Electronic Supplementary Material

A Visual Language for Defining Recognizable Modeling Activities

Mario Janke, Tobias Kuschke, and Patrick Mäder

Technische Universität Ilmenau, Ilmenau, Germany
e-mail: mario.janke@tu-ilmenau.de, patrick.maeder@tu-ilmenau.de

Contents:
Experimental material containing the questionnaire, the tutorial, the instructions and tasks shown during the experiment and the visual and textual activity definitions for the tasks used for the experiment reported in our paper. Participants were randomly assigned to group A or B which performed tasks in opposite representations (visual or textual). This material presents instructions only of group A.

ID_____

Your knowledge and experience in the area of the experiment

We want to understand your prior knowledge and experience relevant to the experiment. Please tell us how you would rate your knowledge in the areas below. Also, include how many years of practical project experience you have in these areas. If you don't have practical experience in an area, please select "0".

	(←less) Knowledge (more →)					Project experience
Software and system development	0 ☐	1 ☐	2 ☐	3 ☐	4 ☐	_____ years
Model-based development	0 ☐	1 ☐	2 ☐	3 ☐	4 ☐	_____ years
UML modeling	0 ☐	1 ☐	2 ☐	3 ☐	4 ☐	_____ years
Complex event processing	0 ☐	1 ☐	2 ☐	3 ☐	4 ☐	_____ years

Reading activity definitions (Task 1)

Question	Answer				
Q1	A ☐	B ☐	C ☐	D ☐	**1 correct answer**
Q2	A ☐	B ☐	C ☐	D ☐	
Q3	A ☐	B ☐	C ☐	D ☐	
Q4	A ☐	B ☐	C ☐	D ☐	
Q5	A ☐	B ☐	C ☐	D ☐	**2 correct answers**
Q6	A ☐	B ☐	C ☐	D ☐	

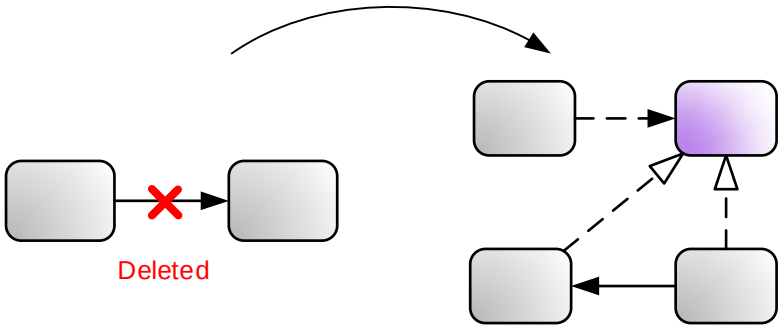
Your feedback

Which kind of language would you prefer for reading and writing activity definitions?

- Reading: ☐ Visual definition language ☐ Textual definition language
- Writing: ☐ Visual definition language ☐ Textual definition language

Tutorial

During the model-driven design of software systems developers perform ***editing operations*** that ***add, delete*** or ***modify*** model elements. Often, certain sequences of editing operation recur during the design phase. We call such sequences ***modeling activities,*** e.g., deleting an association between two classes in a UML class diagram and replacing the connection by an interface and a proxy class as shown below.

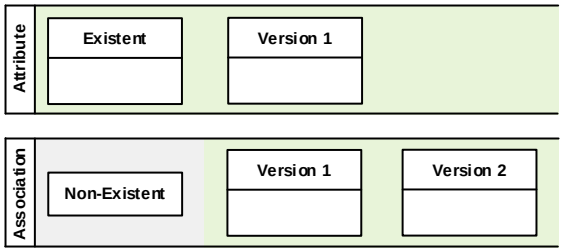


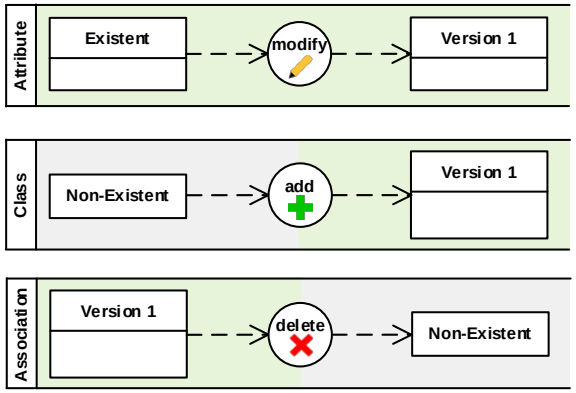
When analyzing the editing operations of a developer the intention to perform a certain modeling activity can be recognized and a suitable automatic activity completion can be suggested. We developed an approach that can recognize pre-defined modeling activities and auto-complete them by analyzing editing operation events. It uses event pattern matching for computing solutions. To enable the approach, modeling activities must be defined by specifying all required information:

- comprised model elements and the states that they run through during the activity
- editing operations that must be performed to trigger the state transitions
- constraints of element properties and timing that must be fulfilled by elements states and editing operations
- default values of certain element properties for preforming automatic completion actions

So far, modeling activities had to be defined using a textual event processing language. As an alternative, we developed a visual definition language, which is aimed to make the reading and writing of modeling activity definitions easier. The purpose of this experiment is to compare the usability of the common textual event processing language *RAPIDE* with the new visual definition language.

Modeling activity definitions specify editing operations on model elements. In this experiment, we focus on editing operations performed on UML class models. The following UML classifiers are regarded as ***model elements***: class, component, package, interface, method, attribute, association (and its sub types aggregation and composition) and dependency. The following table explains and compares the concepts for defining modeling activities in the new visual language and the textual event processing language *RAPIDE*.

Concept	Visual definition language	Textual language RAPIDE
Model Element represents an individual element that is comprised in the modeling activity, in this experiment: **package**, **component**, **class**, **interface**, **attribute**, **association** (**aggregation** and **composition**), **dependency**	Specified as an line that shows the element type: 	Not specified in RAPIDE
Element Version represents a certain state of a model element. 3 different types are distinguished: *Existent*: saying that a model element must exist before the modeling activity is started Non-Existent: saying that a model element doesn't exist before the start of the activity or doesn't exist anymore after its deletion Version X: specifying state X of a model element meaning that there are X-1 versions defined before	Specified as class within the line of a model element: 	Specified as object type **ElementVersion**: `ElementVersion ?V1, ?V2;`

Editing Operation represents an event that is triggered by an operation of the type **add**, **delete** or **modify** on a model element which causes a new element version.
 	
