CAVIAR

Context Aware Vision using Image-based Active Recognition IST- 2001- 37540

D16

Context model specification tool

User Manual

CAVIAR :	D16.1 Context model specification tool : User manual
Date :	20 October 2005
Author(s) :	Patrick Reignier
Work package :	5
Document status :	Version 1.0
Usage :	Internal
Keywords :	Context toolkit, Petri nets, Production rules, Eclipse plugins

Abstract

This document is the user manual of the context modeling tool. The context modeling tool is a graphical tool integrated in the Eclipse IDE. It allows to graphically specify a context model as a set of relations, roles, situations and a graph of situations. This context model can then be compiled into a corresponding recognition program.

Contents

1 Fundamental concepts

The Context model specification tool is an Eclipse Plug-in that let the user graphically edit a context model and transform it into a recognition program. A recognition program is a program interpreting perceptual data and verifying that they correspond to the context model. Before describing the modeling tool, we will first describe the basic concepts.

1.1 Situations

The fundamental concept for our framework is the concept of situation. A situation is defined using a predicate expression. The logical functions that make up this expression are functions of properties observed in the world. Each possible combination of predicates (or their negation) defines a state. A context is a composition of situations that share the same set of roles and relations. Thus a context determines the collection of roles and relations to observe. These are the roles and relations that are relevant to the task of the user.

$$Context(U,T) \Rightarrow \{Role_1, Role_2, \ldots, Role_n; Relation_1, \ldots, Relation_m\}$$

1.2 Roles

A role is potential set of actions within a task. The actions of a role may be enabled by certain entities. When an entity enables the actions of a role, it is said to be able to “play” the role. For example, when a human “actor” walks to the front and addresses a group, he can be said to play the role of “teacher”. When he uses a laser to direct attention to an image, the laser plays the role of pointer. The director determines if an entity can satisfy a role by applying an acceptance test. The system may anticipate (and monitor) such entities based on their properties. Such assignment is provided by a process that applies the test to observable variables. An object may serve as a pointer if it is of a graspable size and appropriately elongated. In the user’s environment, pens, remote controls, and even a wooden stick may all meet this test and be potentially used by the user to serve the role

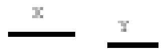
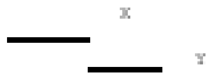
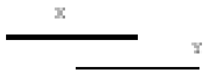
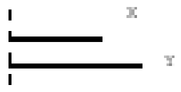
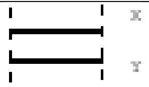
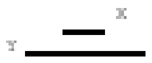
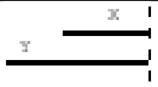
A system to observe the act of pointing as a situation must recognize that any of these objects can play the role of pointer. The set of entities that can provide a role may be open ended. In the users context, the user determines if an entity can satisfy a role for a task by applying the acceptance test. The system may anticipate (and monitor) such entities based on their properties. In the system’s context, the system may assign entities to roles. Such assignment is provided by a process that applies a predicate function defined over entities and their properties.

$$Role(E_1, E_2, \ldots, E_m) \Rightarrow (Role - Class, ID, CF, E_1, E_2, \ldots, E_n)$$

When the test is applied to multiple entities, the most suitable entity may be selected based on a confidence factor, CF. The set of entities is not bijective with the set of roles. One or more entities may play a role. A role may be played by one or several entities. The assignment of entities to roles may (often will) change dynamically. Such changes provide the basis for an important class of events.

1.3 Context

A context model is a graph in which situations are connected by arcs. Theses arcs represent temporal constraints between the situations. They are decorated using the temporal operators defined by Allen [1] :

Relation	*Graphical example*
X before Y	
X meets Y	
X overlaps Y	
X starts Y	
X equals Y	
X during Y	
X finishes Y	

The graph structure is given by the temporal relations. A path inside the graph is the result of the observation of the on-going situations.

1.4 Events

A situation is the assignment of entities to roles, and the relations between these entities. A change in the assignment of entities to roles or a change in the relations between those entities is represented as a change in situation. Such changes in situation constitute an important class of events that we call Situation-Events. A Situation-Event triggers a node changing in the context graph.

1.5 Petri Nets

The context is a graph of situations linked together with temporal relations. The transitions are event-driven and are based on observed Situation-Events. If we associate Situations to Places and Situation-Events to Transitions, the conceptual graph can then be mapped on the *Petri Nets* formalism.

1.6 Software architecture

Specifying a context is part of the development process of a context aware application. In the same manner that GUI builders are part of the integrated development environments, the context specification tool should also be integrated in the IDE. We have selected an open source multi language platform : Eclipse.

