8th DSM Workshop

Undoing Operational Steps of Domain-Specific Modeling Languages

Tim Hartmann, Daniel A. Sadilek
Humboldt-Universität zu Berlin

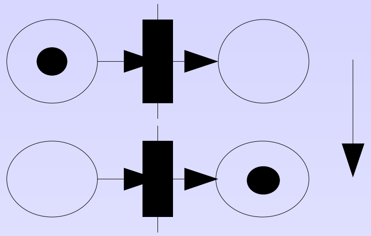
Outline

- Introduction
  - Development of executable DSMLs
  - Animated execution
  - Operational semantics
- Undoing operational steps
- Open issues
- Conclusion

Use Case: DSML Development

Carl Adam Petri

Draft



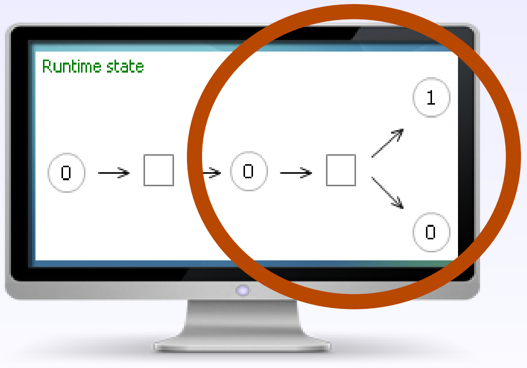
Language engineer

Prototype

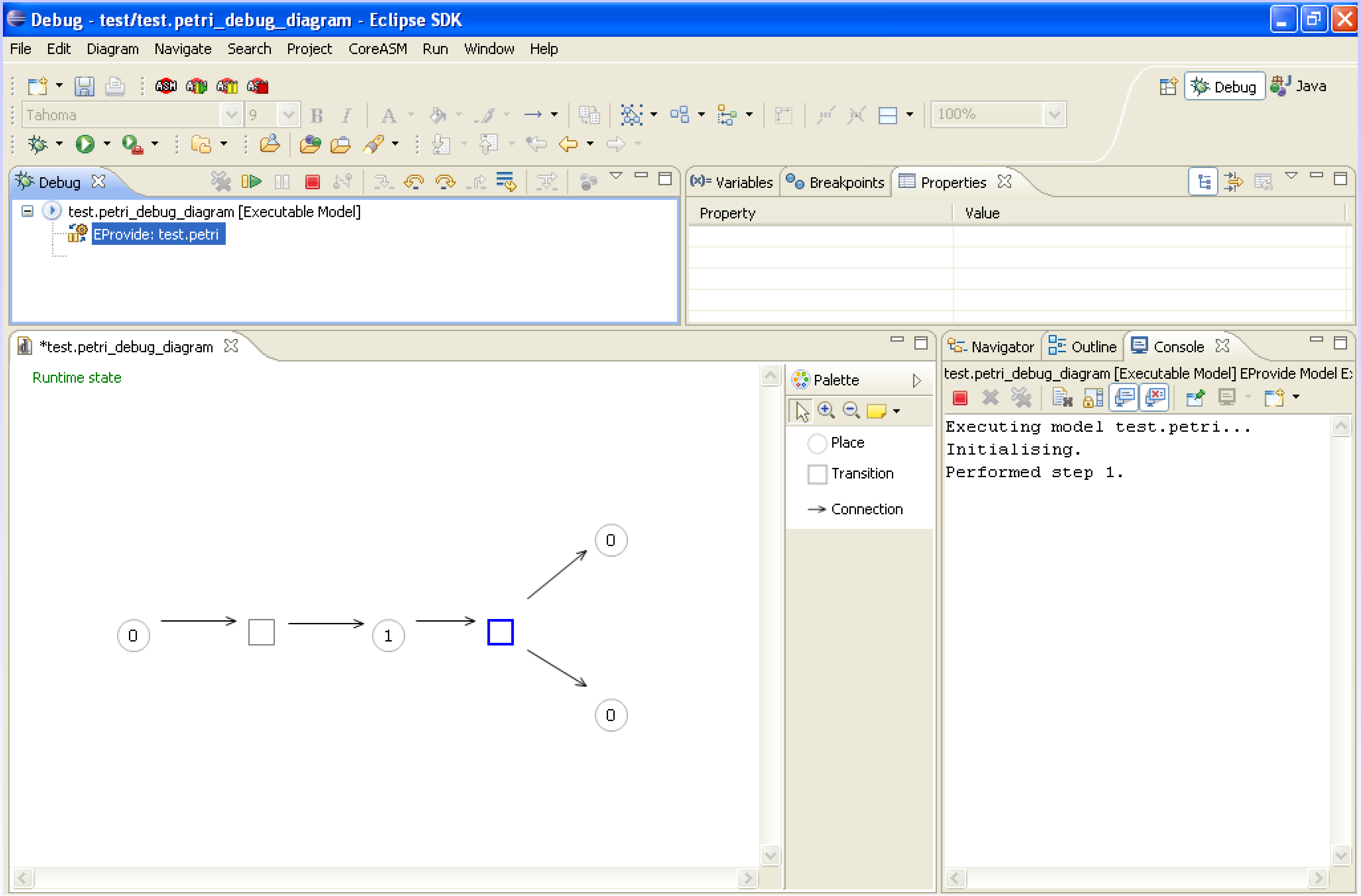
- Metamodel
- Graphical editor
- Operational semantics

