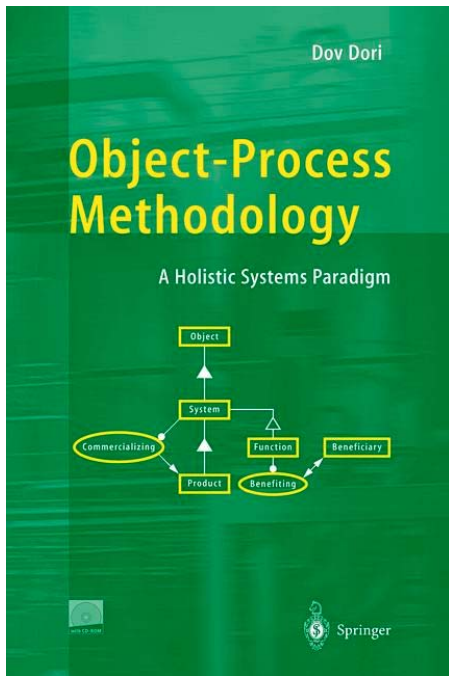
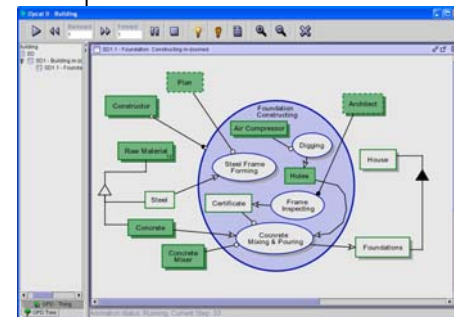
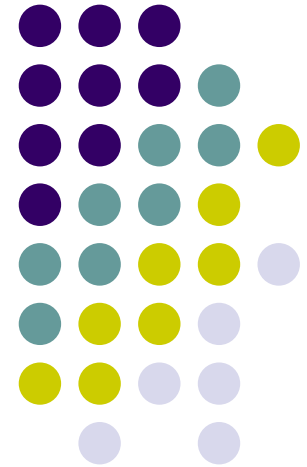


# Object-Process Methodology and Its Application to the Visual Semantic Web

Pre-Conference Tutorial PT1  
ER-2003, Chicago  
October 12, 2003



Dov Dori  
Technion, Israel; MIT, USA  
[www.ObjectProcess.org](http://www.ObjectProcess.org)