It comprises the following information:
• operation type
• time stamp
• resulting element version
	Specified as a circle between a previous and the resulting element version within the line of a model element:
	 
		Specified as an object of the type **EditingOperation** that has the properties **Operation_Type**, **TimeStamp** and **Element_Version**:
		
		`EditingOperation( Element_Version is ?V1,`
		`Operation_Type is "add",`
		`TimeStamp is "2015-04-16T13:04:25"`
		`)`
		
		`EditingOperation ?O1, ?O2;`
		
		`?O1(...) || ?O2(...)`

Property Constraints represent conditions that a certain element version has to fulfill, i.e., properties of the model element that must have certain values:

Nothing: Element1.Stereotype = -

Fix value: Element1.Type = class

Reference: Element1.Parent.ID = Package1.ID

Logical expressions:

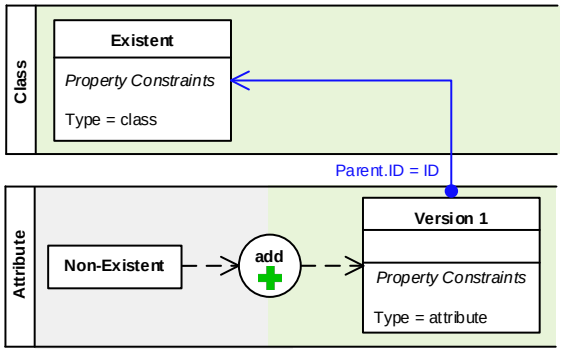
Element1.Name = (!NoName && !Element2.Name)

Element3.SubType = aggregation, composition

Logical operators:

Equal: "="
Unequal: "!="
And: "&&"
Or: "||" alternatively ","
Not: "!"
Greater than: ">"
Less than: "<"

Specified as a logical expressions within element versions or on connectors between element verions (references):



A **reference** is represented by a blue connector that points to the referenced element version (arrow end). The connector source is the element version to which the constraint belongs (dot end). The left side of the logical expression specifies a property of the source element version while the right side specifies the compared property of the referenced element version.

Logical expressions and operators:
as explained in the left column

Specified as logical expression after **where** or within the declaration:

```
(ElementVersion ?V1, ?V2; EditingOperation ?O1)
(
 EditingOperation(Element_Version is ?V1)

                  where (?V1.Type = "class") ||

 ?O1(Element_Version is ?V2, Operation_Type is "add")

                  where (?V2.Type = "attribute" and
                         ?V2.Parent.ID = ?V1.ID)
)
```

Nothing: ?V1.Stereotype = ""
Fix value: ?V1.Type = "class"
Reference: ?V2.Parent.ID = ?V1.ID

Logical operators:

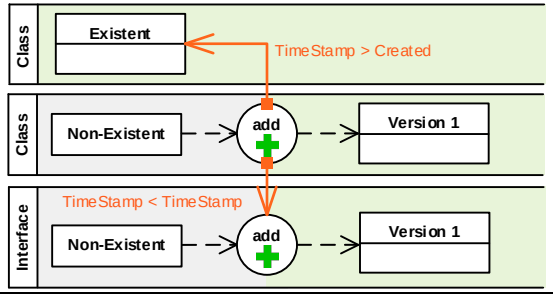
Equal: "=" alternatively "**is**" (within declarations)
Unequal: "/="
And: "**and**"
Or: "**or**"
Not: "**not**"
Greater than: ">"
Less than: "<"

Sequence Constraints represent timing conditions that editing operations have to fulfill (on the left always editing operation):

Operation before operation:
Operation1.TimeStamp < Operation2.TimeStamp

Operation after existence of an element:
Operation1.TimeStamp > Element2.Created

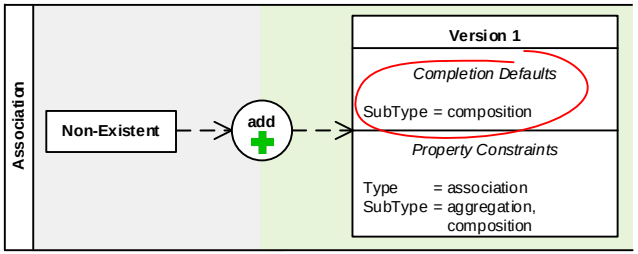
Specified as a logical expression on an orange connector between editing operations or an editing operation and an element version (source indicated by solid square):



Specified as logical expression after **where:**

```
(ElementVersion ?V1; EditingOperation ?O1, ?O2)
(
 EditingOperation(Element_Version is ?V1) ||

 ?O1(Operation_Type is "add") ||

 ?O2(Operation_Type is "add") where (

                  ?O2.TimeStamp > ?V1.Created and
                  ?O2.TimeStamp < ?O1.TimeStamp)
)
```

Completion Action represents an automatically executed operation to complete a partially performed modeling activity. An activity definition must comprise a completion action for every specified editing operation.
 	
It comprises the following information:
• Action type
• Element properties
 	
The action type equals the editing operation type while element properties equal and the property constraints of the resulting element version.
	Not <u>explicitly</u> specified:
	
	• are derived from editing operations and their resulting element versions
	• for each editing operation a corresponding completion action will be generated
	• each property constraint will be taken as an element property
	• each completion default will be added as an element property (or overwrite values of already contained properties)
		Specified as object of the type **CompletionAction** and a set of value assignments:
		
		`(ElementVersion ?V1,?V2,?V3;`
		` EditingOperation ?O1, ?O2;`
		` CompletionAction ?A1, ?A2 )`
		`(`
		`  EditingOperation(Element_Version is ?V1) ||`
		
		`  ?O1(Element_Version is ?V2, Operation_Type is “add”)`
		`  where (`
		`  ?V2.Type = “class” and`
		`  ?V2.ParentID = ?V1.ID) ||`
		`  ?O2( ... ) where (`
		`  ... )`
		
		`) ||>`
		
		`  ?A1.Action = “add”;`
		`  ?A1.Type = “class”;`
		`  ?A1.Parent.ID = ?V1.ID;`
		
		`  ?A2.Action ...`
		
		All value assignments have to be written explicitly!

