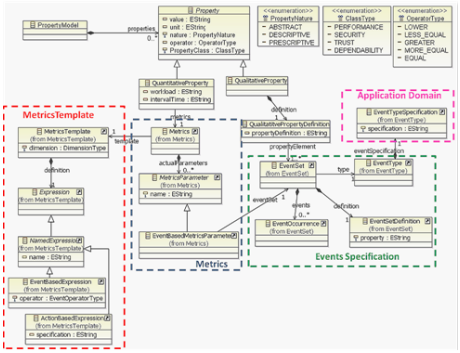

CONNECT Property Meta-Model

User Guide

March 18, 2011

Contents

List of Figures

Chapter 1

Introduction

The Connect Property Meta-Model (CPMM) is an eCore model developed into the Eclipse EMF framework [2] for specifying the Connect properties and metrics. It includes: EventType.ecore and EventSet.ecore modeling the event and the eventSet respectively, Metrics.ecore and MetricsTemplate.ecore for specifying the metrics and metricsTemplate concepts and the Property.ecore representing the Property meta-model.

The editor, obtained as an Eclipse Plugin, contains the information of the defined eCore models and allows to create new model instances of the Property, Metrics, MetricsTemplate, EventType and EventSet meta-models.

This tutorial provides a guide for installing and using the CPMM editor.

Chapter 2

Running CPMM editor

Download the CPMM editor from `http://labsewiki.isti.cnr.it/labse/tools/cpmm/public/main`, in the selected destination folder. Into your Eclipse workspace import the project (via `File/New/Import/Existing Projects into Workspace`), then select from your destination folder, the *connectpropertymetamodel* as the root directory.

The following Eclipse projects will be created: *it.cnr.isti.labse.connect.metrics*, *it.cnr.isti.labse.connect.metrics.edit*, *it.cnr.isti.labse.connect.metrics.editor*.

For running the CPMM Editor, select the *it.cnr.isti.labse.connect.metrics.editor* project and run the plugin. Selecting the *plugin.xml* file, right click on it and then press `Launch as Eclipse application`. This should start a new Eclipse runtime application.

Chapter 3

Using CPMM Editor

We will show in this chapter how to use the CPMM Editor.

As a complete example, we show how to generate the models for the latency property required in the system for the terrorist alert scenario. Specifically, the required latency property is: *average time needed by the CONNECTEed system to reach k% guard devices must be at maximum equal to 10 seconds when in the system there are 10 alerts*. For "time needed by the Connected system to reach a set percentage of guard devices" we mean the average latency experienced in the system from the incoming EmergencyAlert message to the reception of a percentage of eAck coming back from the reached guards'devices. The models for the property, metric, metricsTemplate, eventSet and eventTypes associated to this latency property are presented in Deliverable 5.2 [1].

In the new Eclipse application obtained by running the Editor plugin (see Chapter 2), you can model: EventSet, EventType, Metrics, MetricsTemplate and Property (see Figure 3.1). We show in detail in the following sections how to model these elements.

3.1 Create MetricsTemplate model

We start by modeling the *AverageLatencyMetricsTemplate*.

Select `File/New/MetricsTemplate Model` and specify a file name for your model (see Figure 3.2).

You should now see an editor for your *AverageLatencyMetricsTemplate* model. To edit the model use the `Properties View`. You can set the dimension that for this metricTemplate will be *TIME*, a name and a description for this model (Figure 3.3).

Right click on *MetricsTemplate latency_AVG* and pressing `New Child` create a new element (Figure 3.4).

Create a complex mathematical expression by selecting the *AVG* operator and then the *DIFFERENCE* operator that are in turn other Expressions (Figure