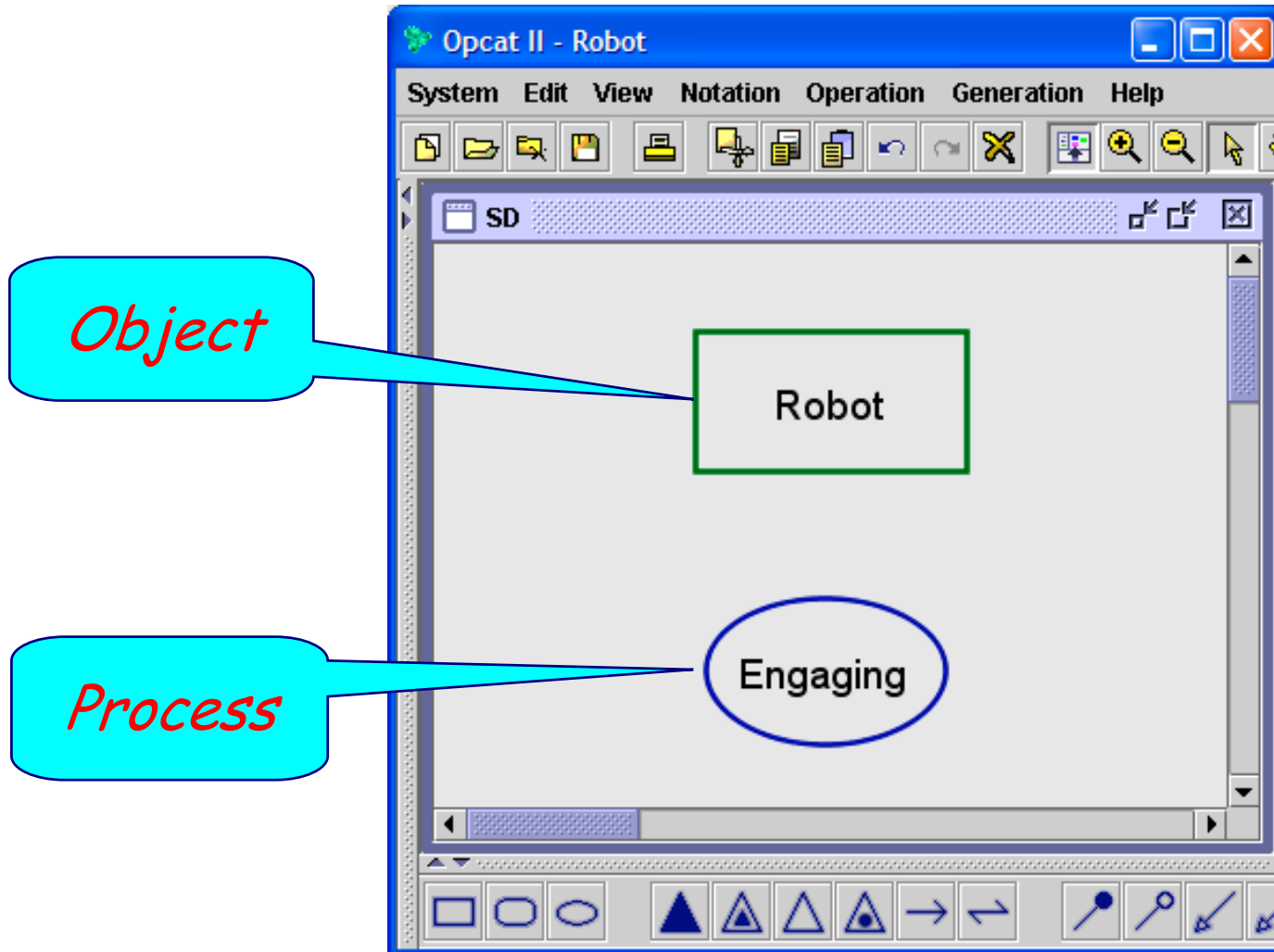


# What is Object-Process Methodology (OPM)?

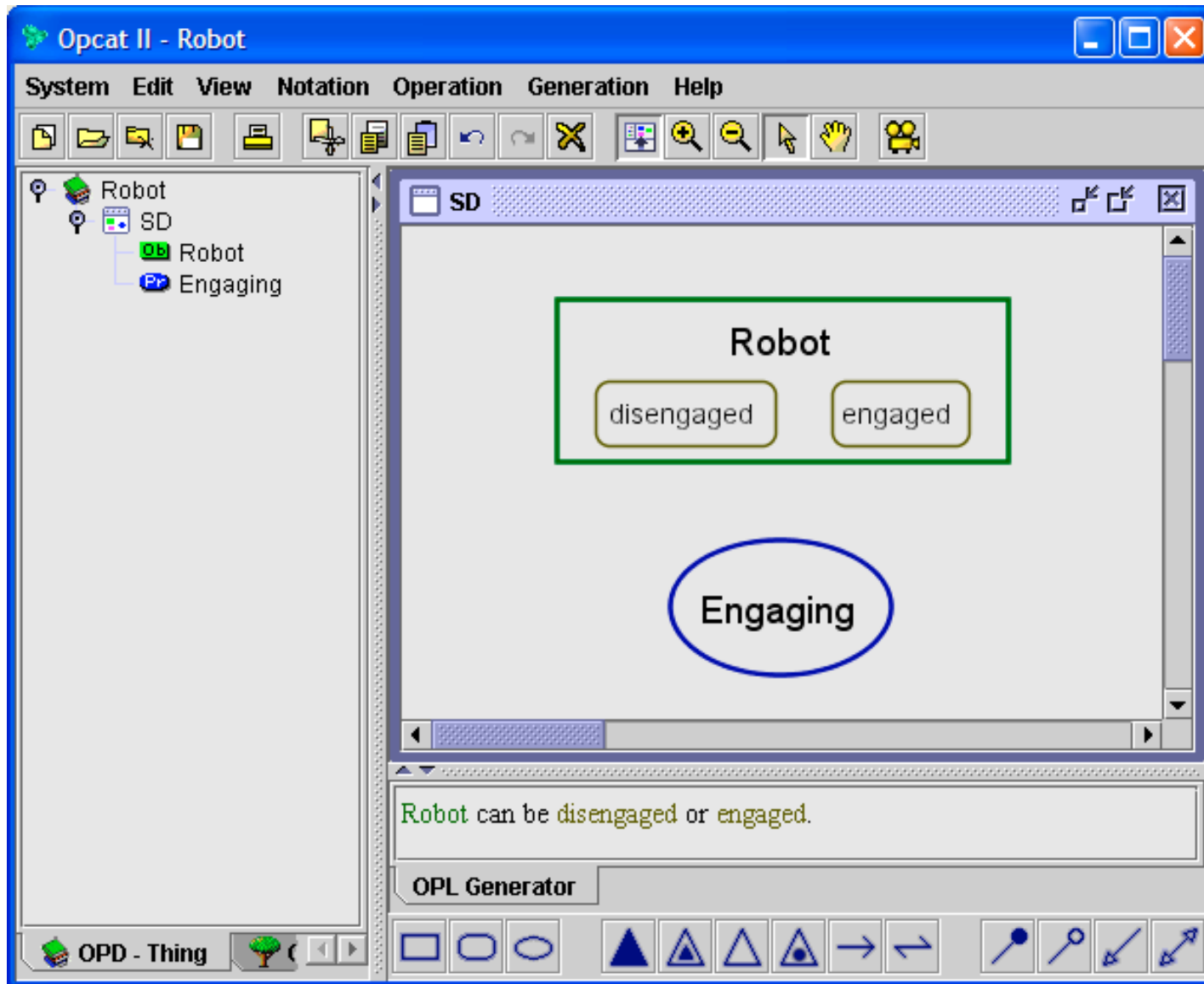
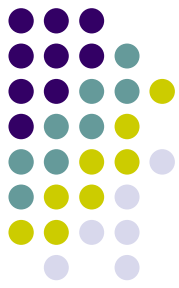