Completion Defaults represent default values of element properties that are applied when a completion action is executed. They overwrite specified property constraints:
 	
Completion Default: SubType = composition
Property Constraint: SubType= aggregation, composition
 	
Meaning the model element can have the sub type aggregation or composition when edited by the developer. When being automatically edited for completing the modeling activity, the sub type is set to composition.
	Specified as a value assignment within an element version:
	
	 
	
		Specified within the set of completion action assignments:
		
		`(ElementVersion ?V1, ...;`
		` EditingOperation ?O1, ...;`
		` CompletionAction ?A1, ...; )`
		`(`
		`  ...`
		`  ?O1(Element_Version is ?V1, Operation_Type is “add”)`
		`  where (`
		`  ?V1.Type = “association” and`
		`  (?V1.SubType = “aggregation” or`
		`  ?V1.SubType = “composition”))`
		`  ...`
		
		`) ||>`
		
		`  ?A1.Action = “add”;`
		`  ?A1.Type = “association”;`
		`  ?A1.SubType = “composition”;`
		`  ...`

Experiment instructions and tasks (as shown to group A)

Please answer the first question on the questionnaire concerning „**Your knowledge and experience in the area of the experiment**“!

When finished please click *Next*.

Please read the tutorial now!

You may use it throughout the experiment.

After reading the tutorial please click *Next.*

Task 1: Reading activity definitions

You will now be asked to answer six questions. Please note the answers for each question in the corresponding table on the questionnaire.

Please click on *Start* now and follow the instructions.

Please take the activity definition ***R1G*** and answer the following question!

Q1: How many different model elements does this activity definition include?

A) 1 pre-existing and 2 new
B) 3 pre-existing and 4 over all
C) 5 over all
D) 2 pre-existing and 3 new

Note your answers in the table and click *Next*.

Please take the activity definition ***R2T*** and answer the following question!

Q2: How many model elements must exist before starting this activity and how many do at least exist after the activity has been finished?

A) 3 before and 6 after

B) 4 before and 3 after

C) 4 before and 6 after

D) None of the above

Note your answers in the table and click *Next*.

Please take the activity definition ***R1G*** and answer the following question!

Q3: What editing operations are specified in this activity definition?

A) 2 add-operations of associations, 2 modify-operations on associations and 1 delete-operation of a class

B) 1 delete-operation of an association, 1 add-operation of an association, 1 modify-operation on an association, 1 modify-operation on a class and 1 add-operation of a class

C) 1 delete-operation of an attribute, 2 add-operations of classes and 2-modify-operations on associations

D) 1 add-operation of a class, 1 modify-operation on a class, 1 add-operation of an association, 1 modify-operation on an association and 1 delete-operation of an attribute

Note your answers in the table and click *Next*.

Please take the activity definition ***R2T*** and answer the following question!

Q4: Which of the statements are specified in the activity definition? (Check all that apply!)

A) The added class will get the name “New” when automatically added for completion.

B) The added association which is modified to the sub type composition ends at the added class.

C) The class must be added after the deletion of the pre-existing association.

D) The 2 pre-existing classes are the source elements of the 2 added associations.

Note your answers in the table and click *Next*.

Please take the activity definition ***R1G*** and answer the following question!

Q5: Which of the statements are specified in the activity definition? (Check all that apply!)

A) The name of the added class must be modified to the name of the pre-existing class.

B) The added association must not have the sub type “composition“.

C) When automatically adding the association for completion, its sub type would be set to “aggregation“.

D) The new class must be added to a package.

Note your answers in the table and click *Next*.

Please take the activity definition ***R2T*** and answer the following question!

Q6: Which of the following performed editing operations would match this activity definition, i.e., fulfill the specified constraints? (Check all that apply!)

A) A many-to-many association was deleted that started and ended at the same pre-existing class.

B) A class was added after the pre-existing component has been created.

C) A navigable aggregation pointing from a pre-existing to the new class was added and its source- and target multiplicity modified to “*” and “1”.

D) A class was added to the parent of the pre-existing classes.

Note your answers in the table and click *Next*.

Task 2: Writing activity definitions

You will now be asked to perform two writing tasks.

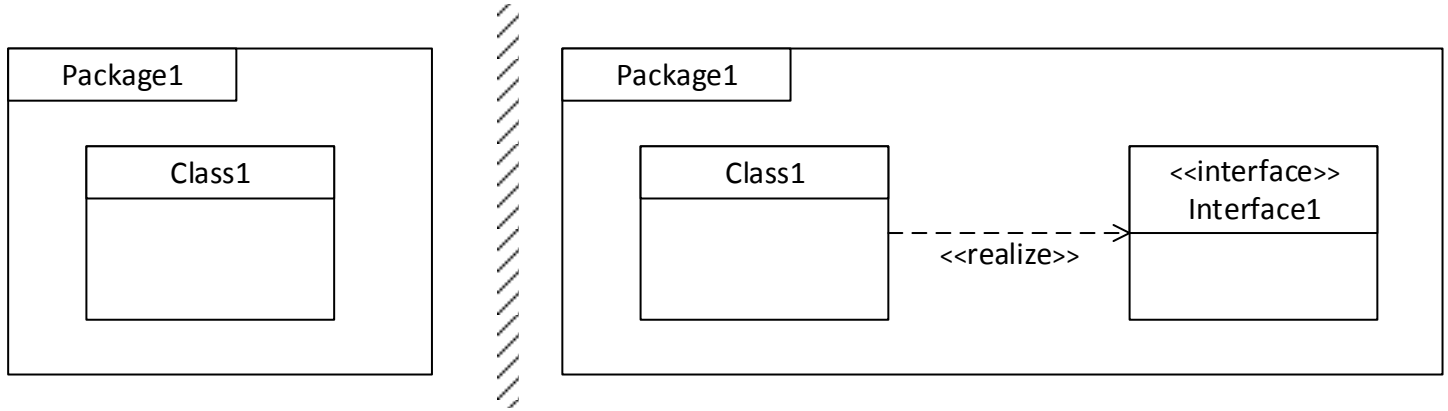
Please click on *Start* and follow the instructions!

Please take the activity definition ***W1T***. On the next slide you will see a description of a modeling activity. Please complete the given definition according to this description by using the textual definition language. You can use the definition ***R2T*** and the tutorial as a guideline.