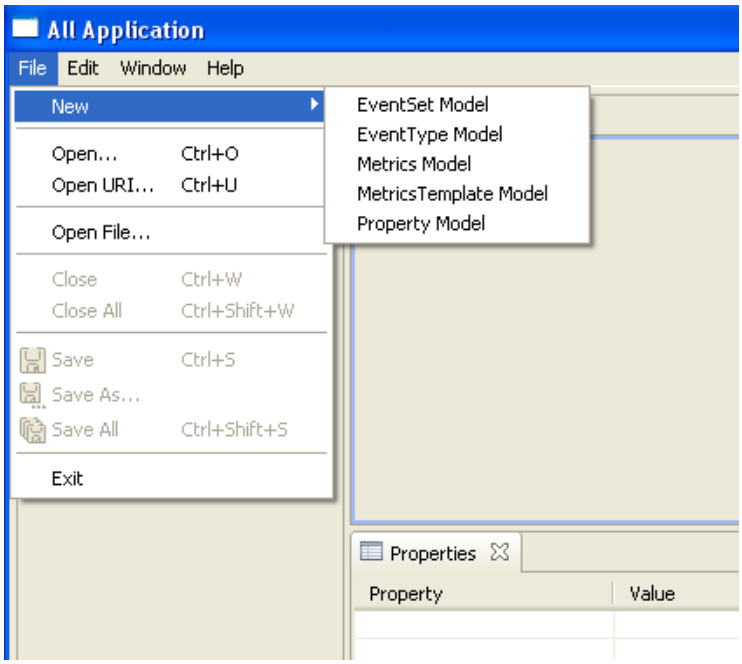

Figure 3.1: CPMM models

3.5).

Create the two operands of the mathematical expression as two event based expressions named x and y with operator *TIMESTAMP* (Figure 3.6).

Specify two TemplateParameters setting the *e1* and *e2* keys respectively. Link *e1* and *e2* to NamedExpression *y* and *x* respectively (Figure 3.7).

Note that, this template model is generic and can be used in different scenarios.

3.2 Create EventType model

You have to define two eventType needed for specifying the required latency property: the *emergencyAlert* and the *SeqOfAck*. Select `File/New/EventType Model` and specify a file name for your model.

The former eventType that you define is a simple eventType named *emergencyAlert* (Figure 3.8). You need to define a specification, that is a label identifying the type or class of the observable events and, one or more parameters and one or more constraints. As shown in Figure 3.9 for the emergencyAlert eventType you have to define the specification *emergencyAlert(IDe)* and one parameter named *IDe* representing the event identifier.

