

Creating an ECO state machine

Table of Contents

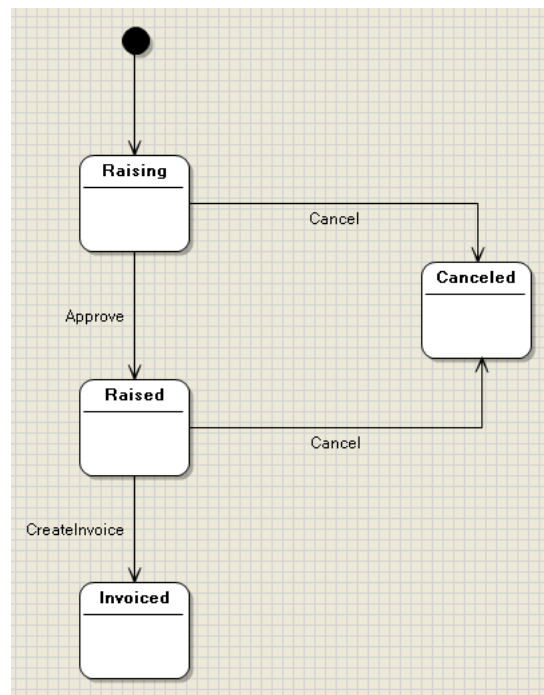
Overview	1
Prerequisites and goals	2
Adding a state machine to the Article class	3
Adding regions	9
Entry, Exit, and Effect	12
Summary	15
Index	a

1 Overview

An ECO state machine is a way of defining all states an object may be in, in addition to this the state machine controls which transitions are valid at any given point preventing the object from moving to an illogical state. For example, a PurchaseOrder may have the following states:

State	Description	Valid transitions
Raising	The order is new and may still be modified.	<ul style="list-style-type: none"> • Raised • Canceled
Raised	The order has been approved and may no longer be modified.	<ul style="list-style-type: none"> • Canceled
Invoiced	The order has had an invoice raised against it.	None
Canceled	The order has been canceled, no goods will be delivered.	None

The above rules would require an ECO state machine looking something like the following diagram:



When an instance of the PurchaseOrder class is created ECO will create a state machine for that object, this state machine will execute the ECO state machine diagram modeled against the class. When the state machine executes it will start at the "Initial state" in the diagram, illustrated as a solid black circle, immediately the state machine will follow the transition into the "Raising" state. The PurchaseOrder instance will remain in this state until either its "Approve" or "Cancel" methods are executed, at which point it will move to the appropriate state. The "Invoiced" and "Canceled" states have no outgoing transitions, so once the object enters one of these states there is no possibility for the state to change again.

2 Prerequisites and goals

Prerequisites

To successfully follow this document the user should have read the preceding articles in this series and have a copy of the model created. An understanding of UML state diagrams would be an advantage.

Goals

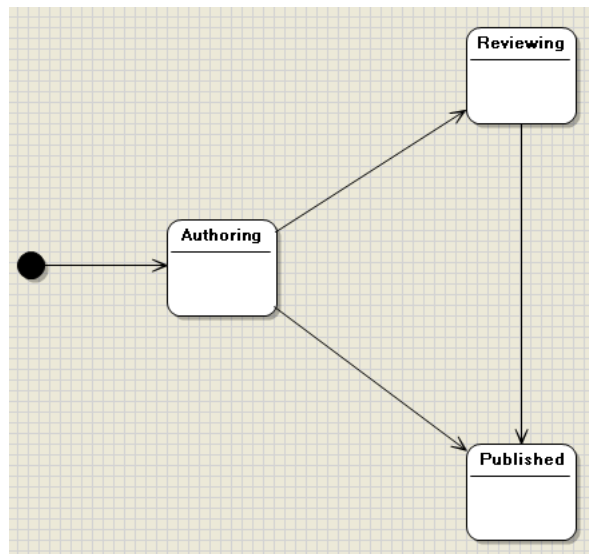
By the end of this document you will be able to

- Create ECO state machines.
- Add transitions between various states.
- Add OCL "guard" expressions to states to prevent transitions.
- Create embedded state machines (regions).