1.6.1 The Eclipse framework

Eclipse [2] is an open source IDE mainly written in Java. It is available on many platforms as Windows, MacOS X, Linux and Solaris. Eclipse is :

- an open platform proposing a plug-in mechanism based on the OSGI standard [3].
- a set of plug-ins proposing basic IDE services : editors, property sheets, code browser etc.
- A plug-in for the Java language programming.

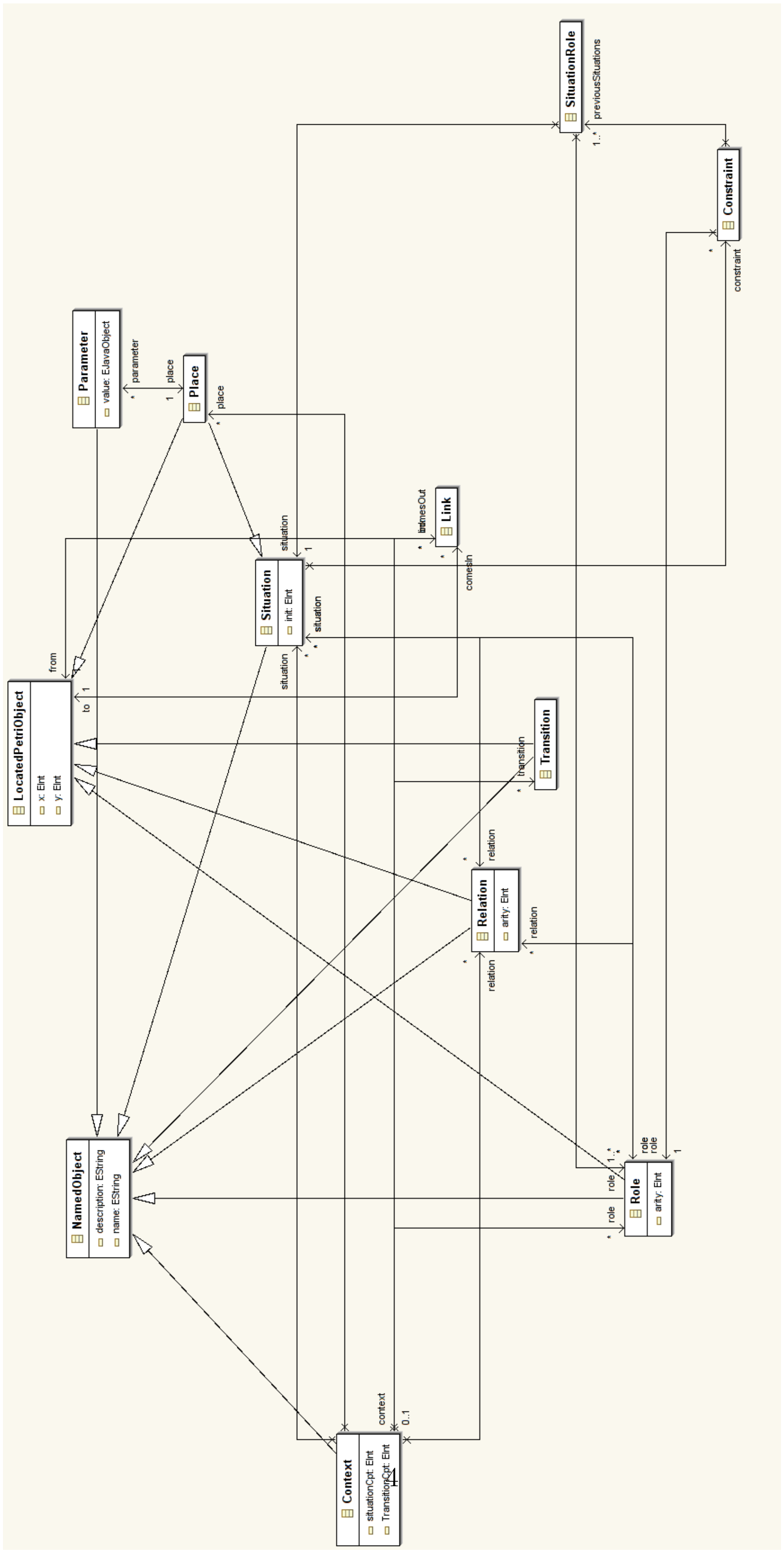
Eclipse functionalities can be extended by downloading new plug-ins like C++ programming environment, GUI builder, UML modeler etc.

1.6.2 Context plug-ins

We have developed several plug-ins to add context functionality to the Eclipse Platform :

1. a context model plug-in.
2. a context editor.
3. context compilers (see section 4

The context model plug-in manages the data structure representing a context (list of roles, relations and graph of situations inside Eclipse. It proposes an API to load and save a context in an XML file (using the XMI format) and access all the elements. More details on this API is given in the programmer's reference document.