Then, you have to define the *SeqOfAck* eventType (see Figure 3.10). This

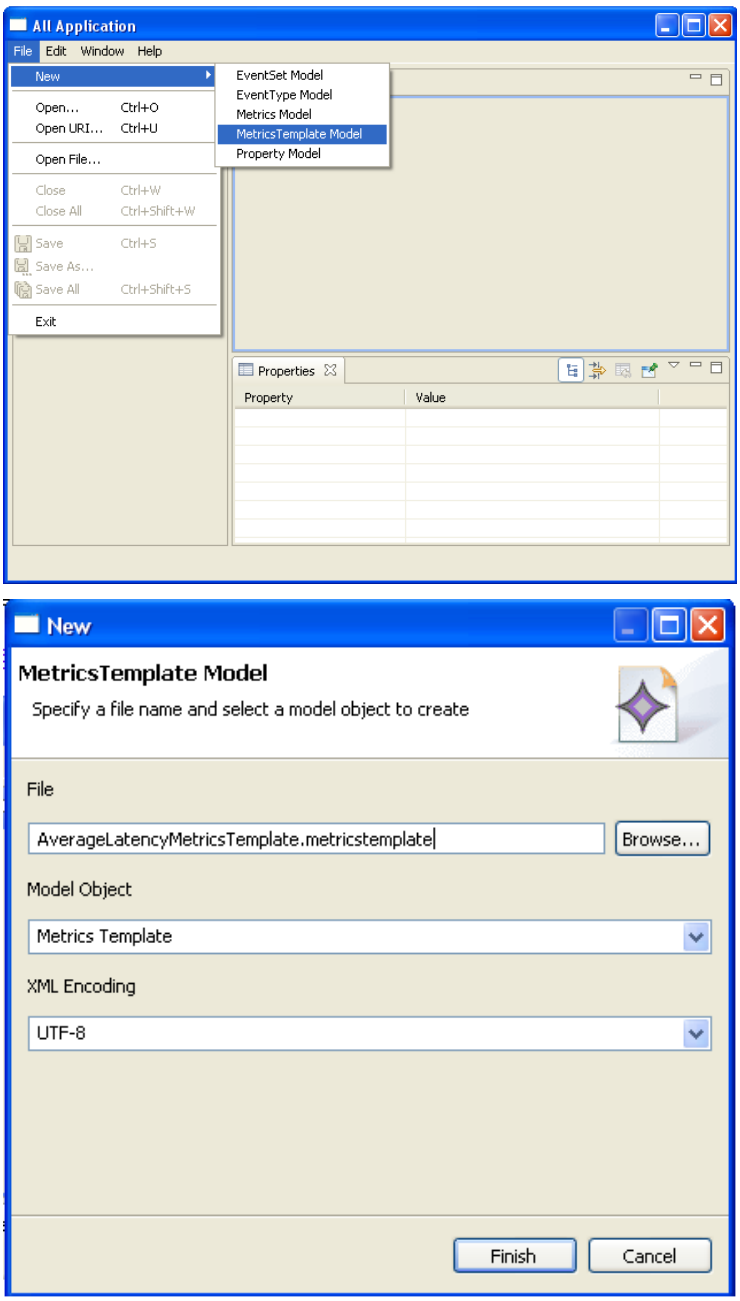

Figure 3.2: Create AverageLatencyMetricsTemplate model

latter eventType is a complex event type since it is defined as a sequence of

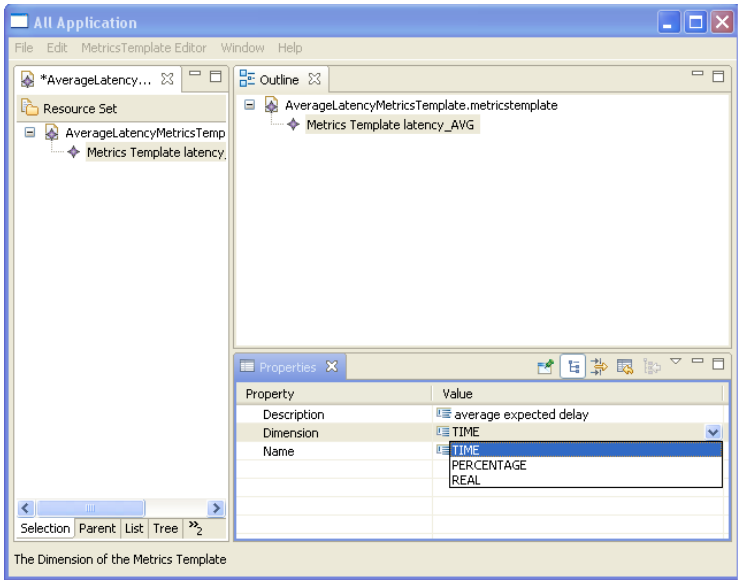

Figure 3.3: Edit AverageLatencyMetricsTemplate model

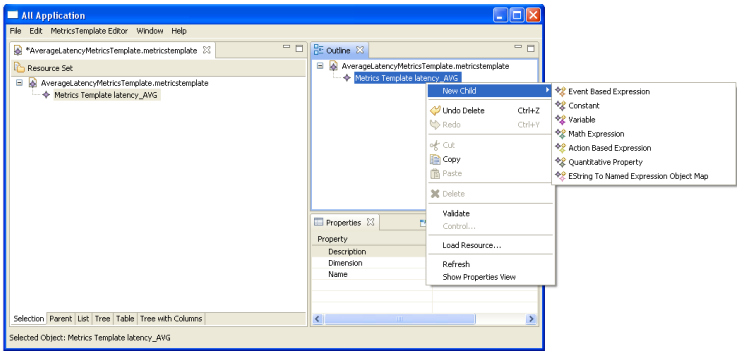

Figure 3.4: Create new elements

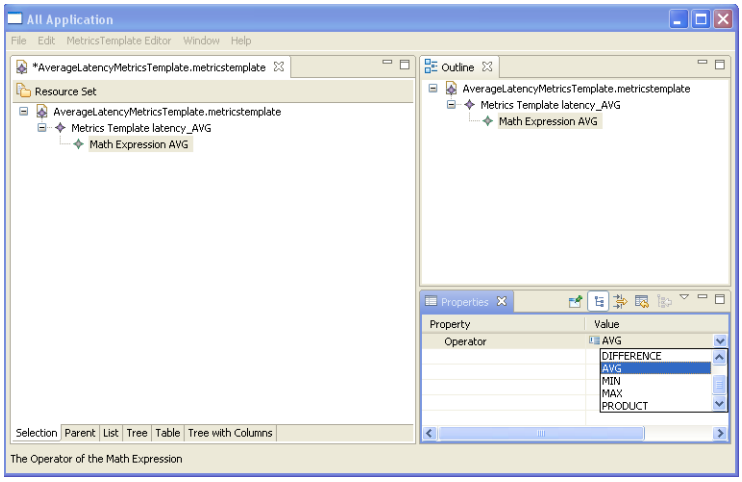

Figure 3.5: Set Math expression

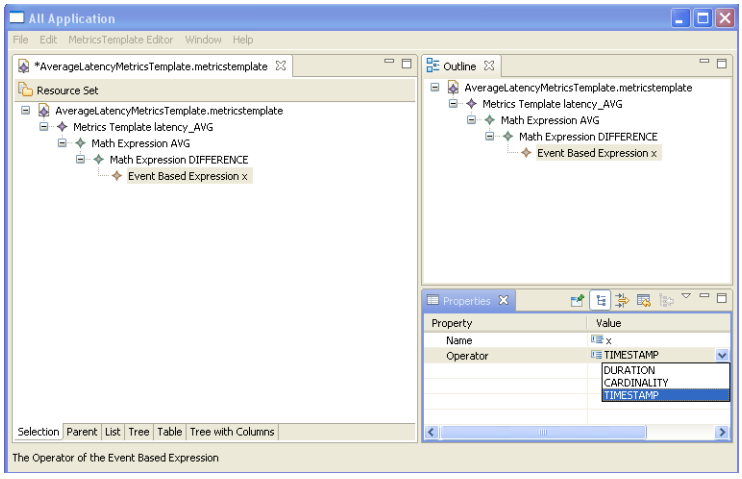

Figure 3.6: Create operand

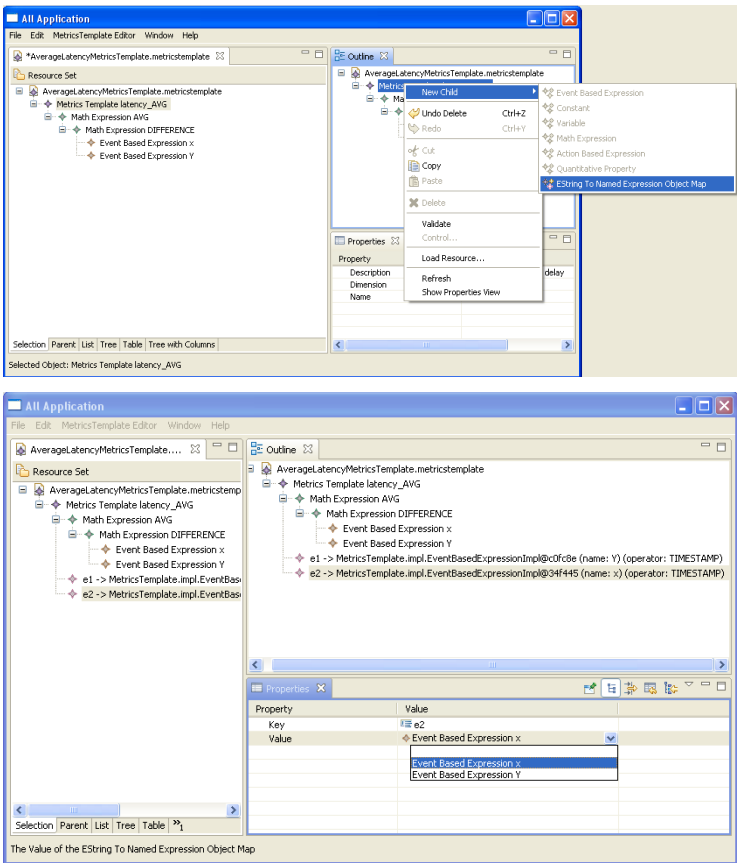

Figure 3.7: Create template parameter

eAck simple EventType. It has two parameters: the *emergencyAlert ID* (named *IDe*) that refers to an emergencyAlert simple eventType and the list of sender identifiers (named *IDgList*) representing the list of guard messages acknowledging an alert message reception. For specifying this complex event you define, by a label, a relation between the sequence of sender identifiers acknowledging an alert message, identified by *IDgList* and the emergencyAlert ID, named *IDe*, representing the *emergencyAlert* the sequence of *eAck* refers to.

3.3 Create EventSet model

You have to define two eventSet named *e1* and *e2* that refer to *emergencyAlert* and *sequenceOfAck* eventType presented in Section 3.2 respectively.

Select `File/New/EventSet Model` and specify a file name for your model.