Iterative development cycle

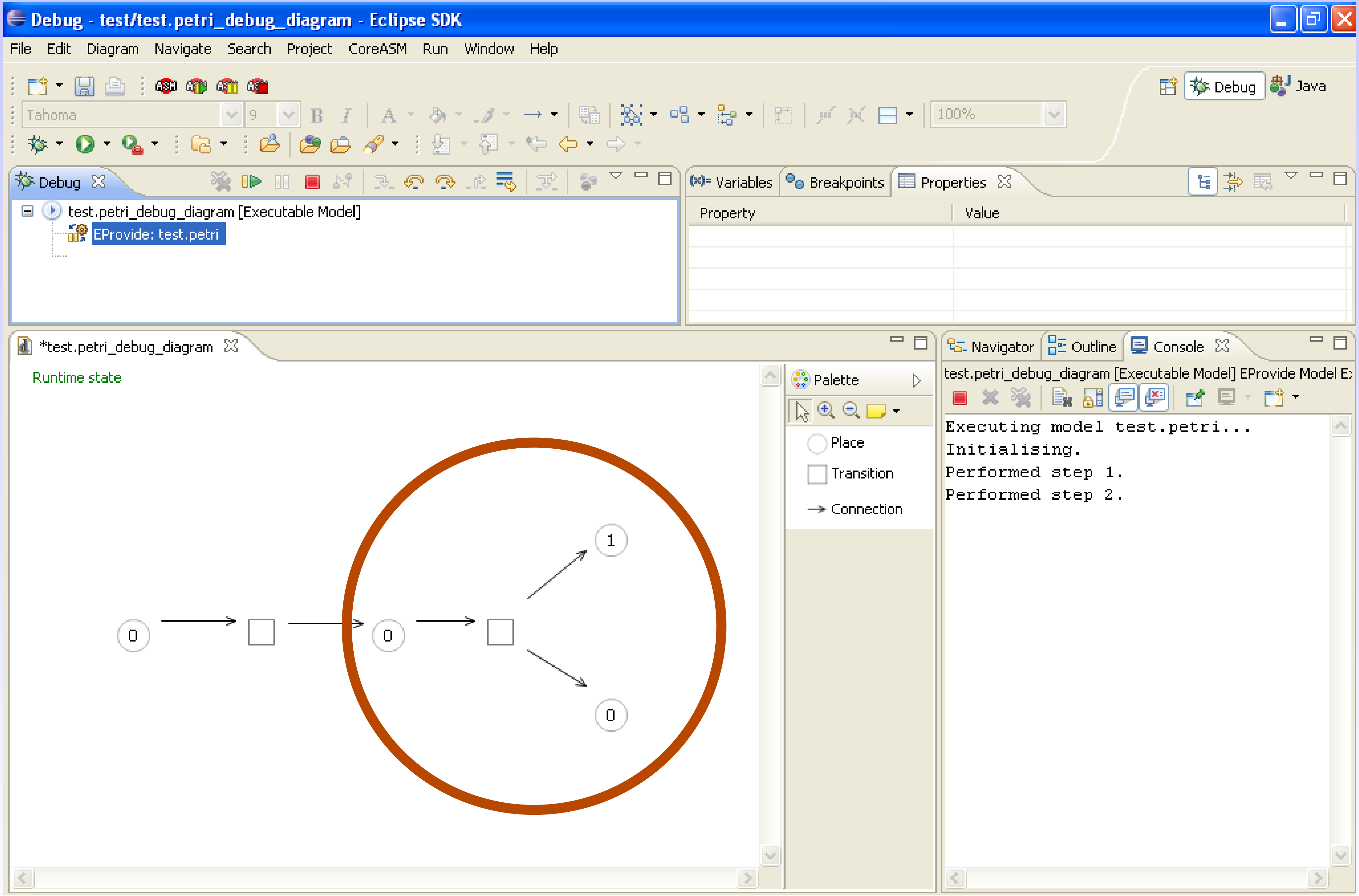
Evaluation



Error


Debug - test/test.petri_debug_diagram - Eclipse SDK
File Edit Diagram Navigate Search Project CoreASM Run Window Help
Debug
Java
Debug
test.petri_debug_diagram [Executable Model]
EProvide: test.petri
Variables
Breakpoints
Properties
Property
Value
*test.petri_debug_diagram
Runtime state
0
1
0
0
Palette
Place
Transition
Connection
Navigator
Outline
Console
test.petri_debug_diagram [Executable Model] EProvide Model E

```
Executing model test.petri...
Initialising.
Performed step 1.
```