SituationRole
previousSituations
Constraint
constraint
Parameter
value: EJavaObject
parameter
place
Place
place
Link
timesOut
comesIn
Situation
init: EInt
situation
LocatedPetriObject
x: EInt
y: EInt
from
to
Transition
transition
Relation
arity: EInt
relation
NamedObject
description: EString
name: EString
Role
arity: EInt
role
Context
situationCpt: EInt
TransitionCpt: EInt
context
0..1
1..*
1

2 Software installation

In this section, we will explain how to get and install the software

2.1 Pre-requisite

The context editor requires two additional plug-ins to work :

- EMF 2.1.0 (Eclipse Model Framework) :
  `http://download.eclipse.org/tools/emf/scripts/downloads.php`
- GEF 3.1 (Graphical Editing Framework) :
  `www.eclipse.org/gef/`

Those two plug-ins must be first downloaded and unpacked in the Eclipse directory.

2.2 Getting the software

The plug-ins are available on the Prima web site [1] :

- *context.zip* : the formal context model.
- *contextEditor.zip* : the graphical context editor plug-in
- *petriCompiler.zip* : the context compiler to Jess rules.
- *hmmCompiler.zip* : the context compiler to Jess rules.

2.3 Installation

1. Unzip the three plug-ins into the Eclipse installation directory.
2. Restart Eclipse using the following command :

   ```
   $ eclipse -clean
   ```

   The `-clean` argument tells Eclipse that new plug-ins have been installed. It is necessary on Windows installation.