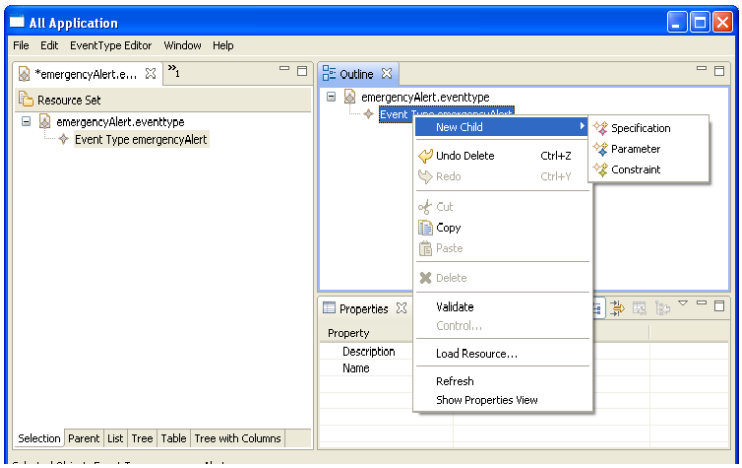

Figure 3.8: Create emergencyAlert eventType

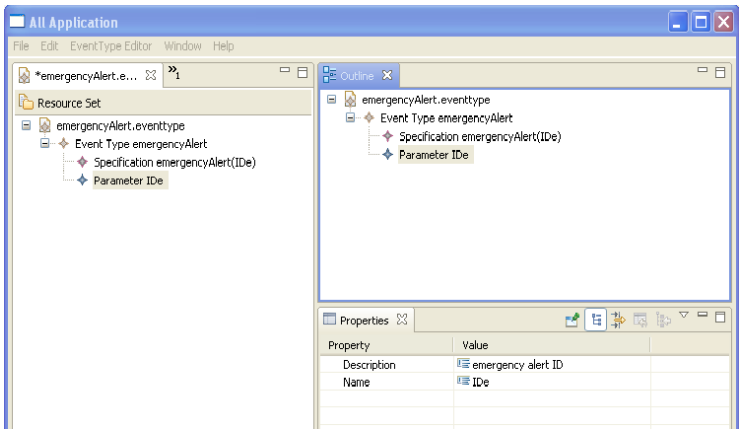

Figure 3.9: emergencyAlert eventType Parameters

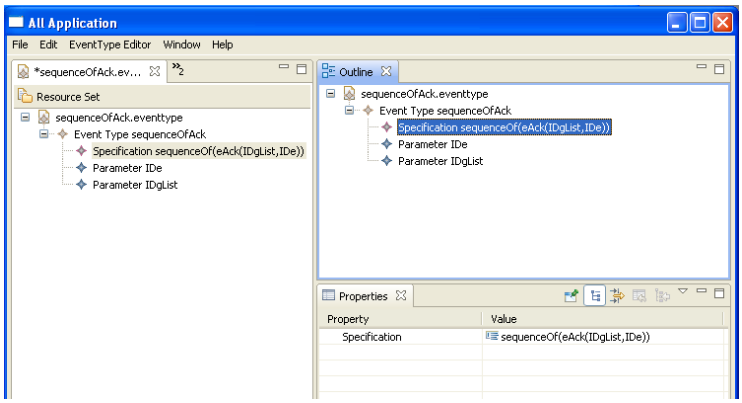

Figure 3.10: Create sequenceOfAck eventType

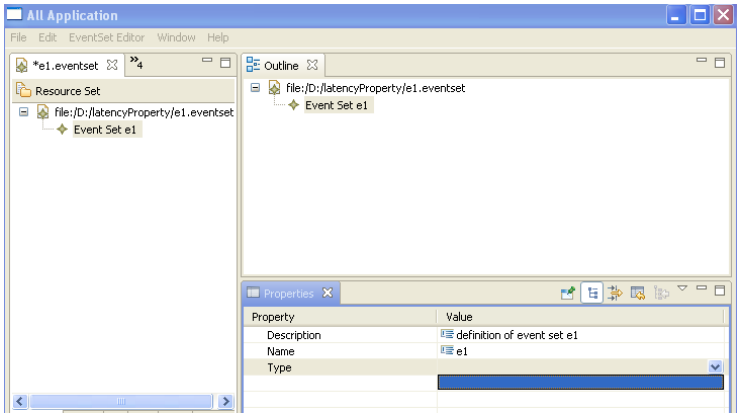

Figure 3.11: Create eventSet e1

For defining the Event set *e1* you need to specify a name, a description and an eventType. Initially the eventType field is empty (see Figure 3.11). You need to load the eventTypes previously defined by right clicking on the eventSet element and then press `Load Resource`. The selected eventType, that in this case is the *emergencyAlert*, will be loaded into the eventSet model (Figure 3.12).

Figure 3.13 shows the *e2* eventSet that refers to *SeqOfAck* eventType. As the *e1* eventSet it has a description, a name and a type. The type of *e2* EventSet is *SeqOfAck* EventType. Moreover, the *e2* eventSet model defines an additional condition specified by the property of the definition element. This condition specifies that *e2* eventSet includes the occurrences of *SeqOfAck* containing at least k *eAck* occurrences referring to the same emergencyAlert message.