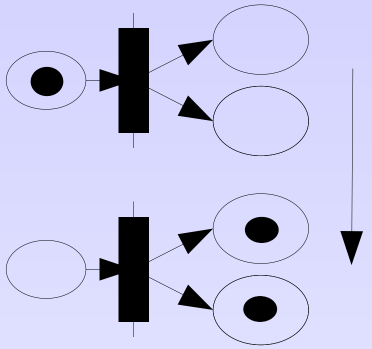

Debug - test/test.petri_debug_diagram - Eclipse SDK
File Edit Diagram Navigate Search Project CoreASM Run Window Help
Debug
Java
Tahoma
9
100%
Debug
test.petri_debug_diagram [Executable Model]
EProvide: test.petri
Variables
Breakpoints
Properties
Property
Value
*test.petri_debug_diagram
Runtime state
0
0
1
0
Palette
Place
Transition
Connection
Navigator
Outline
Console
test.petri_debug_diagram [Executable Model] EProvide Model E:

```
Executing model test.petri...
Initialising.
Performed step 1.
Performed step 2.
```

2nd Iteration of DSML Development Example

Draft

Carl Adam Petri



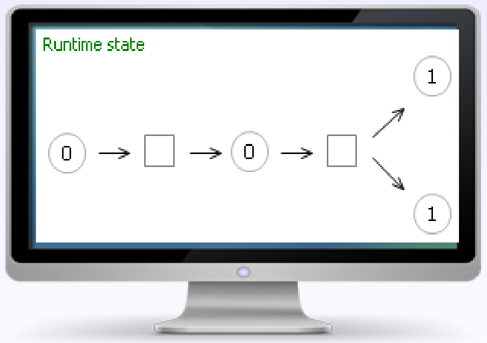
Language engineer

Prototype

- Metamodel
- Graphical editor
- Corrected operational semantics

Iterative development cycle

Evaluation

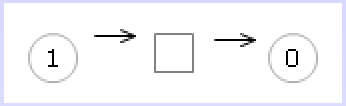


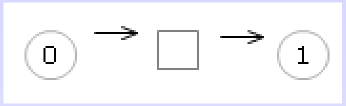
Operational Semantics

- Interpretable operational semantics
- Transition system: $\langle \Gamma, \rightarrow \rangle$
- Configurations: Γ
- Transition relation: $\rightarrow \subseteq \Gamma \times \Gamma$

Operational Semantics

- Configurations are represented as models:

  { 

, 

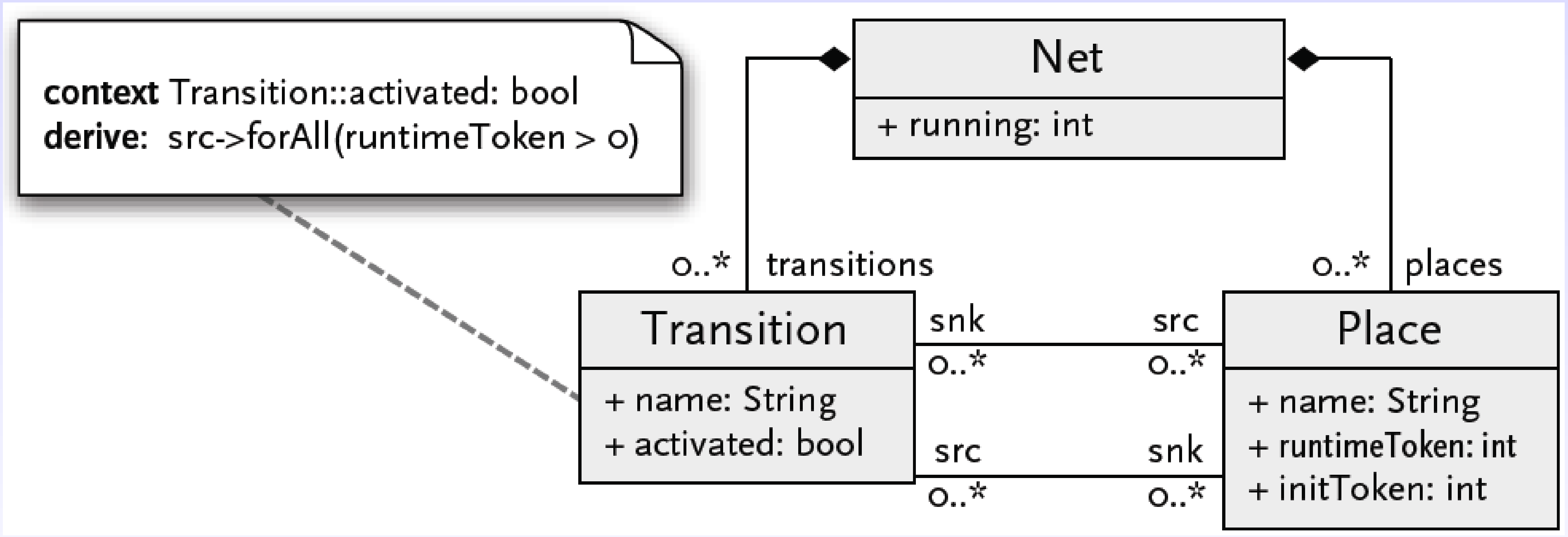
, ...}$\in \Gamma$

- Configurations are defined by a metamodel

- Transition relation → can be defined with a model-to-model transformation

Configuration Metamodel

Petri Net example



Transition Transformation

Part of Petri Net Java semantics
(erroneous version)

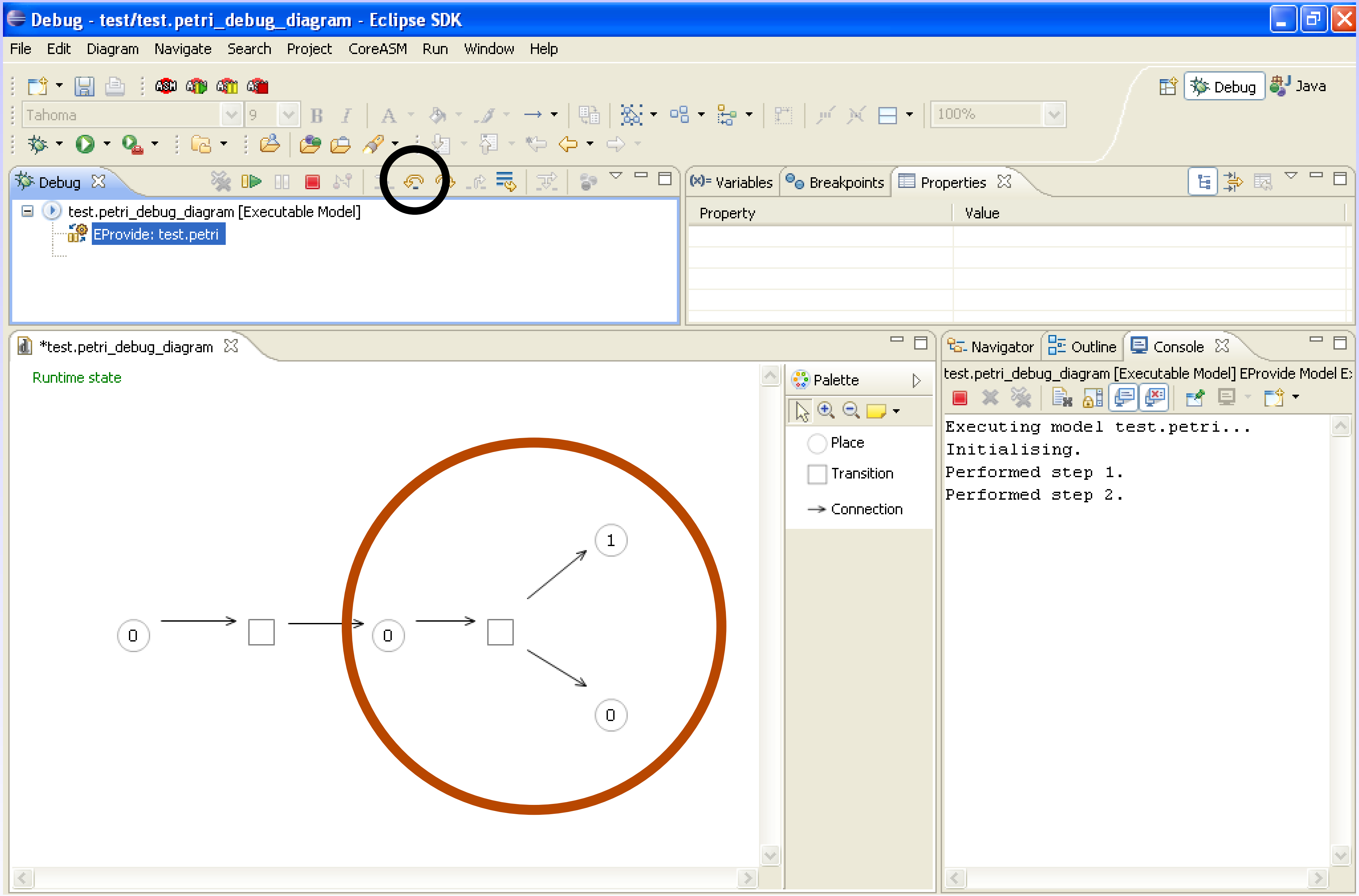
```
protected void run(Net net) {
   Transition t = getActivated(net);
   if (t != null) {
         consume(t.getSrc.get(0));

         produce(t.getSnk.get(0));
      }
   }
}
```