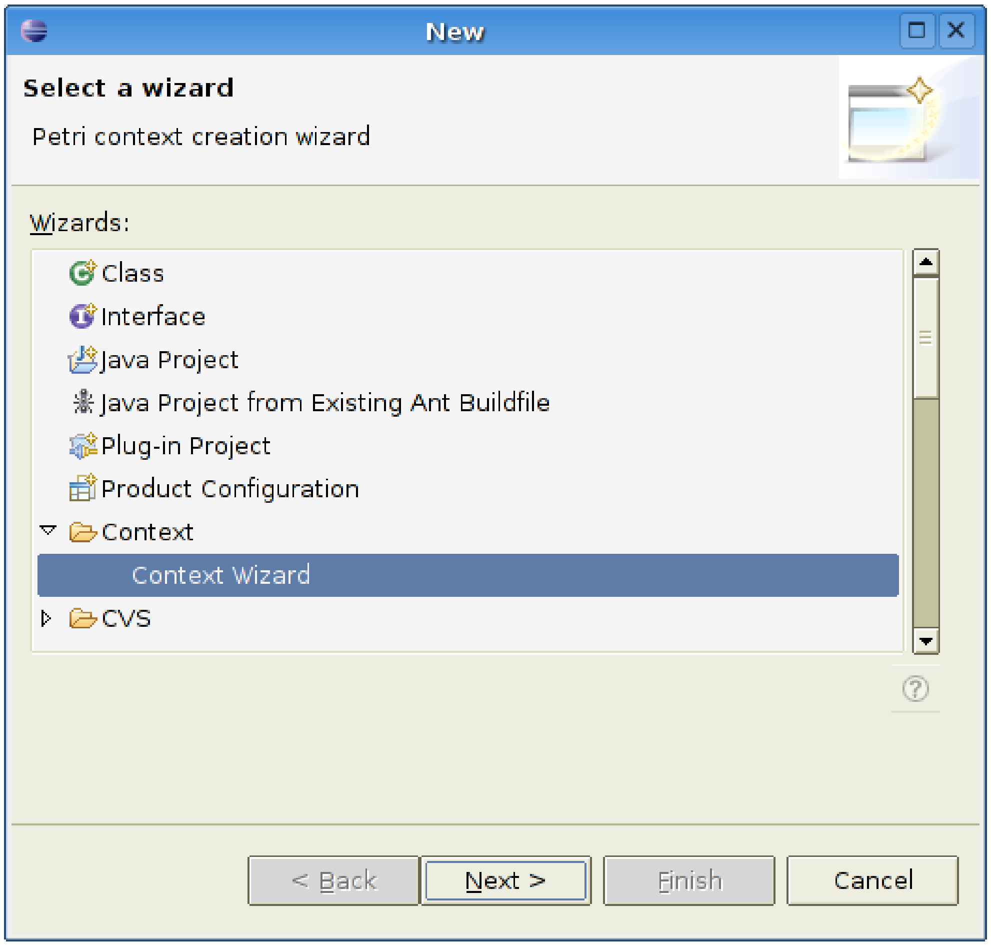
3 Context editor

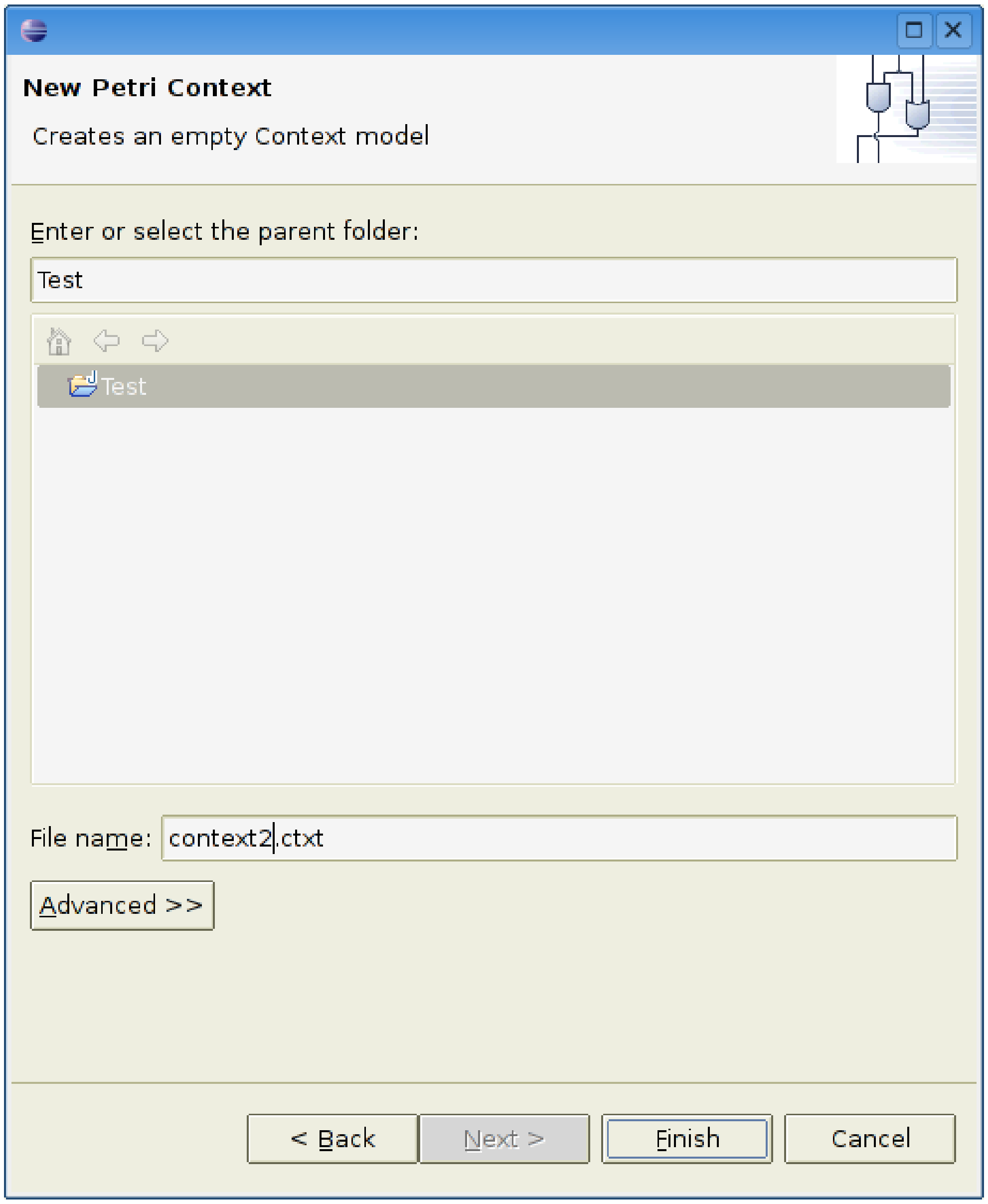
We will present in this section how to use the context editor by illustrating the different steps : creating a new context, defining roles, relations, creating situations based on those roles and relations, and finally, specifying the temporal constraints between situations using transitions and links.

[1] `http://www-prima.inrialpes.fr/reignier/softs/`

3.1 Creating a new context

The "Context Wizard" allows to create a new context file. This context file must be stored into an existing Eclipse project (a Java project for instance).




New Petri Context

Creates an empty Context model

Enter or select the parent folder:

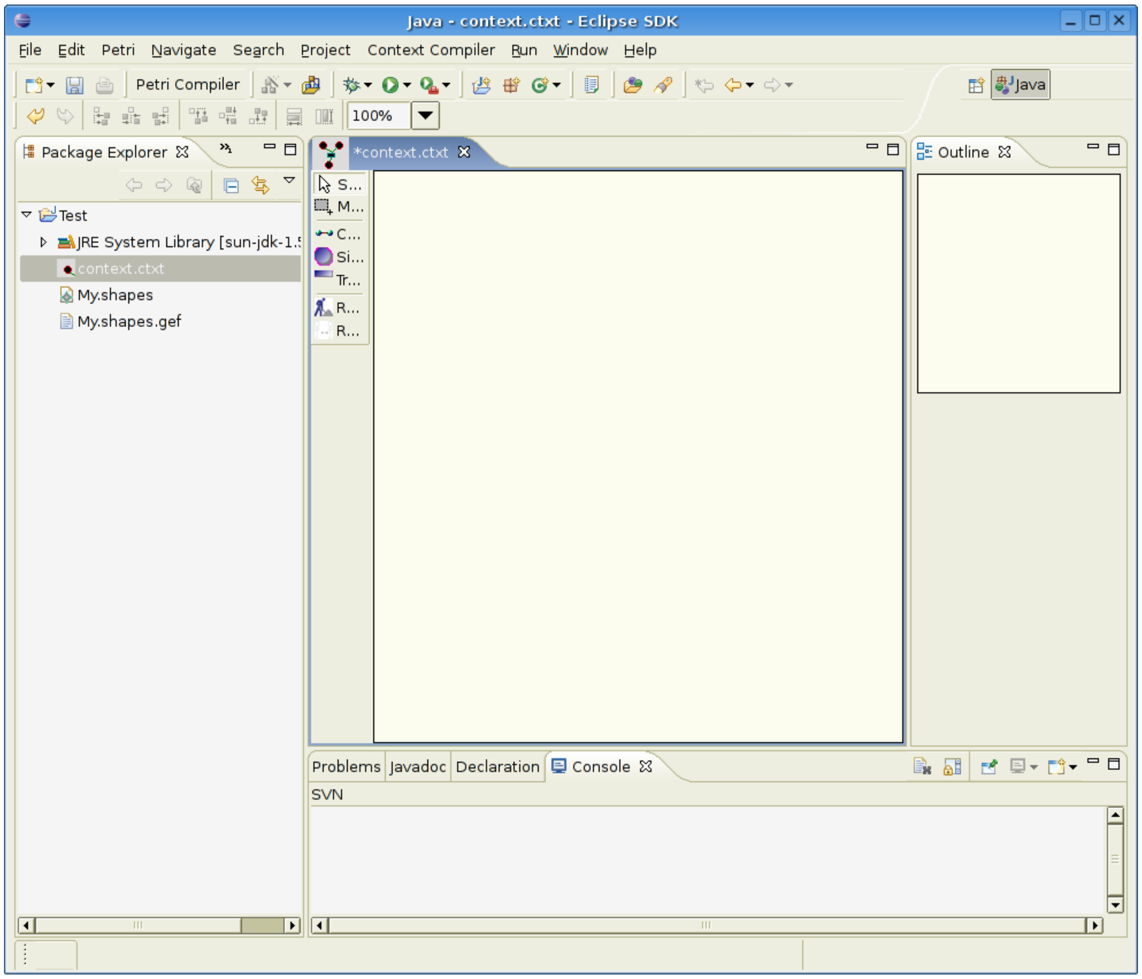
Test

Test

File name: context2.ctxt

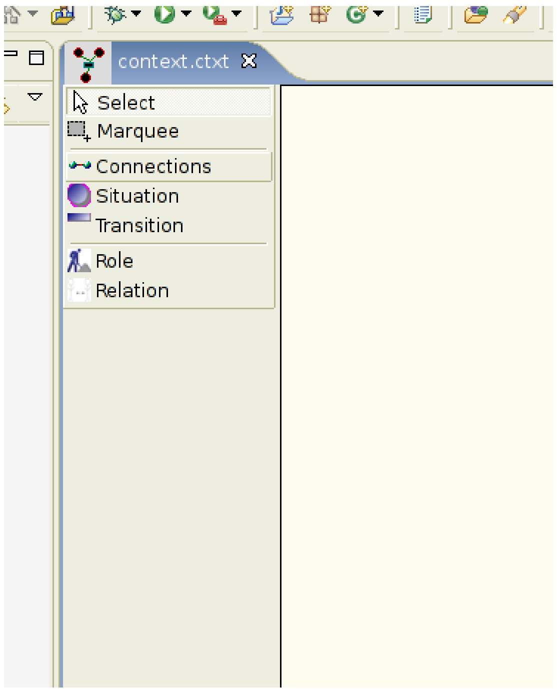
A context file is associated to a context editor. The context editor is composed of three parts

- the toolbar
- the palette
- the graphical editor


Java - context.ctxt - Eclipse SDK
File Edit Petri Navigate Search Project Context Compiler Run Window Help
Petri Compiler
100%
Java
Package Explorer
Test
JRE System Library [sun-jdk-1.5
context.ctxt
My.shapes
My.shapes.gef
*context.ctxt
S...
M...
C...
Si...
Tr...
R...
R...
Outline
Problems Javadoc Declaration Console
SVN

3.2 Context palette

The palette allows to select object, to create roles, relations, transitions and situations. Situations and transitions can be linked to create the situation graph.