3 Adding a state machine to the Article class

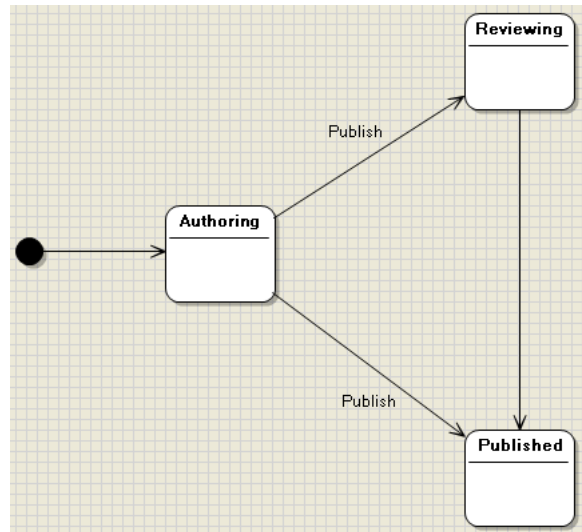
First an ECO state machine will be added to the Article class. At first this state machine will merely determine whether or not to automatically publish the article based on the trust level of its author, later it will be extended to include a review process.

1. Right-click the Author class and add a new UML attribute named "AutoPublishArticles" of the type "Boolean".
2. Right-click the Article class and select Add -> ECO State Machine.
3. From the tool palette select the "Initial" item and then click the diagram surface to drop one onto the diagram.
4. From the tool palette drop a "State" onto the diagram surface and name it "Authoring".
5. Now drop two more states onto the diagram surface and name them "Reviewing", and "Published".
6. Select the "Transition" item from the tools palette.
7. To add a transition from the Initial state to the first "Authoring" state hold down the mouse on the Initial state and then move the mouse to the target state "Authoring" before releasing the mouse.
8. Recreate the following ECO state machine diagram.



9. Go back to the class diagram by selecting its tab in the IDE. If its tab is not visible in the IDE then you can expand the "QuickStartPackage" in the Model View window and then double-click the embedded "Class diagram" node.
10. Right-click the Article class and select Add -> Trigger, name the new trigger "Publish".
11. Return to the ECO state machine diagram by clicking its tab in the IDE. If its tab is not visible you can expand the package node in the Model View window, then expand the Article class node within it and you will see a node for the ECO state machine diagram that you can double-click.
12. Select one of the two outgoing transitions from the "Authoring" state.
13. In the Object Inspector window find the "Trigger" property and from the drop down list select "Publish".
14. Now set the Trigger property for the other outgoing transition to "Publish".

You should now have a diagram that looks something like this, note how the Trigger names are displayed for both outgoing transitions of the "Authoring" state:



A running instance of an ECO state machine may have its state changed by invoking a trigger, triggers are generated as methods on the class to which the state machine diagram belongs. As you can see from the previous diagram it is possible to have more than one transition out of a single state, when modeling multiple transitions out of a single state you must make it absolutely clear which route ECO should take by ensuring only one of the transitions is valid.

- One way of implementing multiple transitions out of a state is to provide each of the transitions with a different trigger. This will make it obvious which transition to take as there will be only one that is linked to the trigger method that was executed.
- Another way of specifying a specific path is to add a "Guard" expression to each transition. When two outgoing transitions need to share the same trigger (as in the previous diagram - "Publish") an OCL expression that evaluates to true or false should be assigned to the transition's "Guard" property.

A guard will now be added to each of the "Authoring" state's outgoing transitions to ensure that only one of them is valid.

15. Click the bottom transition (the one leading to the "Published" state).

16. In the object inspector enter the following expression into the Guard property (Note that "self" is case sensitive and lowercase)

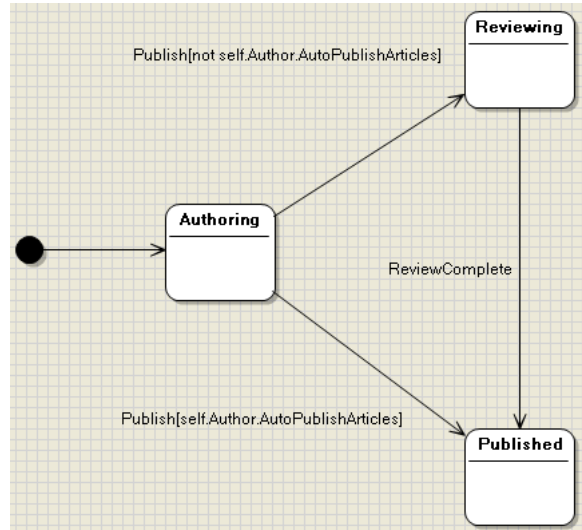
```
self.Author.AutoPublishArticles
```

17. Now add the following guard to the transition that leads to the "Reviewing" state.

```
not self.Author.AutoPublishArticles
```

