:=new CustomerShoppingCart;
            csc.customer:=self.session.customer;
            csc.shoppingCartItem:=sci;
            self.session.customerShoppingCart:=csc;
        endif
    endif
}
CSTL and its application to the osCommerce case study.
Albert Tort

```java
//The customer is logged in
if self.session.customer.customerShoppingCart->size()>0 then
    //The customer has a previous shopping cart
    self.session.customer.customerShoppingCart.shoppingCartItem := self.session.customer.customerShoppingCart
        .shoppingCartItem->asSet()->union(Set{sci}) ->asSequence();
else
    //The customer does not have a previous shopping cart
    csc:=new CustomerShoppingCart;
    csc.customer:=self.session.customer;
    csc.shoppingCartItem:=self.session.shoppingCart
        .shoppingCartItem;
    self.session.shoppingCart:=csc;
    csc.shoppingCartItem:=sci;
endif
endif

method RestorePreviousShoppingCart{
    self.session.shoppingCart:=self.customer.customerShoppingCart;
}

method LogOut{
    self.session.customer:=oclEmpty(Set{Customer});
}

method NewReview{
    r:=new Review;
    r.review:=self.review;
    r.rating:=self.rating;
    r.customer:=self.customer;
    r.product:=self.product;
    r.language:=self.language;
    self.createdReview:=r;
}

method EditReview{
    r:=self.review;
    r.review:=self.newReview;
    r.rating:=self.newRating;
    r.customer:=self.newCustomer;
    r.product:=self.newProduct;
    r.language:=self.newLanguage;
}

method DeleteReview{
    delete self.review;
}

method NewOrderStatus{
    os:=new OrderStatus;
    osi=0;
    while Language.allInstances->size()>osi do
        l:=Language.allInstances->asSequence()->at(osi+1);
        osname:=HasOrderStatusName.allInstances->select(languageOfOrderStatus=l)
            ->select(orderStatusNameEvent=self)->any(true).orderStatusName;
        osl:=new OrderStatusInLanguage(orderStatus:=os,language:=l);
        osl.name:=osname.string;
        osi:=osi+1;
    endwhile
    self.createdOrderStatus:=os;
}

method EditOrderStatus{
    os:=self.orderStatus;
    os.language:=oclEmpty(Set{Language});
    i:=0;
    while Language.allInstances->size()>i do
        ```
CSTL and its application to the osCommerce case study.
Albert Tort

```
l:=Language.allInstances->asSequence()->at(i+1);
osname:=HasOrderStatusName.allInstances->select(languageOfOrderStatus=l)
->select(orderStatusNameEvent=self)->any(true).orderStatusName;
os1:=new OrderStatusInLanguage(orderStatus:=os, language:=l);
os1.name:=osname.string;
i:=i+1;
endwhile
}

method DeleteOrderStatus{
  os:=self.orderStatus;
  if Order.allInstances.orderStatus->includes(os)
  then
    self.orderStatus.status:=#disabled;
  else
    os.language:=oclEmpty(Set(Language));
    delete os;
  endif
}

method CancelOrder{
  cancelledStatus:=Store.allInstances->any(true).cancelledStatus;
  osc:=new OrderStatusChange(order:=self.order, orderStatus:=cancelledStatus);
}

method SetCancelledOrderStatus{
  self.myStore.cancelledStatus:=self.orderStatus;
}

method SetDefaultOrderStatus{
  self.myStore.defaultStatus:=self.orderStatus;
}

method SetCurrentCurrency{
  self.session.currentCurrency:=self.newCurrentCurrency;
}

method SetCurrentLanguage{
  self.session.currentLanguage:=self.newCurrentLanguage;
  changeCurrency:= self.newCurrentLanguage.defaultCurrency->notEmpty();
  if changeCurrency
  then
    if switch then
      currentCurrency:=self.newCurrentLanguage.defaultCurrency;
      new SetCurrentCurrency(session:=self.session, 
      newCurrentCurrency:=currentCurrency) occurs;
    endif
  endif
}