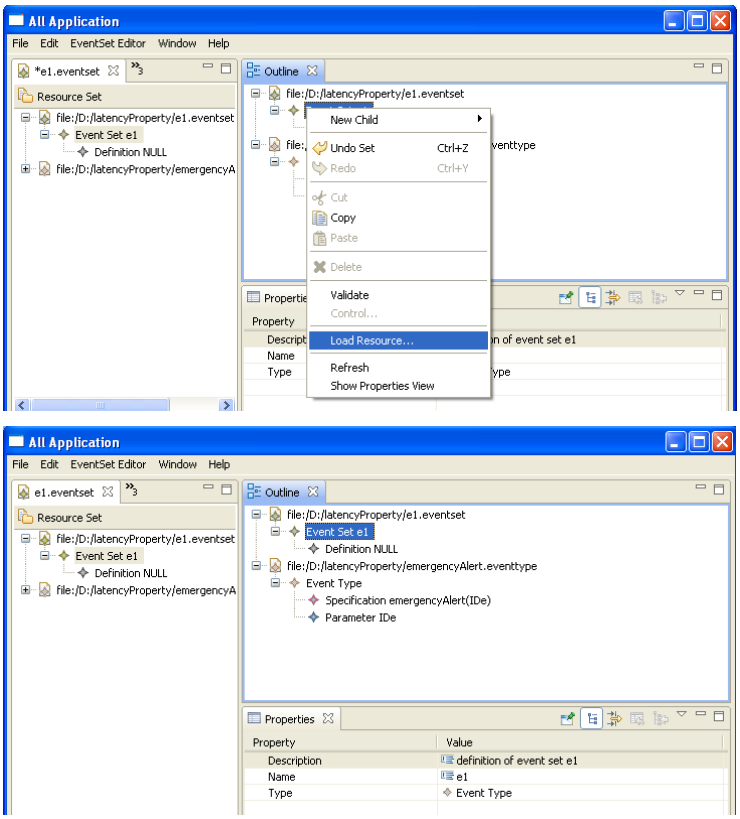

Figure 3.12: Load eventType emergencyAlert

3.4 Create Metrics model

After defining the l*atency_AVMetricsTemplate* and the *e1* and *e2* eventSet, you can define the *LatencyReachingGuard* metric.

Select `File/New/Metrics Model` and specify a file name for your model. In this model you have to define a description, a name and the template that the metric actualizes.

For defining the *LatencyReachingGuard* metric you need to specify a name, a description and a metricsTemplate that the metric actualizes. Initially the template field is empty. You need to load the *latency_AVG MetricsTemplate* defined as in Section 3.1, by right clicking on the *latencyReachingGuard* metric and then pressing `Load Resource`. The selected *latency_AVG MetricsTemplate* will be loaded into the *LatencyReachingGuard* metric model (Figure 3.14).

For actualizing the l*atency_AVG MetricsTemplate* you need to specify two event based metric parameters that link the generic template parameters to the eventSet *e1* and *e2* (defined as in Section 3.3). An event based metric

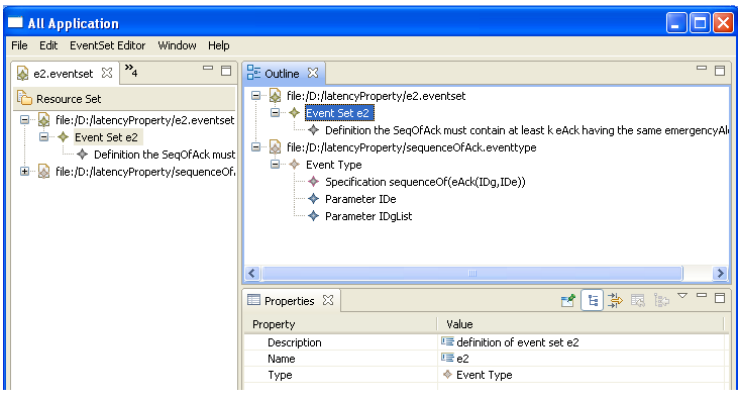

Figure 3.13: Create eventSet e2

parameter has a name and an eventSet that can be loaded into the metric model (Figure 3.15 shows how to load the *e2* eventSet). Then, you have to define a metric constraint specifying that the two event sets *e1* and *e2* must be related to each other and they refer to the same *emergencyAlert*.