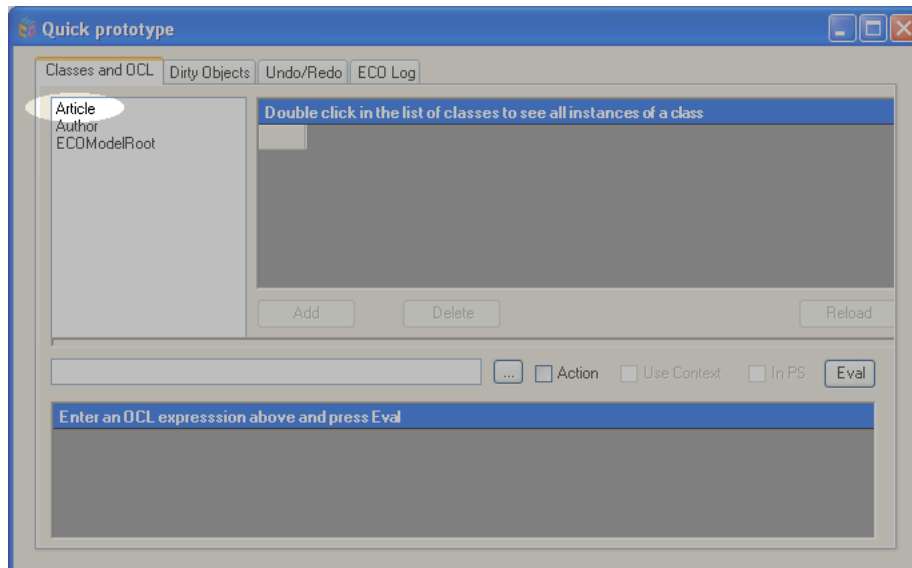
When the Article.Publish method is executed ECO will find a collection of transitions out of the current state that uses the "Publish" trigger. Next ECO will evaluate the Guard expression on each transition and ensure that only one evaluates to "true". If no guard expression evaluates to true, or more than one guard expression evaluates to true then an exception will be thrown, otherwise the transition will be taken.

In this example an article will enter the state "Published" if the author has `AutoPublishArticles = true`, or enter the "Reviewing" state if the author has `AutoPublishArticles = false`. However, in this example there is no trigger on the transition out of the "Reviewing" state so ECO will immediately follow the transition out of "Reviewing" and into "Published". To avoid this add a new trigger named "ReviewComplete" to the Article class, and then set this as the trigger for the transition from "Reviewing" to "Published". Your diagram should now look something like this:

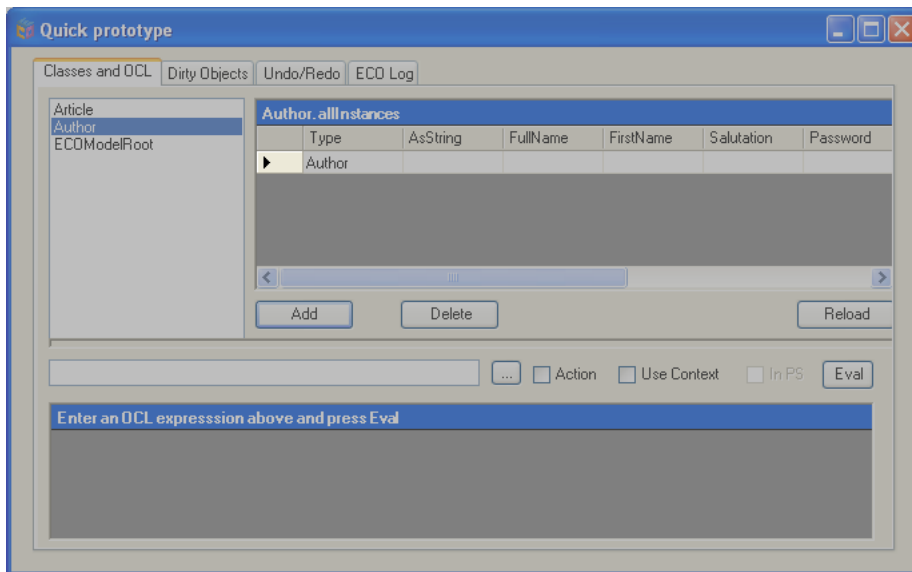


To test the state machine follow these steps (previously outlined in article #03 - Prototyping):