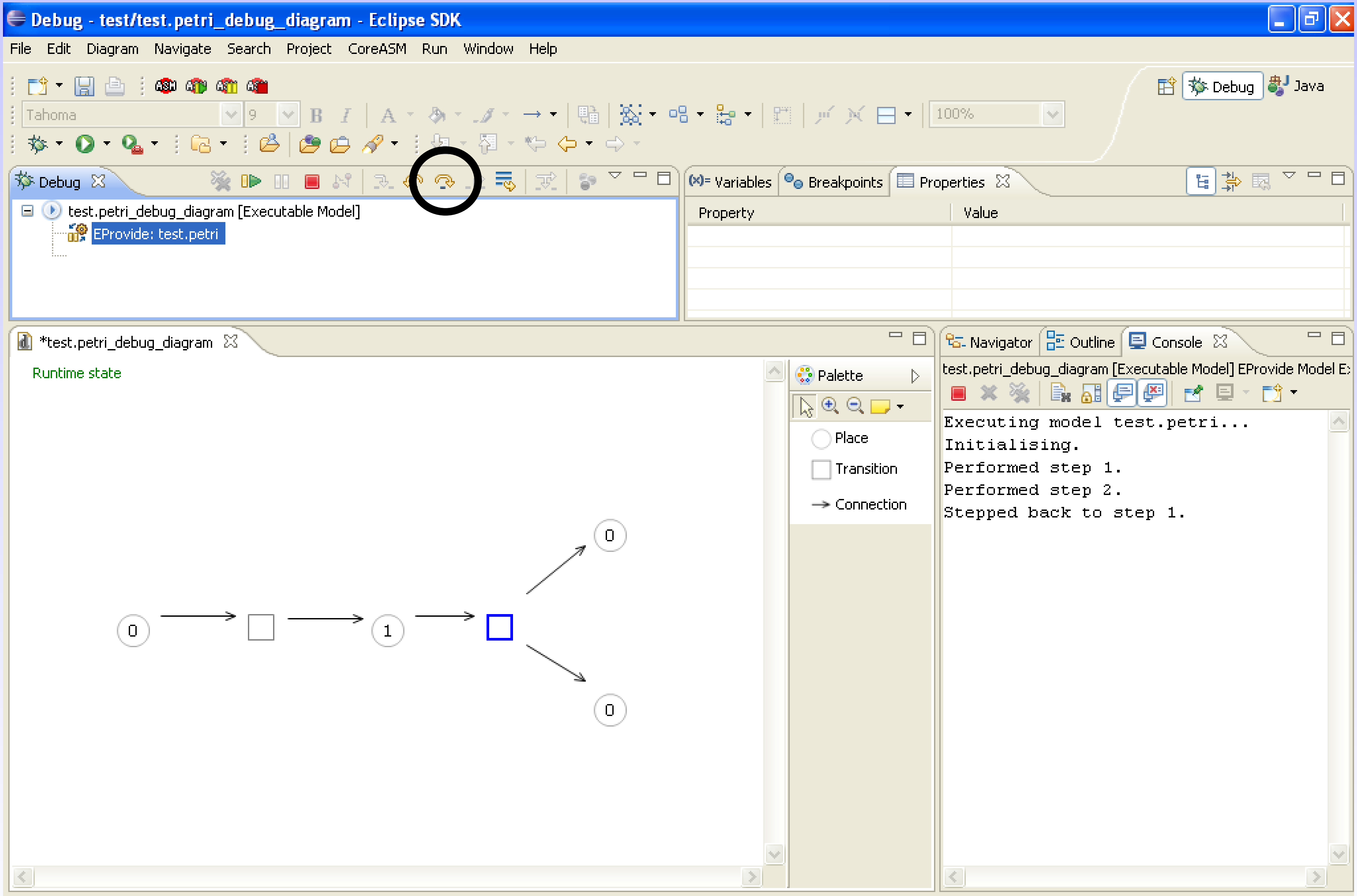
Transition Transformation

Part of Petri Net Java semantics
(corrected version)