3.5 Create Property model

Finally, you can model the required latency property. Select `File/New/Property Model` and specify a file name for your model. As model object select `quantitative property` (Figure 3.16) since the property can be associated to a metric. In particular, the associated metric is the *LatencyReachingGuard* metric, defined as in Section 3.4, that you can load as in Figure 3.17.

You have to specify for this property a name and a description. Moreover, the editor allows you to select: the nature of the property that in this case will be `PRESCRIPTIVE`, an operator that will be `LESS_EQUAL` and a *propertyClass* that will be `PERFORMANCE`. Then, you have to specify a value and a unit for the property and finally the workload. The final model for the *ReachingGuard* Property is shown in Figure 3.18.

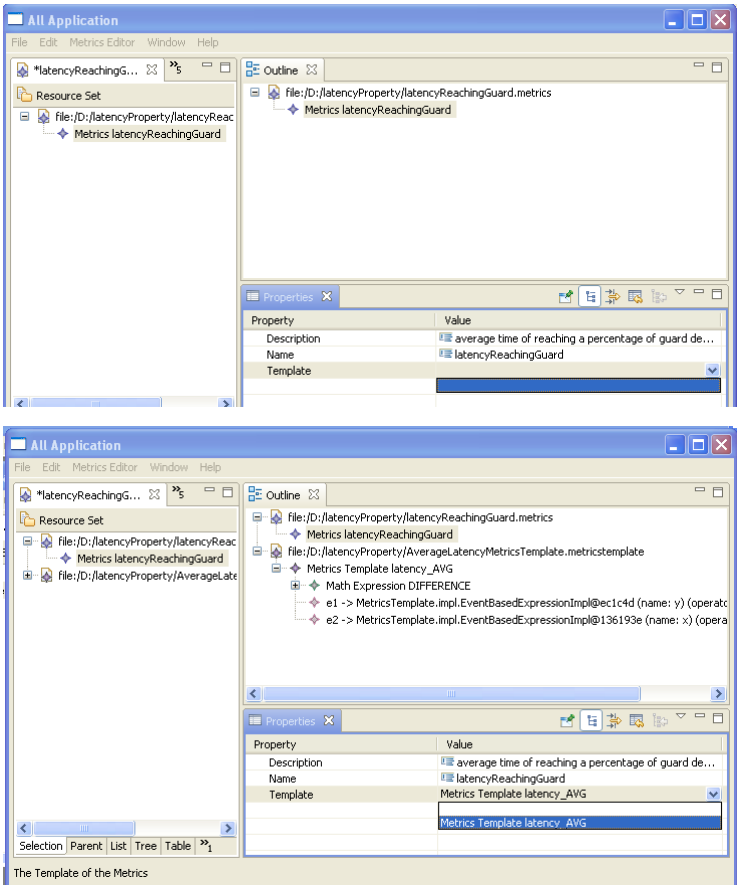

Figure 3.14: Load metricsTemplatelatency_AVG

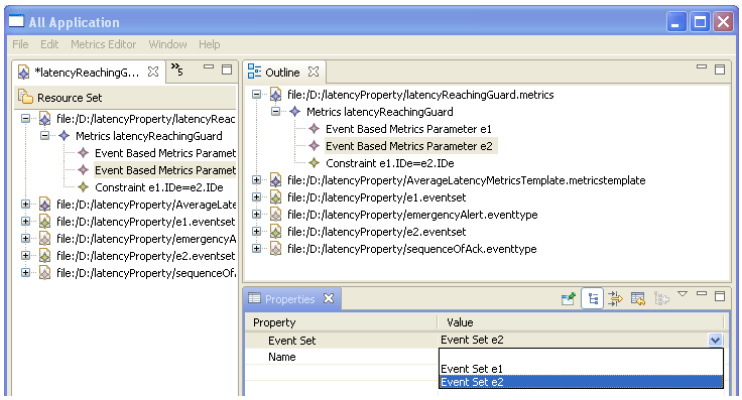

Figure 3.15: Load eventSet e2

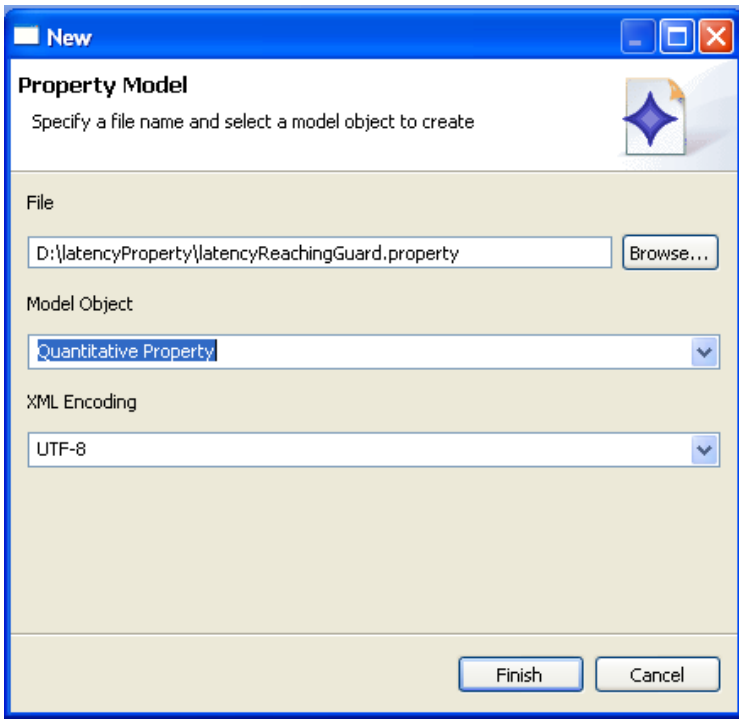

Figure 3.16: Create LatencyReachingGuard property

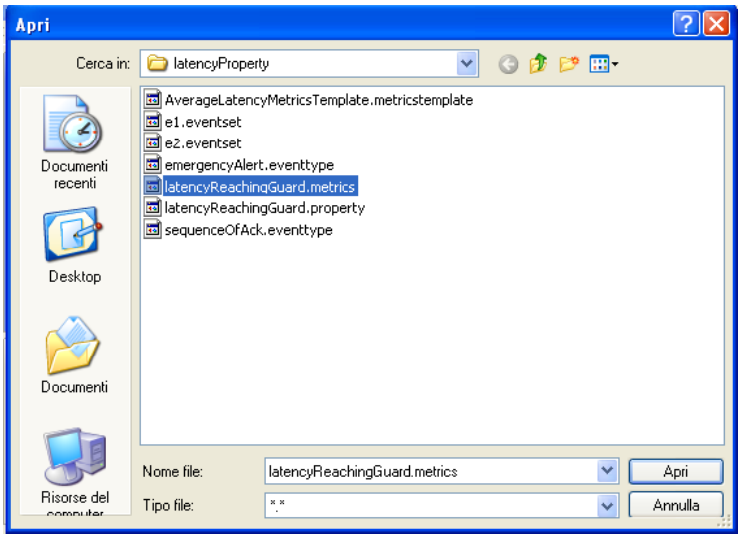

Figure 3.17: Load LatencyReachingGuard metric

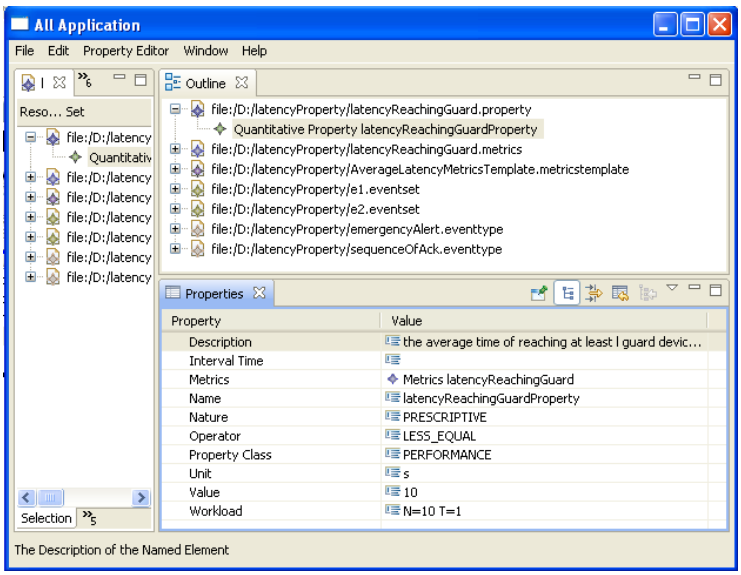

Figure 3.18: LatencyReachingGuard property model

Bibliography

[1] CONNECT Consortium. Deliverable 5.2 – Design of approaches for dependability and initial prototypes, 2011.

[2] Eclipse Modeling Project. http://www.eclipse.org/modeling/.