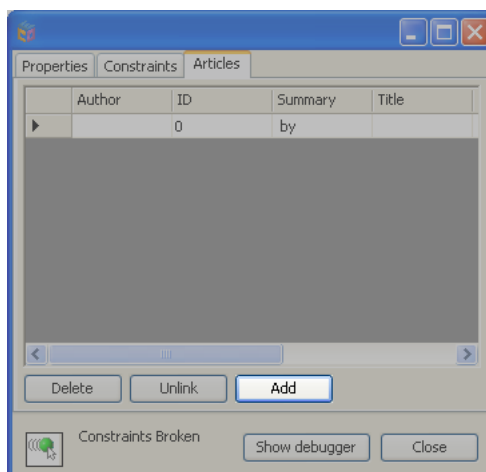
1. Open the prototype form by selecting the menu ECO Utils -> Quick Prototype.
2. Double-click the Author class in the list on the left of the form.



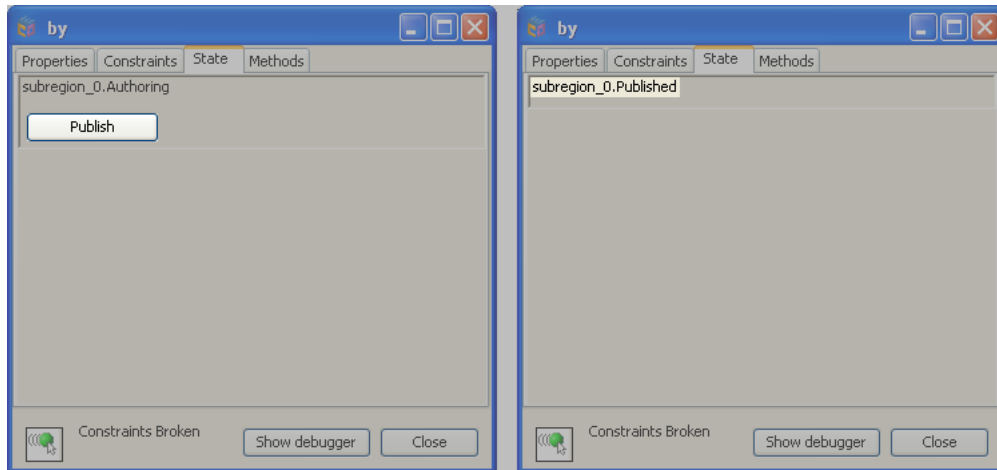
3. Click the "Add" button at the bottom of the object grid in order to add a new author.
4. Tick the "AutoPublishArticles" check box for the new Author.
5. Invoke the Author auto-form by double-clicking the first blank column on the data grid to the left of the new Author row.



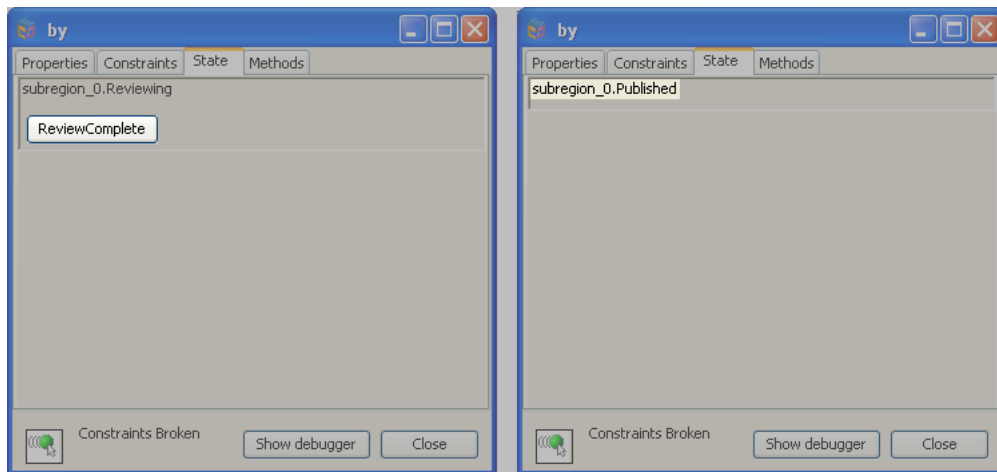
6. Select the [Articles] tab on the Author auto-form.
7. Click the "Add" button at the bottom of the Articles grid to add a new Article belonging to this author.