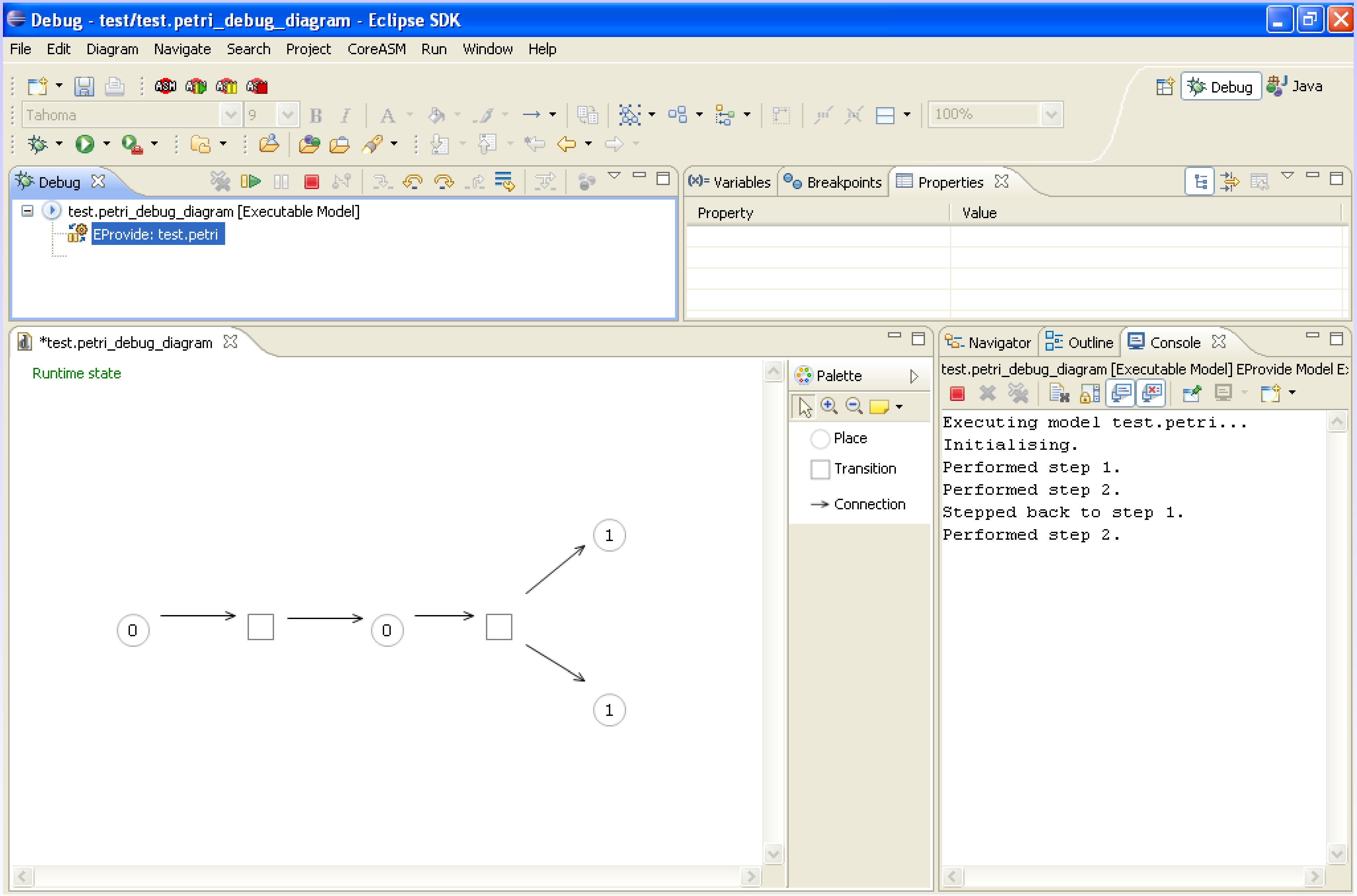
```
protected void run(Net net) {
   Transition t = getActivated(net);
   if (t != null) {
      for (Place p : t.getSrc()) {
         consume(p);
      }
      for (Place p : t.getSnk()) {
         produce(p);
      }
   }
}
```


Debug - test/test.petri_debug_diagram - Eclipse SDK
File Edit Diagram Navigate Search Project CoreASM Run Window Help
Debug
Java
Tahoma
9
100%
Debug
test.petri_debug_diagram [Executable Model]
EProvide: test.petri
Variables
Breakpoints
Properties
Property
Value
*test.petri_debug_diagram
Runtime state
0
0
1
0
Palette
Place
Transition
Connection
Navigator
Outline
Console
test.petri_debug_diagram [Executable Model] EProvide Model E:

```
Executing model test.petri...
Initialising.
Performed step 1.
Performed step 2.
```


Debug - test/test.petri_debug_diagram - Eclipse SDK
File Edit Diagram Navigate Search Project CoreASM Run Window Help
Debug
Java
Debug
test.petri_debug_diagram [Executable Model]
EProvide: test.petri
Variables
Breakpoints
Properties
Property
Value
*test.petri_debug_diagram
Runtime state
Palette
Place
Transition
Connection
0
1
0
0
Navigator
Outline
Console
test.petri_debug_diagram [Executable Model] EProvide Model E:

```
Executing model test.petri...
Initialising.
Performed step 1.
Performed step 2.
Stepped back to step 1.
```