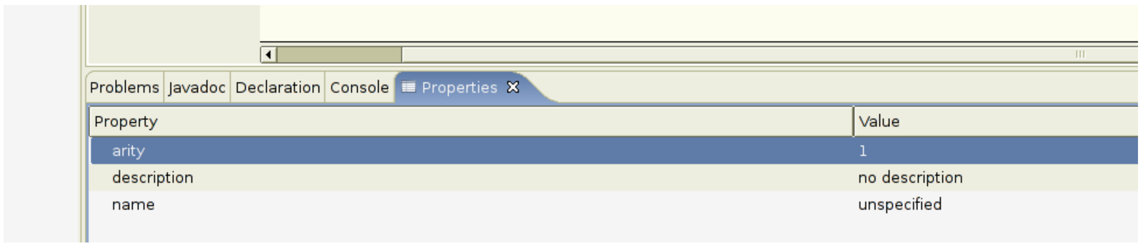


3.3 Roles

To create a new,role, select the role tool in the palette and click in the editor. A default role named "unspecified" is created. The role attributes can be edited in the eclipse *property* window. If this window is not yet opened in your environment, select it in the *Window* → *Show view* → *property* menu. The role properties that can be edited are

- *arity*. A role is a predicate used to select entities. The arity indicates how many entities are necessary to define this role. The usual value is 1.
- *description*. It is a textual value describing the role.
- *name*. The name of the predicate. This name is used in the code generation modules.

A role can be deleted using the contextual menu (right mouse click)



Property	Value
arity	1
description	no description
name	unspecified

3.4 Relations

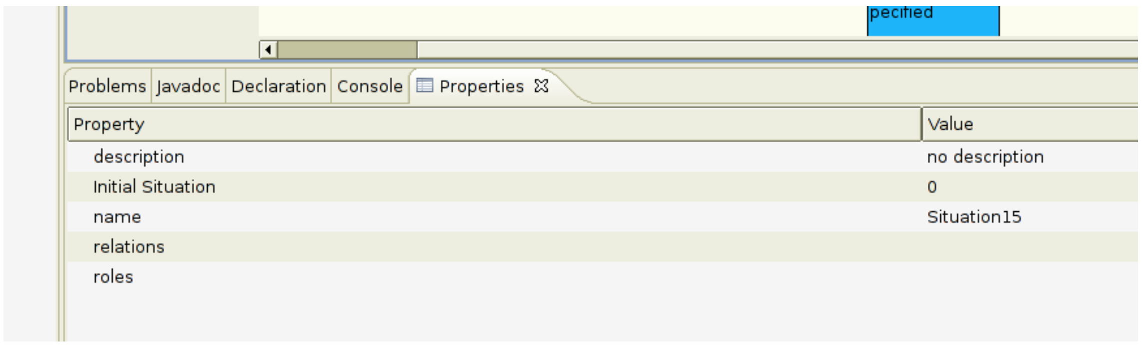
To create a new relation, select the relation tool in the palette and click in the editor. A default relation named "unspecified" is created. The relation attributes can be edited in the eclipse *property* view. If this window is not yet opened in your environment, select it in the *Window* → *Show view* → *property* menu. The relation properties that can be edited are :

- *arity*. A relation is a predicate between entities. The arity indicates how many entities are in relation. .
- *description*. It is a textual value describing the role.
- *name*. The name of the predicate. This name is used in the code generation modules.

A relation can be deleted using the contextual menu (right mouse click)