8. Invoke the Article auto-form, again by double-clicking the first blank column on the grid next to the appropriate row.
9. Select the [State] tab on the Article auto-form.
10. Click the "Publish" button and note how the state alters from "Authoring" to "Published".



11. Now untick the AutoPublishArticles check box for the author.
12. Add another new article.
13. Invoke the Article auto-form for the new article.
14. Select the [State] tab.
15. Click the "Publish" button and note how the state alters from "Authoring" to "Reviewing". Note also how the Publish button is replaced by a "ReviewComplete" button.
16. Click the "ReviewComplete" button and the state will change to "Published".



Transition guards

A transition guard is an OCL expression that evaluates to either True or False. When a transition is about to take place ECO will inspect each of the transitions out of the current state and compile a list of valid ones. A transition is determined to be valid if:

1. The transition is identified as being triggered by the trigger that has been executed, or the transition has no trigger.
2. The transition has no guard expression, or the guard expression evaluates to true.

If no transitions are valid or multiple transitions are valid then an exception will be thrown and no transition will take place. There are a few rules to take into account when creating transitions with no trigger, or transitions with guard expressions.

1. If any of the outgoing transitions have no trigger specified then no transition out of the same state may define a trigger.
2. If any of the outgoing transitions have a trigger then all other transitions out of the same state must also define a trigger.
3. If multiple transitions specify the same trigger then only a single transition must be valid. An OCL guard expression must be defined on each outgoing transition that uses the same trigger, only one of which should evaluate to True. In the case where no triggers are defined all outgoing transitions must have an OCL guard expression.

4 Adding regions

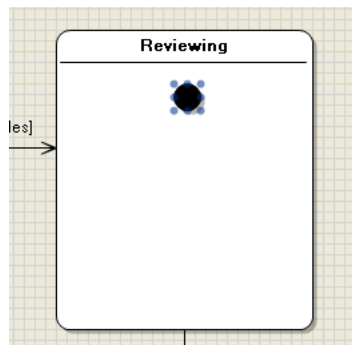
The previous example demonstrated an over simplistic review process. The process consisted merely of indicating that the review was complete by invoking the ReviewComplete trigger; this trigger was only added to prevent the state machine from automatically taking the transition to "Published", which is the default behavior for a state with no trigger.

The following steps will demonstrate how to replace the review process by embedding states within the "Reviewing" state in order to define a kind of embedded state machine. This definition will reflect a more realistic approach to approving an article submitted by an author without the appropriate trust levels by defining the exact steps to take in order to approve or reject the submitted article.

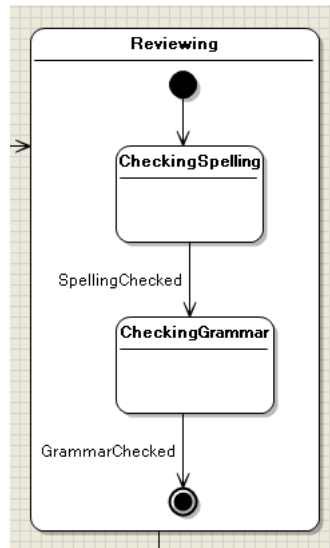
1. On the class diagram rename the UML attribute "State_1" to something more intuitive such as "MainState".
2. Click on the transition from "Reviewing" to "Published" and ensure that the Trigger property is empty.
3. Right-click the "ReviewComplete" trigger method on the Article class and select "Delete".
4. Right-click the Article class and select Add -> State Attribute.
5. Name the new state attribute "ReviewState".
6. Ensure that the type of the attribute is "String".

Note: It is possible to map states attributes to other types, but for simplicity we will continue to use String.