Note: Please include all details: variable declarations, allocations, operators and signs such as brackets, semicolons, quotes and double bars. Please use a structured notation similar to the definition *R2T.*

Modeling activity

A class must exist within a package. A new interface must be added into the same package. The completion default for the property *Name* should be “NewInterface”. A new connector of the *Type* “dependency“ and with the *Stereotype* “realize” must be added between the existing class (*Source*) and the new interface (*Target*).



Model elements before (left) and after (right) performing the modeling activity.

When finished please click *Next*.

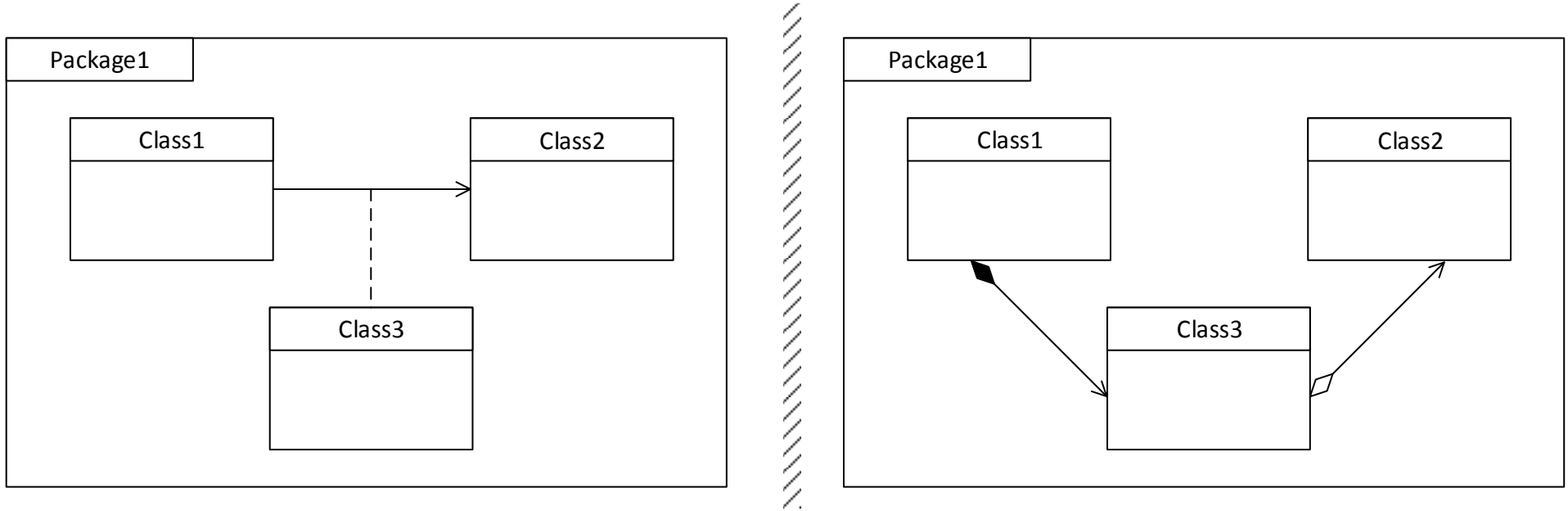
Please take the activity definition ***W2G***. On the next slide you will see a description of an activity modification. Please modify the given definition according to this description.

Note:
If you need to remove parts of the definition, please cross them out. If you remove a whole model element, you only need to cross out the element line. Otherwise cross out every detail (versions, editing operations, defaults, constraints, connectors).

If you add content, you can exclude the following details: the headlines “Completion Defaults“ and “Property Constraints“ and the editing operation icons. But please draw both connector ends, i.e., the arrows and the dots/squares.

Modification

The existing association and its association class Class3 should not be deleted. Instead a modification-operation on the association should be performed that changes the *SubType* into "composition", the *Target.ID* into the ID of Class3 and *AssociationClass.ID* into "-". This editing operation replaces the addition (and modification) of a new composition. Further, no new class should be added because Class3 was not deleted. The *Source.ID* for adding the new aggregation must now be the ID of Class3.



Model elements before (left) and after (right) performing the modeling activity

When finished please click *Next*.

You finished the experiment tasks.

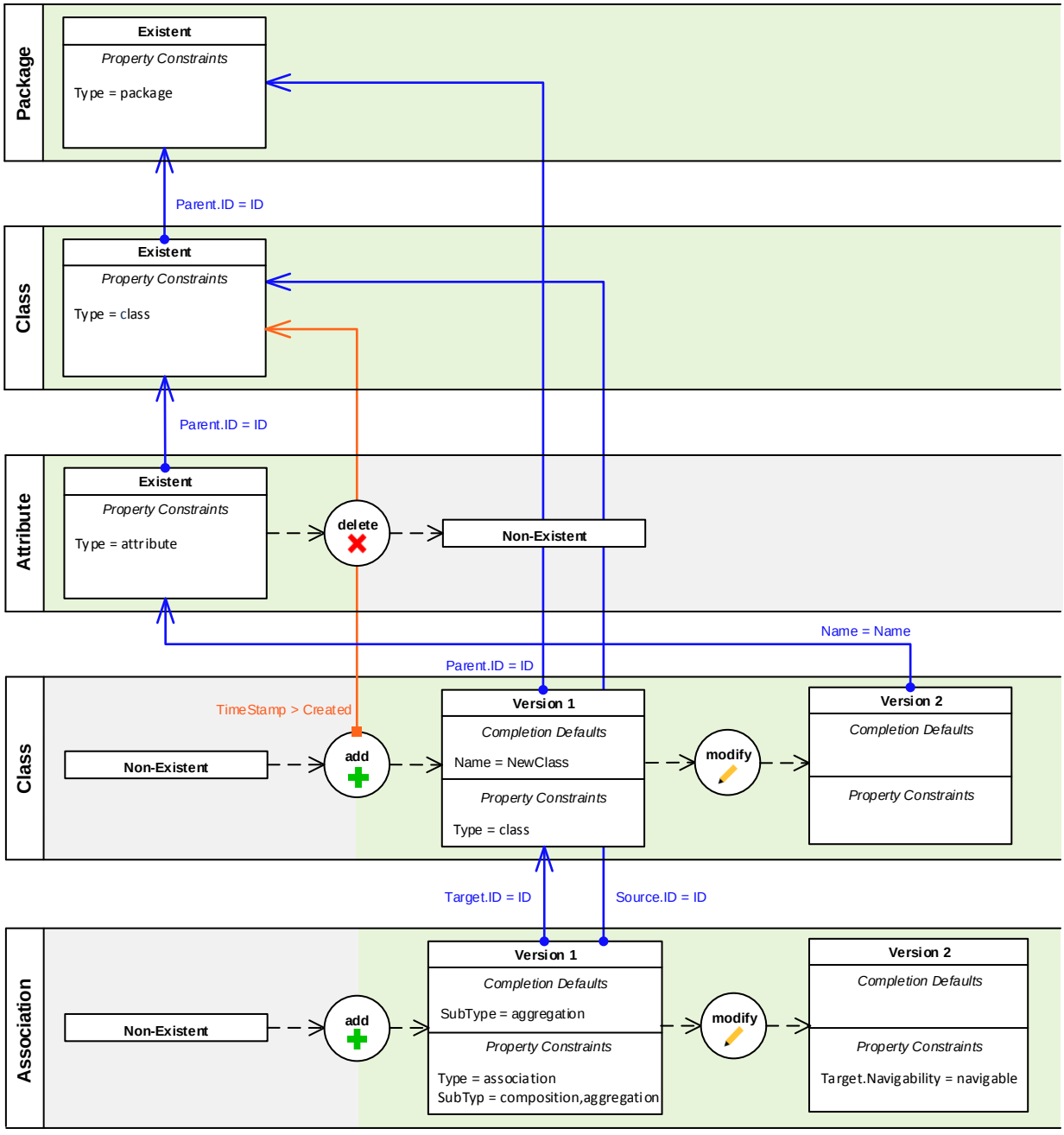
Please answer the feedback question on the questionnaire now!