Debug - test/test.petri_debug_diagram - Eclipse SDK
File Edit Diagram Navigate Search Project CoreASM Run Window Help
Debug
Java
Debug
test.petri_debug_diagram [Executable Model]
EProvide: test.petri
Variables
Breakpoints
Properties
Property
Value
*test.petri_debug_diagram
Runtime state
0
0
1
1
Palette
Place
Transition
Connection
Navigator
Outline
Console
test.petri_debug_diagram [Executable Model] EProvide Model E:

```
Executing model test.petri...
Initialising.
Performed step 1.
Performed step 2.
Stepped back to step 1.
Performed step 2.
```

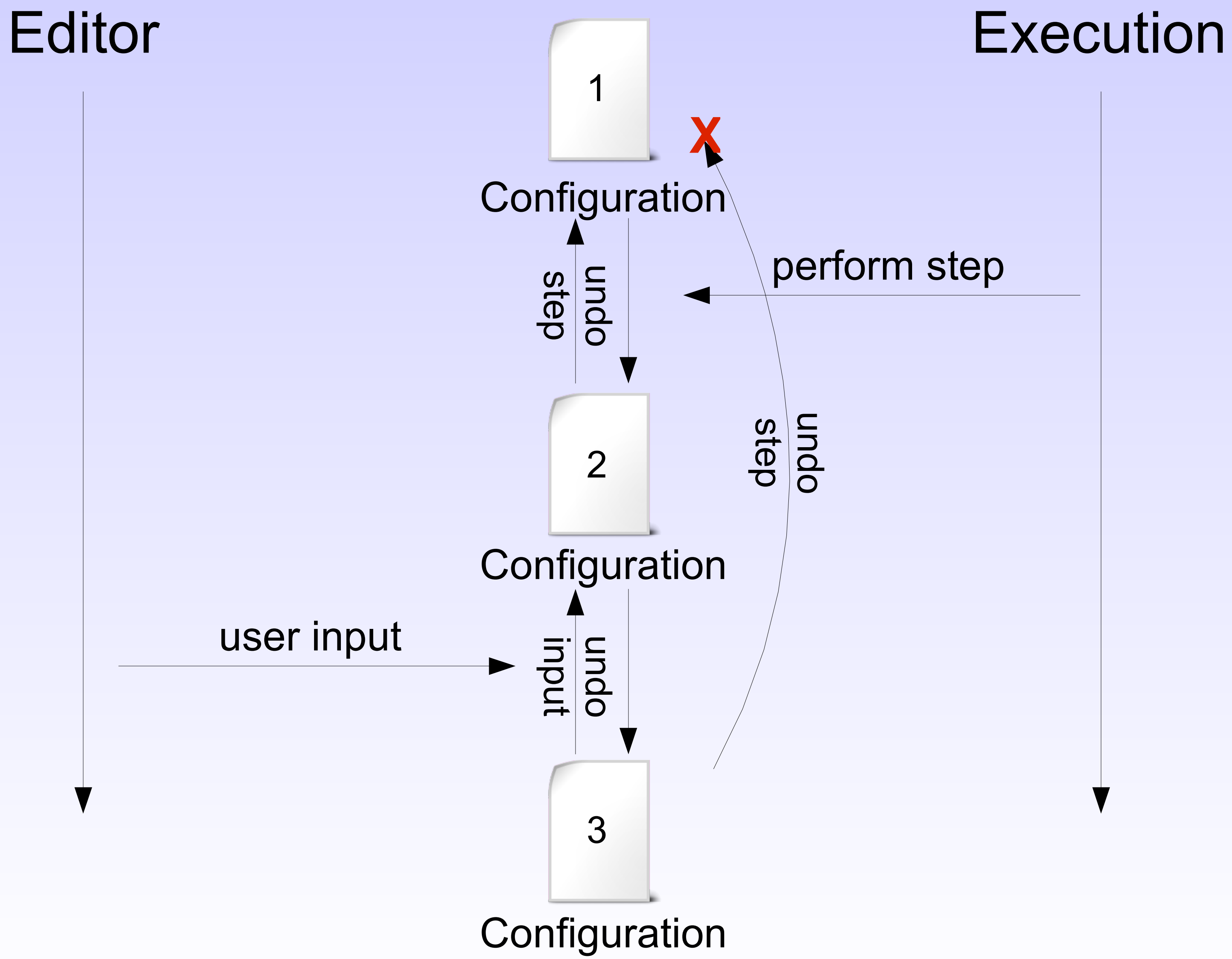
Undoing Operational Steps

- Undo: reverse changes
- Observer for model changes

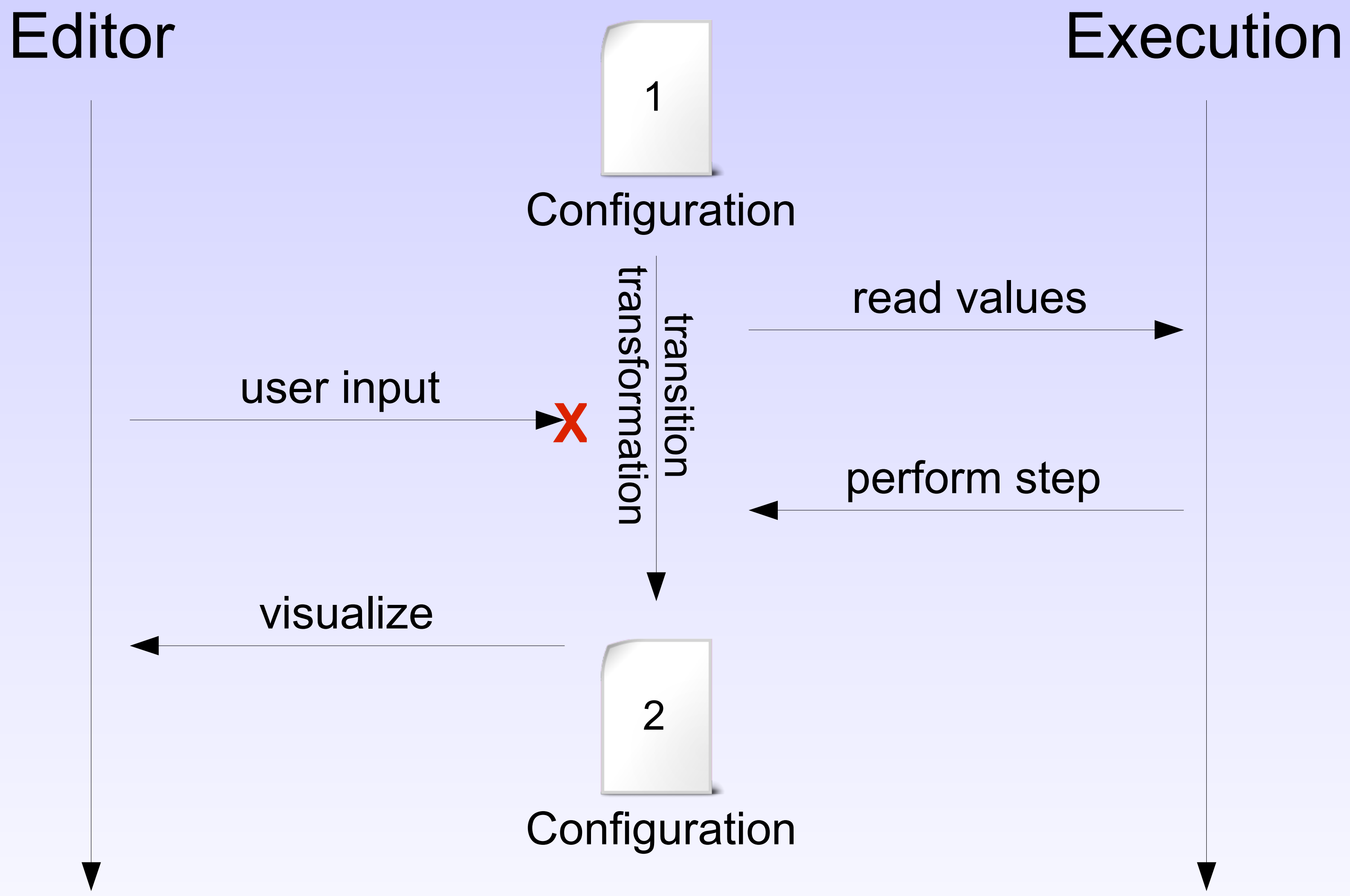
- Execution step: single unit of work
- Composition of elementary changes