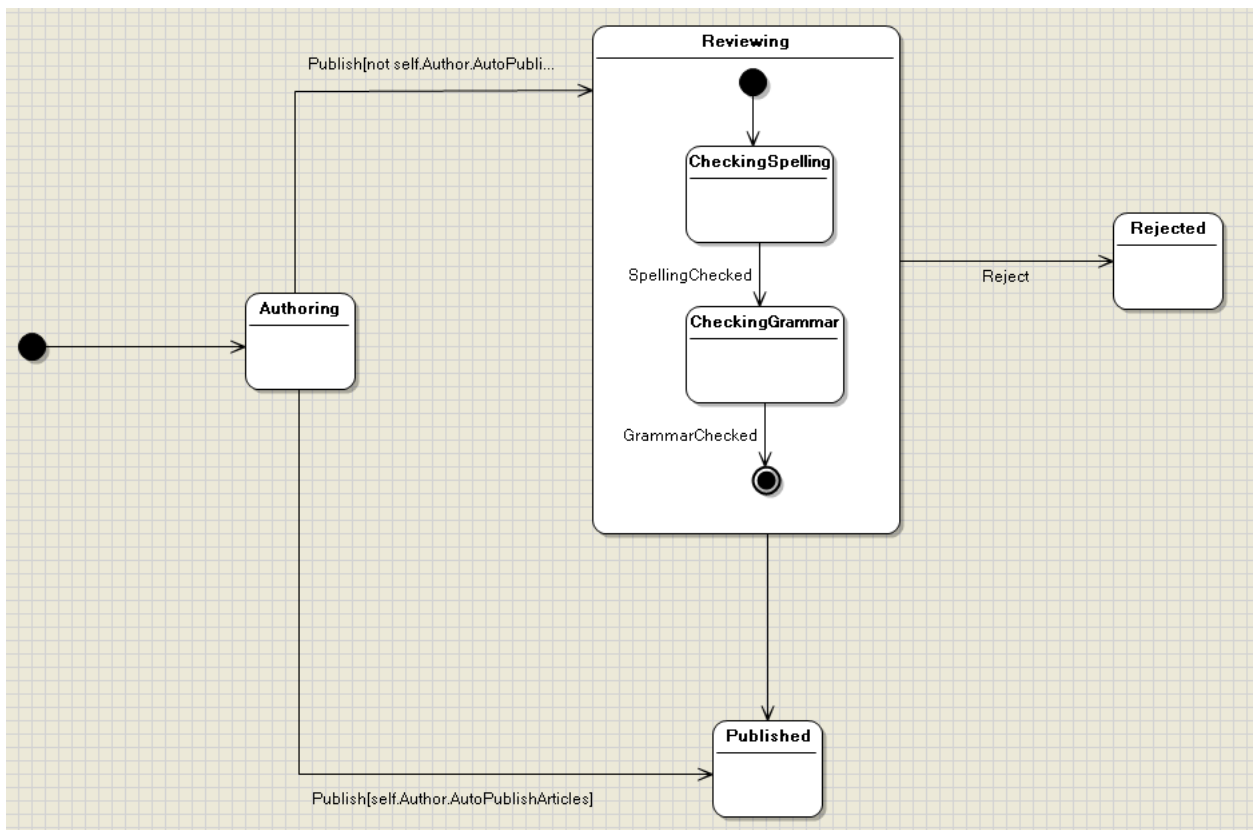
6. Select the client area of the "Reviewing" state, and in the object inspector set its "State attribute" property to the new Reviewing state. If you wish you may also change the name of the region to "ReviewingRegion".
7. Add three more trigger methods to the Article class named "SpellingChecked", "GrammarChecked", and "Reject".
8. On the state diagram select the "Reviewing" state and resize it.
9. Select "Initial" in the Tools Palette and then click the client area within the "Reviewing" state.



9. Now select the "State" item from the Tools Palette and then click inside the client area of the "Reviewing" state to add a new state to the diagram, name it "CheckingSpelling".
10. Add another state within "Reviewing" and name it "CheckingGrammar".
11. Select the "Final" item from the Tools Palette and again add it inside the "Reviewing" state.
12. Add transitions between the states and set the transitions' trigger properties so that the "Reviewing" state looks like the following diagram.

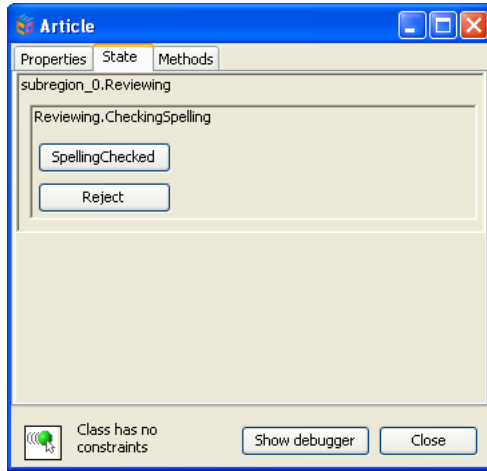


- 13. Now add a new state to the main state diagram (not inside the "Reviewing" state) and name it "Rejected".
- 14. Add a transition from the Reviewing state to the new "Rejected" state and set the transition's trigger to "Reject".