When finished please click *Next*.

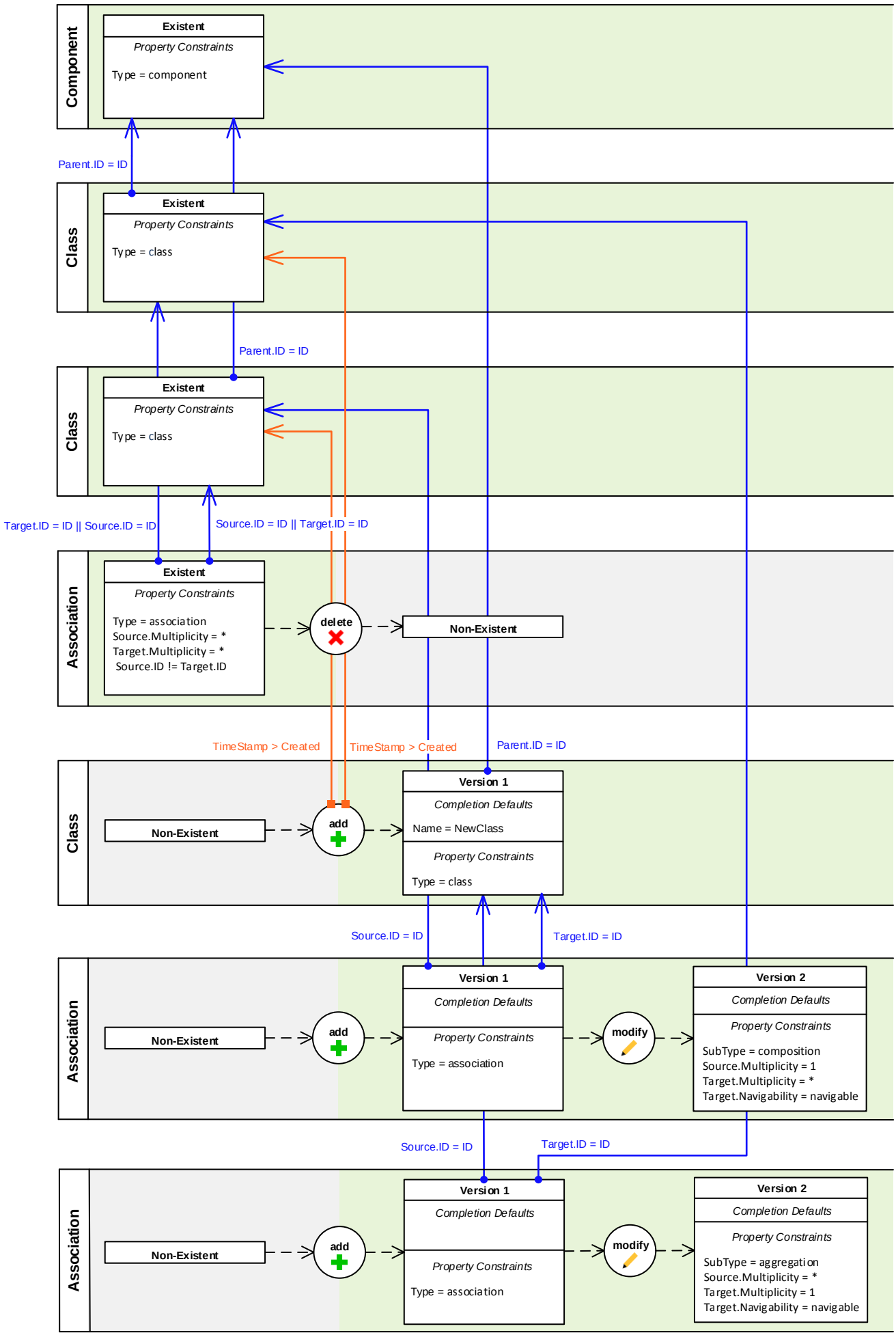
Finally, we would like to ask you about your personal opinion to the visual definition language.

Please click on *Finish* and answer the interview questions!