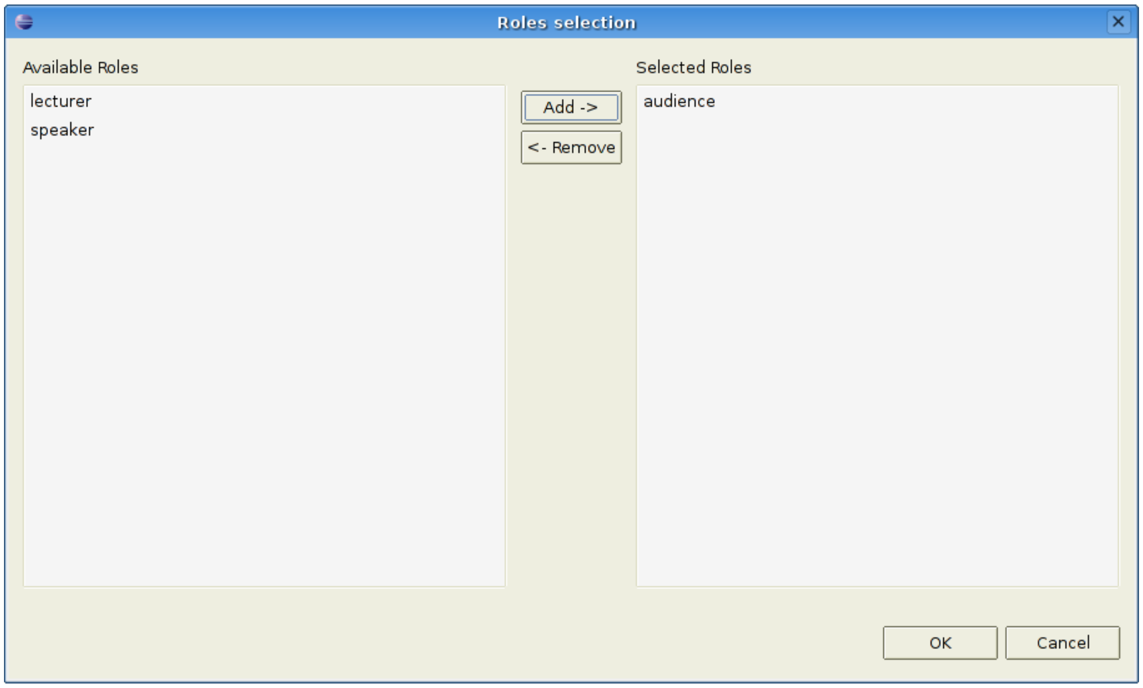
3.5 Situations

To create a situation, select the situation tool in the palette and click in the editor. A new situation is created with an automatically generated name. The relation attributes can be edited in the eclipse *property* view. If this window is not yet opened in your environment, select it in the *Window* → *Show view* → *property* menu. The situation properties that can be edited are :



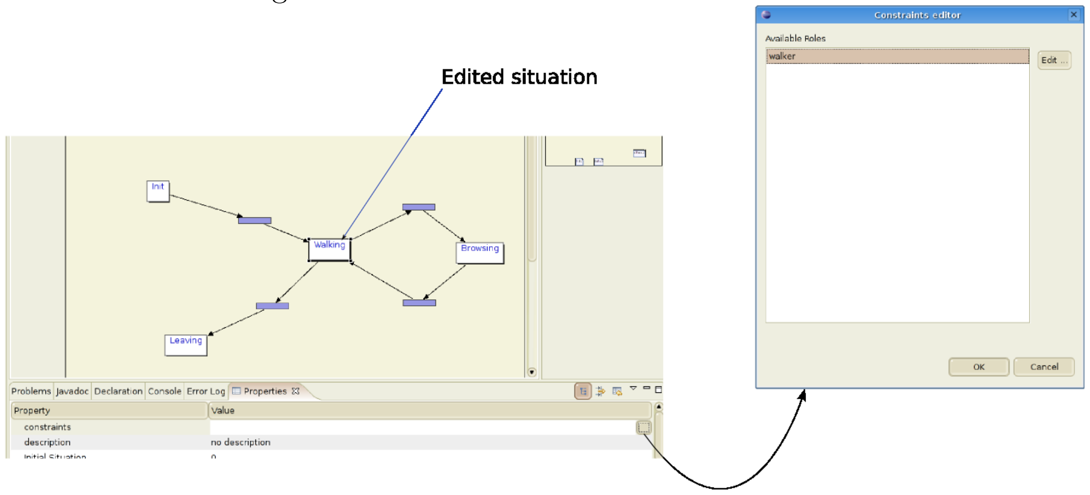
Property	Value
description	no description
Initial Situation	0
name	Situation15
relations	
roles	

- *name* : the name of the situation.
- *roles* : the list of roles associated to this situation. The role selection is done through a dialog box showing all the available roles in the current context.

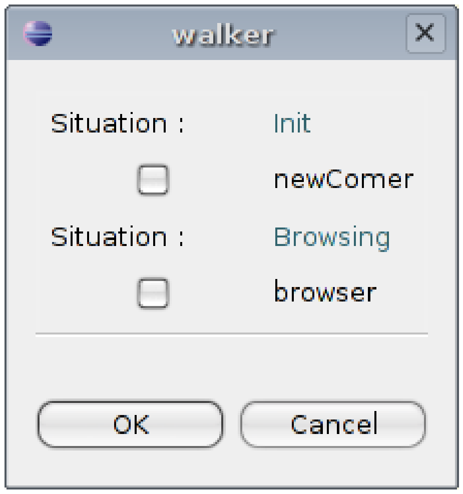


- *relations* : the list of relations associated to this situation. The relation selection is done through a dialog box showing all the available relations in the current context.