Note: Way points may be added to transition lines by clicking any part of the line and dragging it. To remove a way point drag it and drop it onto the target state.

Executing the quick prototype tool will enable you to test your modified state machine.



5 Entry, Exit, and Effect

There are points within an ECO state machine execution that let you trigger methods on your class. These are

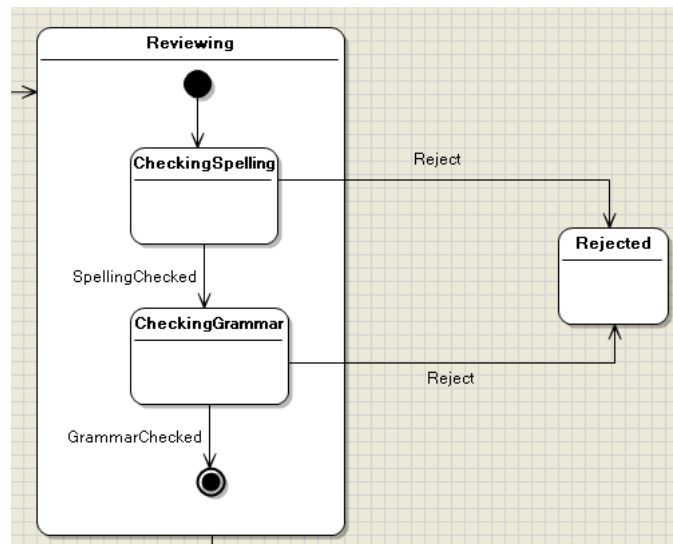
Point	Description
State.Entry	Instructions will be processed when a state is entered.
State.Exit	Instructions will be processed when a state is left.
Transition.Effect	Instructions will be processed when a transition is taken.

These points will now be utilized in order to provide the Author with feedback as to why their article was rejected.

1. Open the project containing the business classes.
2. In the [Model View] tab expand the QuickStartPackage node.
3. Double-click the "Class Diagram" node to edit the diagram in the designer.
4. Right-click the Article class and add a new attribute "ReasonRejected : String", and set the Length of the new attribute to 64.
5. Locate the Article node parented by the QuickStartPackage and expand it.
6. Double-click its child state machine node to open the state machine diagram it in the designer.

Next a rejected reason will be automatically assigned depending on which stage of the reviewing process the article was rejected.

7. Now change the state diagram so that instead of having a single transition from Reviewing to Rejected you instead have two transitions. One from the embedded "Reviewing.CheckingSpelling" state to Rejected and one from the "Reviewing.CheckingGrammar" state to Rejected.



8. Click the transition from CheckingSpelling to Rejected.
9. In the Object Inspector enter the following expression for the "Effect" property

```
self.ReasonRejected := 'Spelling'
```

Note: There is no terminating semi-colon.

- Repeat these steps for the transition from CheckingGrammar to Rejected but specify the reason 'Grammar' instead of 'Spelling'.

Next the Exit action for the Authoring state will be used to clear the rejected reason, and transitions will be added to enable the author to resubmit their article.

- Click the Authoring state to select it in the Object Inspector.