Activity definitions for the tasks


Package
Existent
Property Constraints
Type = package
Parent.ID = ID
Class
Existent
Property Constraints
Type = class
Parent.ID = ID
Attribute
Existent
Property Constraints
Type = attribute
delete
Non-Existent
Name = Name
Parent.ID = ID
TimeStamp > Created
Class
Non-Existent
add
Version 1
Completion Defaults
Name = NewClass
Property Constraints
Type = class
modify
Version 2
Completion Defaults
Property Constraints
Target.ID = ID
Source.ID = ID
Association
Non-Existent
add
Version 1
Completion Defaults
SubType = aggregation
Property Constraints
Type = association
SubTyp = composition,aggregation
modify
Version 2
Completion Defaults
Property Constraints
Target.Navigability = navigable

```
(ElementVersion ?V1, ?V2, ?V3, ?V4, ?V5, ?V6, ?V7, ?V8;
 EditingOperation ?O1, ?O2, ?O3, ?O4, ?O5;
 CompletionAction ?A1, ?A2, ?A3, ?A4, ?A5)
(
   EditingOperation(Element_Version is ?V1) where ( ?V1.Type = "package") ||
   EditingOperation(Element_Version is ?V2) where ( ?V2.Type = "class" and
                                                    ?V2.Parent.ID = ?V1.ID) ||
   EditingOperation(Element_Version is ?V3) where ( ?V3.Type = "attribute" and
                                                    ?V3.Parent.ID = ?V2.ID) ||
   ?O1(Element_Version is ?V4, Operation_Type is "delete") where ( ?V4.ID = ?V3.ID) ||
   ?O2(Element_Version is ?V5, Operation_Type is "add")    where ( ?O2.TimeStamp > ?V2.Created and
                                                                   ?V5.Type = "class" and
                                                                   ?V5.Parent.ID = ?V1.ID) ||
   ?O3(Element_Version is ?V6, Operation_Type is "modify") where ( ?V6.ID = ?V5.ID and
                                                                   ?V6.Name = ?V3.Name) ||
   ?O4(Element_Version is ?V7, Operation_Type is "add")    where ( ?V7.Type = "association" and
                                                                   (?V7.SubType = "composition" or
                                                                    ?V7.SubType = "aggregation") and
                                                                   ?V7.Source.ID = ?V2.ID and
                                                                   ?V7.Target.ID = ?V5.ID) ||
   ?O5(Element_Version is ?V8, Operation_Type is "modify") where ( ?V8.ID = ?V7.ID and
                                                                   ?V8.Target.Navigability = "navigable")
) ||>
        ?A1.Action = "delete";
        ?A1.Type = "attribute";
        ?A1.ID = ?V3.ID;

        ?A2.Action = "add";
        ?A2.Type = "class";
        ?A2.Parent.ID = ?V1.ID;
        ?A2.Name = "NewClass";

        ?A3.Action = "modify";
        ?A3.Type = "class";
        ?A3.ID = ?V5.ID;
        ?A3.Name = ?V3.Name;

        ?A4.Action = "add";
        ?A4.Type = "association";
        ?A4.SubType = "aggregation";
        ?A4.Source.ID = ?V2.ID;
        ?A4.Target.ID = ?V5.ID;

        ?A5.Action = "modify";
        ?A5.Type = "association";
        ?A5.ID = ?V7.ID;
        ?A5.Target.Navigability = "navigable";

        CompletionAction[] Actions  = new CompletionAction[]{?A1,?A2,?A3,?A4,?A5};
        EditingOperation[] Events   = new EditingOperation[]{?O1,?O2,?O3,?O4,?O5};
        ElementVersion[]   Versions = new ElementVersion[]{?V1,?V2,?V3,?V4,?V5,?V6,?V7,?V8};

        generate Match( Actions, Events, Versions );
```


Component
Existent
Property Constraints
Type = component
Parent.ID = ID
Class
Existent
Property Constraints
Type = class
Parent.ID = ID
Class
Existent
Property Constraints
Type = class
Target.ID = ID || Source.ID = ID
Source.ID = ID || Target.ID = ID
Association
Existent
Property Constraints
Type = association
Source.Multiplicity = *
Target.Multiplicity = *
Source.ID != Target.ID
delete
Non-Existent
TimeStamp > Created
TimeStamp > Created
Parent.ID = ID
Class
Non-Existent
add
Version 1
Completion Defaults
Name = NewClass
Property Constraints
Type = class
Source.ID = ID
Target.ID = ID
Association
Non-Existent
add
Version 1
Completion Defaults
Property Constraints
Type = association
modify
Version 2
Completion Defaults
Property Constraints
SubType = composition
Source.Multiplicity = 1
Target.Multiplicity = *
Target.Navigability = navigable
Source.ID = ID
Target.ID = ID
Association
Non-Existent
add
Version 1
Completion Defaults
Property Constraints
Type = association
modify
Version 2
Completion Defaults
Property Constraints
SubType = aggregation
Source.Multiplicity = *
Target.Multiplicity = 1
Target.Navigability = navigable