- Change history:
- Shared command stack for editor and execution

Command Order



Synchronization



Open Issues

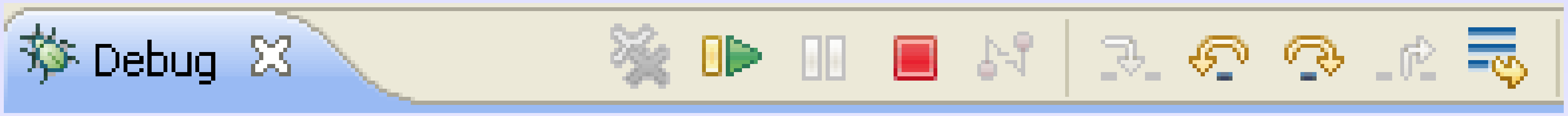
- Breakpoints between execution steps
- Declarative breakpoint description?

- Users can produce invalid configurations
- How to describe and implement constraints?

- Changing operational semantics can affect previous configurations
- How to step back to last state that is consistent with changed semantics?

Conclusion

- New debug feature for DSML prototyping
- Adapting undo of editors for stepwise model execution



step back

- Implementation experience: many building blocks available in EMF

Undoing Operational Steps of
Domain-Specific Modeling Languages

Discussion

Tim Hartmann, Daniel A. Sadilek
Humboldt-Universität zu Berlin