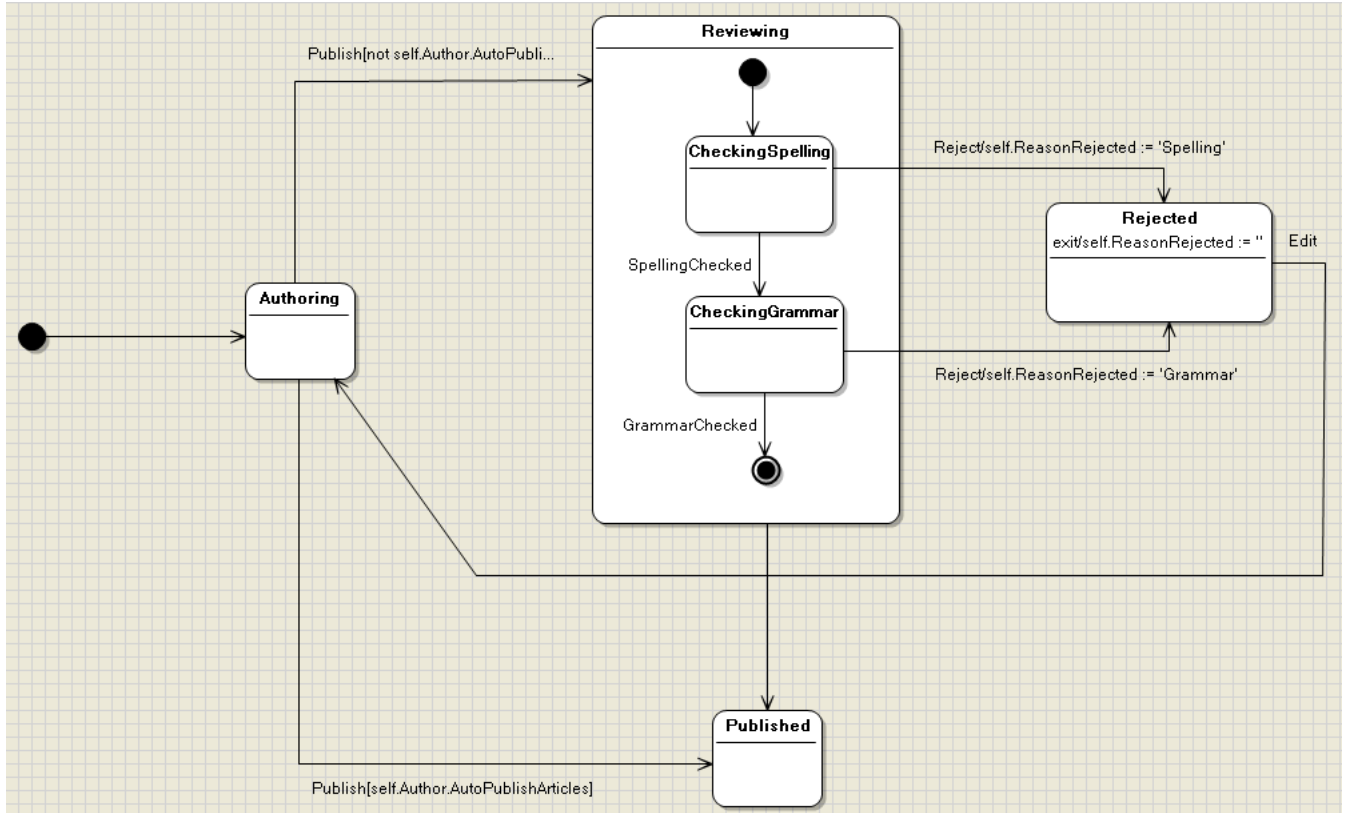
- Set the "Exit" property to the following expression. This will ensure that whenever the Article is re-edited there will be no ReasonRejected text.

```
self.ReasonRejected := ''
```

- To save time opening the Class Diagram just to add another new method you may also add elements to a class by right-clicking its node in the [Model View] tab. Do this now and add a new Trigger named "Edit".

- Select the "Transition" item in the Tool Palette and draw a transition from "Rejected" back to "Authoring".

- In the Object Inspector select "Edit" for the transition's Trigger property.



The behavior we now have will inform the author as to why their article was rejected, and will also allow them to edit their

article and later resubmit it for publishing. If the reviewer wants more control over the text in the reason for rejecting the article it is a simple task to add a new transition from Reviewing to Rejected with a new trigger named "RejectWithReason". The RejectWithReason trigger method would have a single parameter "ReasonRejected: String" and the effect of the transition would be:

```
self.ReasonRejected := ReasonRejected
```

Note: It is possible to embed sub state machines within the already embedded CheckingSpelling and CheckingGrammar states, in fact you can just keep going! It is also possible to have concurrent regions within a state, so the Reviewing state could have a ContentReview region (as outlined above) and a FinancialApproval region where someone from the finance department decides how much the article is worth or may reject the article for being too expensive. Both of these regions would execute concurrently, so the finance department would be able to work at the same time as the review department.

6 Summary

This article has introduced ECO state machines; consisting of states, transitions, triggers, and regions. ECO state machines provide a lot of power to your business layer in a way that is very easy to implement.

Index

A

Adding a state machine to the Article class 3

Adding regions 9

E

Entry, Exit, and Effect 12

O

Overview 1

P

Prerequisites and goals 2

S

Summary 15