R2T

```
(ElementVersion ?V1, ?V2, ?V3, ?V4, ?V5, ?V6, ?V7, ?V8 ,?V9, ?V10;
 EditingOperation ?O1, ?O2, ?O3, ?O4, ?O5, ?O6;
 CompletionAction ?A1, ?A2, ?A3, ?A4, ?A5, ?A6)
(
  EditingOperation(Element_Version is ?V1) where ( ?V1.Type = "component") ||
  EditingOperation(Element_Version is ?V2) where ( ?V2.Type = "class" and
                                                   ?V2.Parent = ?V1.ID) ||
  EditingOperation(Element_Version is ?V3) where ( ?V3.Type = "class" and
                                                   ?V3.Parent = ?V1.ID) ||
  EditingOperation(Element_Version is ?V4) where ( ?V4.Type = "association" and
                                                   (?V4.Source.ID = ?V2.ID or ?V4.Source.ID = ?V3.ID) and
                                                   (?V4.Target.ID = ?V3.ID or ?V4.Target.ID = ?V2.ID) and
                                                   ?V4.Source.ID /= ?V4.Target.ID and
                                                   ?V4.Source.Multiplicity = "*" and
                                                   ?V4.Target.Multiplicity = "*") ||
  ?O1(Element_Version is ?V5, Operation_Type is „delete") where (   ?V5.ID = ?V4.ID) ||
  ?O2(Element_Version is ?V6, Operation_Type is „add")    where (   ?O2.TimeStamp > ?V2.Created and
                                                                    ?O2.TimeStamp > ?V3.Created and
                                                                    ?V6.Type = "class" and
                                                                    ?V6.Parent.ID = ?V1.ID) ||
  ?O3(Element_Version is ?V7, Operation_Type is „add")    where (   ?V7.Type = "association" and
                                                                    ?V7.Source.ID = ?V3.ID and
                                                                    ?V7.Target.ID = ?V6.ID) ||
  ?O4(Element_Version is ?V8, Operation_Type is „modify") where (   ?V8.ID = ?V7.ID and
                                                                    ?V8.SubType = "composition" and
                                                                    ?V8.Source.Multiplicity = "1" and
                                                                    ?V8.Target.Multiplicity = "*" and
                                                                    ?V8.Target.Navigability = "navigable") ||
  ?O5(Element_Version is ?V9, Operation_Type is „add")    where (   ?V9.Type = "association" and
                                                                    ?V9.Source.ID = ?V6.ID and
                                                                    ?V9.Target.ID = ?V2.ID) ||
  ?O6(Element_Version is ?V10, Operation_Type is „modify") where (?V10.ID = ?V9.ID and
                                                                    ?V10.SubType = "aggregation" and
                                                                    ?V10.Source.Multiplicity = "*" and
                                                                    ?V10.Target.Multiplicity = "1" and
                                                                    ?V10.Target.Navigability = "navigable")
) ||>
      ?A1.Action = "delete";
      ?A1.Type = "association";
      ?A1.ID = ?V4.ID;

      ?A2.Action = "add";
      ?A2.Type = "class";
      ?A2.Parent = ?V1.ID;
      ?A2.Name = "NewClass";

      ?A3.Action = "add";
      ?A3.Type = "association";
      ?A3.Source = ?V3.ID;
      ?A3.Target = ?V6.ID;

      ?A4.Action = "modify";
      ?A4.ID = ?V7.ID;
      ?A4.Type = "association";
      ?A4.SubType = "composition"
      ?A4.Source.Multiplicity = "1";
      ?A4.Target.Multiplicity = "*";
      ?A4.Target.Navigability = "navigable";

      ?A5.Action = "add";
      ?A5.Type = "association";
      ?A5.Source = ?V6.ID;
      ?A5.Target = ?V2.ID;

      ?A6.Action = "modify";
      ?A6.ID = ?V9.ID;
      ?A6.Type = "association";
      ?A6.SubType = "aggregation";
      ?A6.Source.Multiplicity = "*";
      ?A6.Target.Multiplicity = "1";
      ?A6.Target.Navigability = "navigable";

      CompletionAction[] Actions  = new CompletionAction[]{?A1,?A2,?A3,?A4,?A5,?A6};
      EditingOperation[] Events   = new EditingOperation[]{?O1,?O2,?O3,?O4,?O5,?O6};
      ElementVersion[]   Versions = new ElementVersion[]{?V1,?V2,?V3,?V4,?V5,?V6,?V7,?V8,?V9,?V10};

      generate Match( Actions, Events, Versions );
```

(ElementVersion ?V1, ?V2, ?V3, ?V4;

W1T

)

(

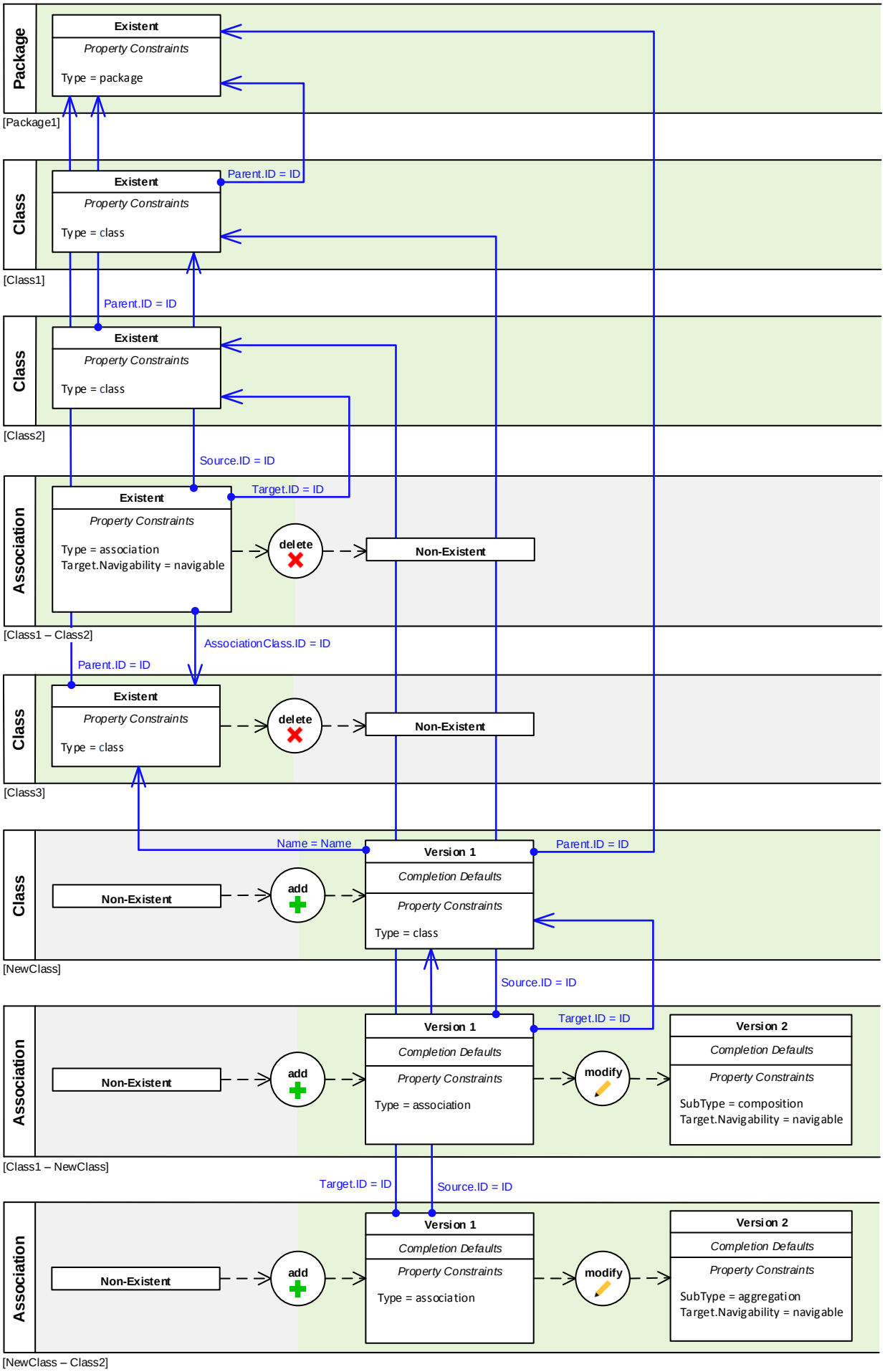
) ||>

CompletionAction[] Actions = new CompletionAction[]{

EditingOperation[] Events = new EditingOperation[]{

ElementVersion[] Versions = new ElementVersion[]{

generate Match(Actions, Events, Versions);