method UpdateOrderStatus{
  s:=self.newOrderStatus;
  osc:=new OrderStatusChange(order:=self.order, orderStatus:=s);
  osc.comments:=self.comments;
}

method EditCustomerDetails{
  c:=self.customer;
  c.gender:=self.newGender;
  c.firstName:=self.newFirstName;
  c.lastName:=self.newLastName;
  c.dateOfBirth:=self.newDateOfBirth;
  c.eMailAddress:=self.newEMailAddress;
  c.phone:=self.newPhone;
  c.fax:=self.newFax;
  c.newsletter:=self.newNewsletter;
}
```
CSTL and its application to the osCommerce case study.
Albert Tort

method EditCustomer:
c:=self.customer;
c.gender:=self.newGender;
c.firstName:=self.newFirstName;
c.lastName:=self.newLastName;
c.dateOfBirth:=self.newDateOfBirth;
c.eMailAddress:=self.newEMailAddress;
c.phone:=self.newPhone;
c.fax:=self.newFax;
c.newsletter:=self.newNewsletter;
c.password:=self.newPassword;
c.globalNotifications:=self.newGlobalNotifications;
}

method NewCustomerAddress:
ad:=new Address;
ad.gender:=self.gender;
ad.firstName:=self.firstName;
ad.lastName:=self.lastName;
ad.company:=self.company;
ad.street:=self.street;
ad.suburb:=self.suburb;
ad.postCode:=self.postCode;
ad.city:=self.city;
ad.state:=self.state;
ad.zone:=self.zone;
ad.country:=self.country;
adSet:=Set{ad};
self.customer.address:=self.customer.address->union(adSet);
}

method EditCustomerAddress{changedAddress:=self.address;
newAddress:=self.newAddress;
oldAddresses:=self.customer.address;
if oldAddresses->size()=1 then
    self.customer.address:=Set{newAddress};
    self.customer.address:=self.customer.address-Set{changedAddress};
else
    self.customer.address:=oldAddresses->union(Set{newAddress});
    self.customer.address:=self.customer.address-Set{changedAddress};
endif
}

method PrimaryCustomerAddressChange{
self.customer.primary:=self.address;
}

method DeleteCustomerAddress{deletedAddress:=self.address;
self.customer.address:=self.customer.address-Set{deletedAddress};
}

method NewProductNotificationSubscription{
previousSubscriptions:=self.customer.explicitNotifications;
newProduct:=self.newSubscribedProduct;
if self.customer.explicitNotifications->size()>0 then
    self.customer.explicitNotifications:=previousSubscriptions->union(Set{newProduct});
else
    self.customer.explicitNotifications:=self.newSubscribedProduct;
endif
}

method DeleteProductNotificationSubscription{deletedSubscription:=self.deletedSubscribedProduct;
previousSubscriptions:=self.customer.explicitNotifications;
self.customer.explicitNotifications:=previousSubscriptions-Set{deletedSubscription};
}
CSTL and its application to the osCommerce case study.
Albert Tort

method EditGlobalNotifications{
    self.customer.globalNotifications:=self.newGlobalNotifications;
}

method DeleteCustomer{
    //Delete reviews of customer
    while self.customer.review->size()>0 do
        r:=self.customer.review->any(true);
        r.product:=oclEmpty(Set(Product));
        r.language:=oclEmpty(Set(Language));
        r.customer:=oclEmpty(Set(Customer));
        delete r;
    endwhile

    //Delete shopping cart if needed
    if self.customer.customerShoppingCart->size()>0 then
        delete self.customer.customerShoppingCart;
    endif

    //Delete customer or set it to disabled
    if self.customer.order->size()>0 then
        new CustomerStatusChange(customer:=self.customer, newStatus:=#disabled)
        occurs;
    else
        delete self.customer;
    endif
}

method CustomerStatusChange{
    self.customer.status:=self.newStatus;
}