- *constraints* : it is possible to specify that some roles of two following scenarios must share the same entities. For instance, a browsing scenario specify a walking situation followed by a browsing situation. We can specify that it must be the same person playing the walker and browser role. Clicking in the constraint field on the property sheet opens a dialog box. This dialog box allows to select a role in the current situation : the sl target role.



Clicking the edit button will open a new dialog box This dialog box displays for every situation connecting to the currently edited one the associated roles. The user can select those that must share the same entities with the target role.


walker

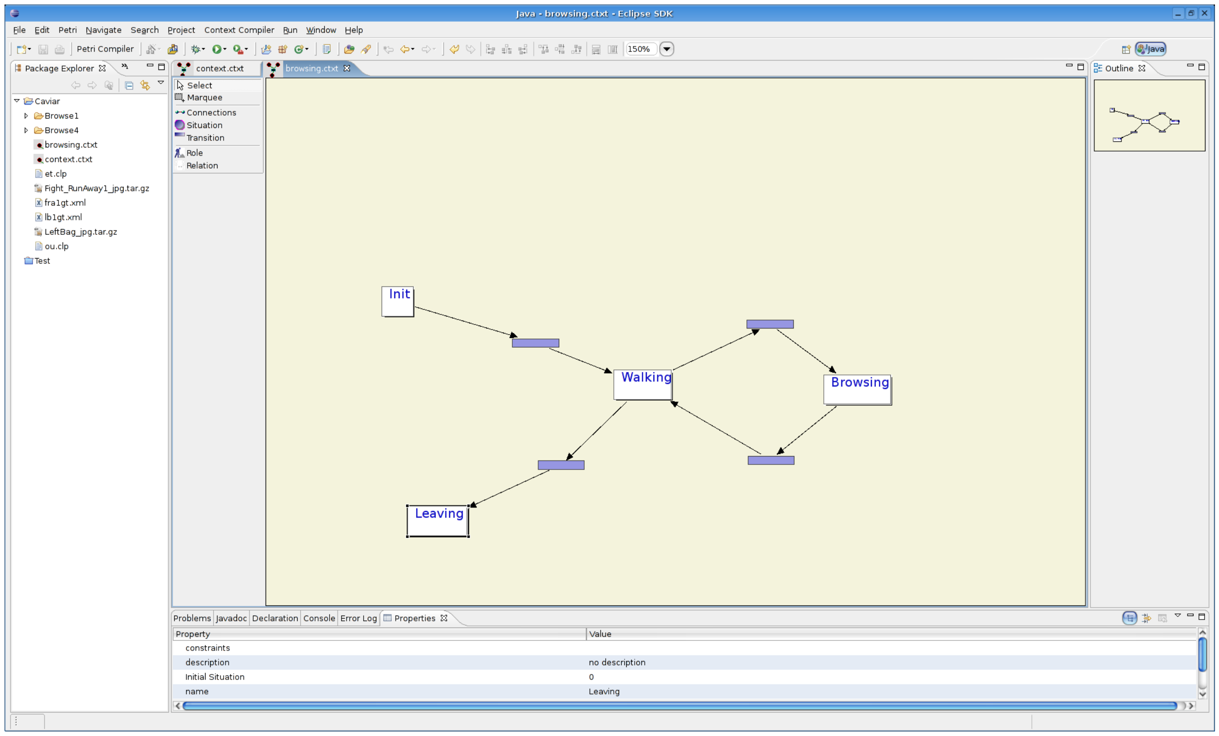
Situation :	Init
☐	newComer
Situation :	Browsing
☐	browser

- *description* : a textual description of the situation
- *initial situation* : a boolean indicating if this situation is the first situation of the currently edited context.

A situation can be deleted using the contextual menu (right mouse click)

3.6 Graph construction

As we have seen in section 1.5, the context is a Petri Net where places are situations. In a Petri Net, places are linked together through transitions. Transitions can be created using the *transition* tool in the editor palette.



Property	Value
constraints	
description	no description
Initial Situation	0
name	Leaving

4 Code generation

The situation graph built with the editor is a formal model of a context. This model must be transformed into a corresponding program that will have to analyse the incoming perceptual events to determin if they are "compatible" with the model. This compilation is a model transformation : from a conceptual model to an executable model. We provide two initial compilers :

1. A Petri to Synchronized Petri net model transformation, producing a Jess set of rules.
2. A Petri to HMM transformation, producing a Java program.

Each compiler is a plug-in (`petriCompiler.zip`, `hmmCompiler.zip`) that must be installed on the eclipse distribution. The compiler can be started on the current context model using the *Context Compiler* menu on the menu bar.
Details can be found in the "Context Compiler reference manual" on how to write a new context compiler.

References

[1] J.F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 13, 1984. 2

[2] Eclipse. http://herzberg.ca.sandia.gov/jess/. 3

[3] The osgi service platform - dynamic services for networked devices. http://www.osgi.org/. 3