Package
Existent
Property Constraints
Type = package
[Package1]
Class
Existent
Property Constraints
Type = class
Parent.ID = ID
[Class1]
Parent.ID = ID
Class
Existent
Property Constraints
Type = class
[Class2]
Source.ID = ID
Association
Existent
Property Constraints
Type = association
Target.Navigability = navigable
Target.ID = ID
delete
Non-Existent
[Class1 – Class2]
AssociationClass.ID = ID
Parent.ID = ID
Class
Existent
Property Constraints
Type = class
delete
Non-Existent
[Class3]
Class
Name = Name
Non-Existent
add
Version 1
Completion Defaults
Property Constraints
Type = class
Parent.ID = ID
[NewClass]
Source.ID = ID
Association
Non-Existent
add
Version 1
Completion Defaults
Property Constraints
Type = association
Target.ID = ID
modify
Version 2
Completion Defaults
Property Constraints
SubType = composition
Target.Navigability = navigable
[Class1 – NewClass]
Target.ID = ID
Source.ID = ID
Association
Non-Existent
add
Version 1
Completion Defaults
Property Constraints
Type = association
modify
Version 2
Completion Defaults
Property Constraints
SubType = aggregation
Target.Navigability = navigable
[NewClass – Class2]

```
(ElementVersion ?V1, ?V2, ?V3, ?V4, ?V5, ?V6, ?V7, ?V8 ,?V9, ?V10, ?V11, ?V12;
 EditingOperation ?O1, ?O2, ?O3, ?O4, ?O5, ?O6, ?O7;
 CompletionAction ?A1, ?A2, ?A3, ?A4, ?A5, ?A6, ?A7)
(
  EditingOperation(Element_Version is ?V1) where ( ?V1.Type = "package") ||
  EditingOperation(Element_Version is ?V2) where ( ?V2.Type = "class" and
                                                   ?V2.Parent = ?V1.ID) ||
  EditingOperation(Element_Version is ?V3) where ( ?V3.Type = "class" and
                                                   ?V3.Parent = ?V1.ID) ||
  EditingOperation(Element_Version is ?V4) where ( ?V4.Type = "class" and
                                                   ?V4.Parent = ?V1.ID) ||
  EditingOperation(Element_Version is ?V5) where ( ?V5.Type = "association" and
                                                   ?V5.Source.ID = ?V2.ID and
                                                   ?V5.Target.ID = ?V3.ID and
                                                   ?V5.Target.Navigability = "navigable" and
                                                   ?V5.AssociationClass.ID = ?V4.ID) ||

  ?O1(Element_Version is ?V6, Operation_Type is „delete")   where ( ?V6.ID = ?V5.ID) ||
  ?O2(Element_Version is ?V7, Operation_Type is „delete")   where ( ?V7.ID = ?V4.ID) ||
  ?O3(Element_Version is ?V8, Operation_Type is „add")      where ( ?V8.Type = "class" and
                                                                    ?V8.Parent.ID = ?V1.ID and
                                                                    ?V8.Name = ?V4.Name) ||
  ?O4(Element_Version is ?V9, Operation_Type is „add")      where ( ?V9.Type = "association" and
                                                                    ?V9.Source.ID = ?V2.ID and
                                                                    ?V9.Target.ID = ?V8.ID) ||
  ?O5(Element_Version is ?V10, Operation_Type is „modify") where ( ?V10.ID = ?V9.ID and
                                                                    ?V10.SubType = "composition" and
                                                                    ?V10.Target.Navigability = "navigable") ||
  ?O6(Element_Version is ?V11, Operation_Type is „add")     where ( ?V11.Type = "association" and
                                                                    ?V11.Source.ID = ?V8.ID and
                                                                    ?V11.Target.ID = ?V3.ID) ||
  ?O7(Element_Version is ?V12, Operation_Type is „modify") where ( ?V12.ID = ?V11.ID and
                                                                    ?V12.SubType = "aggregation" and
                                                                    ?V12.Target.Navigability = "navigable")
) ||>
    ?A1.Action = "delete";
    ?A1.Type = "association";
    ?A1.ID = ?V5.ID;

    ?A2.Action = "delete";
    ?A2.Type = "class";
    ?A2.ID = ?V4.ID;

    ?A3.Action = "add";
    ?A3.Type = "class";
    ?A3.Parent.ID = ?V1.ID;
    ?A3.Name = ?V4.Name;

    ?A4.Action = "add";
    ?A4.Type = "association";
    ?A4.Source = ?V2.ID;
    ?A4.Target = ?V8.ID;

    ?A5.Action = "modify";
    ?A5.ID = ?V9.ID;
    ?A5.Type = "association";
    ?A5.SubType = "composition";
    ?A5.Target.Navigability = "navigable";

    ?A6.Action = "add";
    ?A6.Type = "association";
    ?A6.Source = ?V8.ID;
    ?A6.Target = ?V3.ID;

    ?A7.Action = "modify";
    ?A7.ID = ?V11.ID;
    ?A7.Type = "association";
    ?A7.SubType = "aggregation";
    ?A7.Target.Navigability = "navigable";

    CompletionAction[] Actions  = new CompletionAction[]{?A1,?A2,?A3,?A4,?A5,?A6,?A7};
    EditingOperation[] Events   = new EditingOperation[]{?O1,?O2,?O3,?O4,?O5,?O6,?O7};
    ElementVersion[]   Versions = new ElementVersion[]{?V1,?V2,?V3,?V4,?V5,?V6,?V7,?V8,?V9,?V10,?V11,
                                                       ?V12};

    generate Match( Actions, Events, Versions );
```