- A comprehensive patented systems modeling, engineering, and lifecycle support paradigm
- Two major features:
  - Unification of function, structure and behavior in a **single** model
  - Bi-modal expression of the model via intuitive yet formal **graphics** and equivalent **natural language**

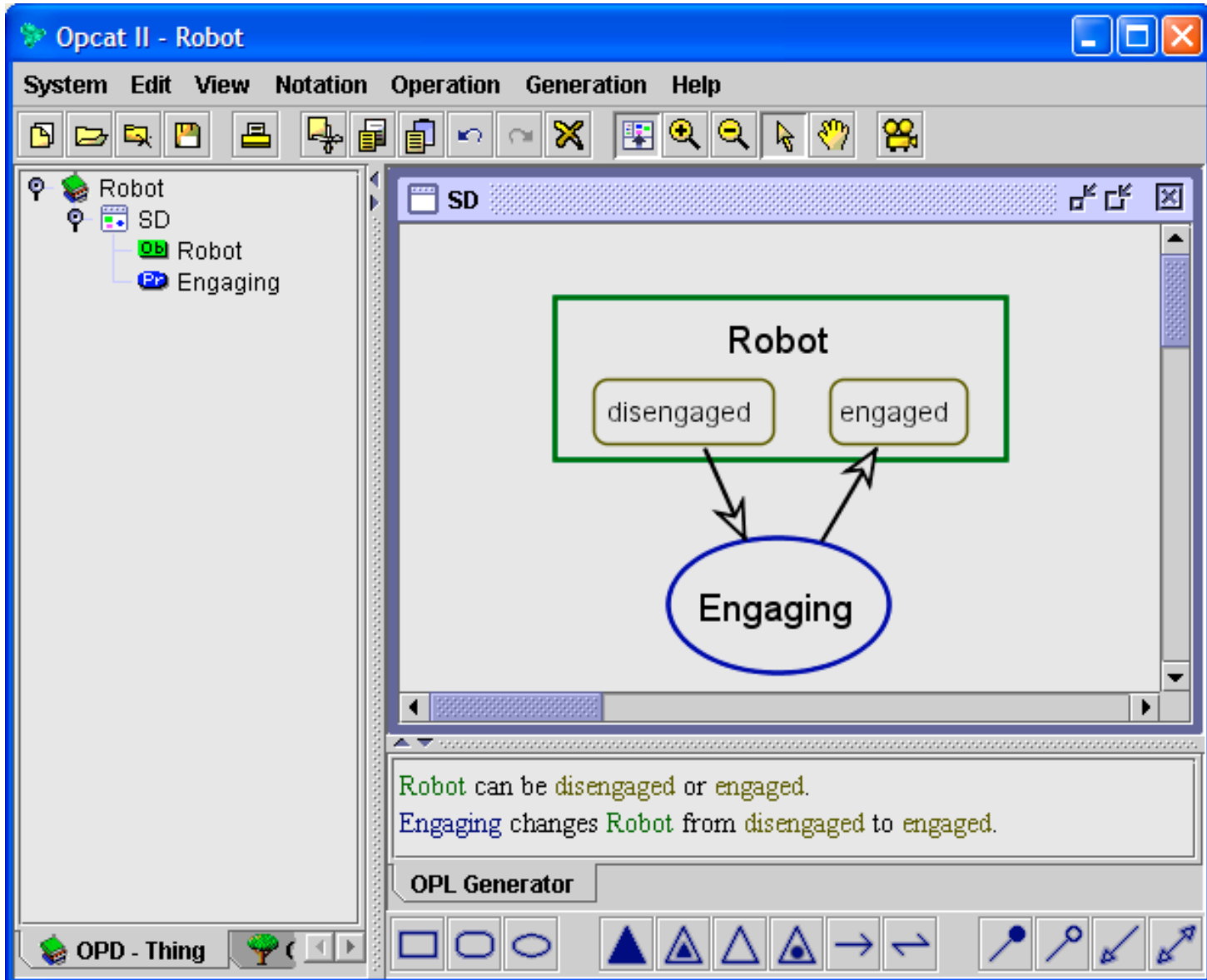
# OPM's Building Blocks are Things: Objects and Processes



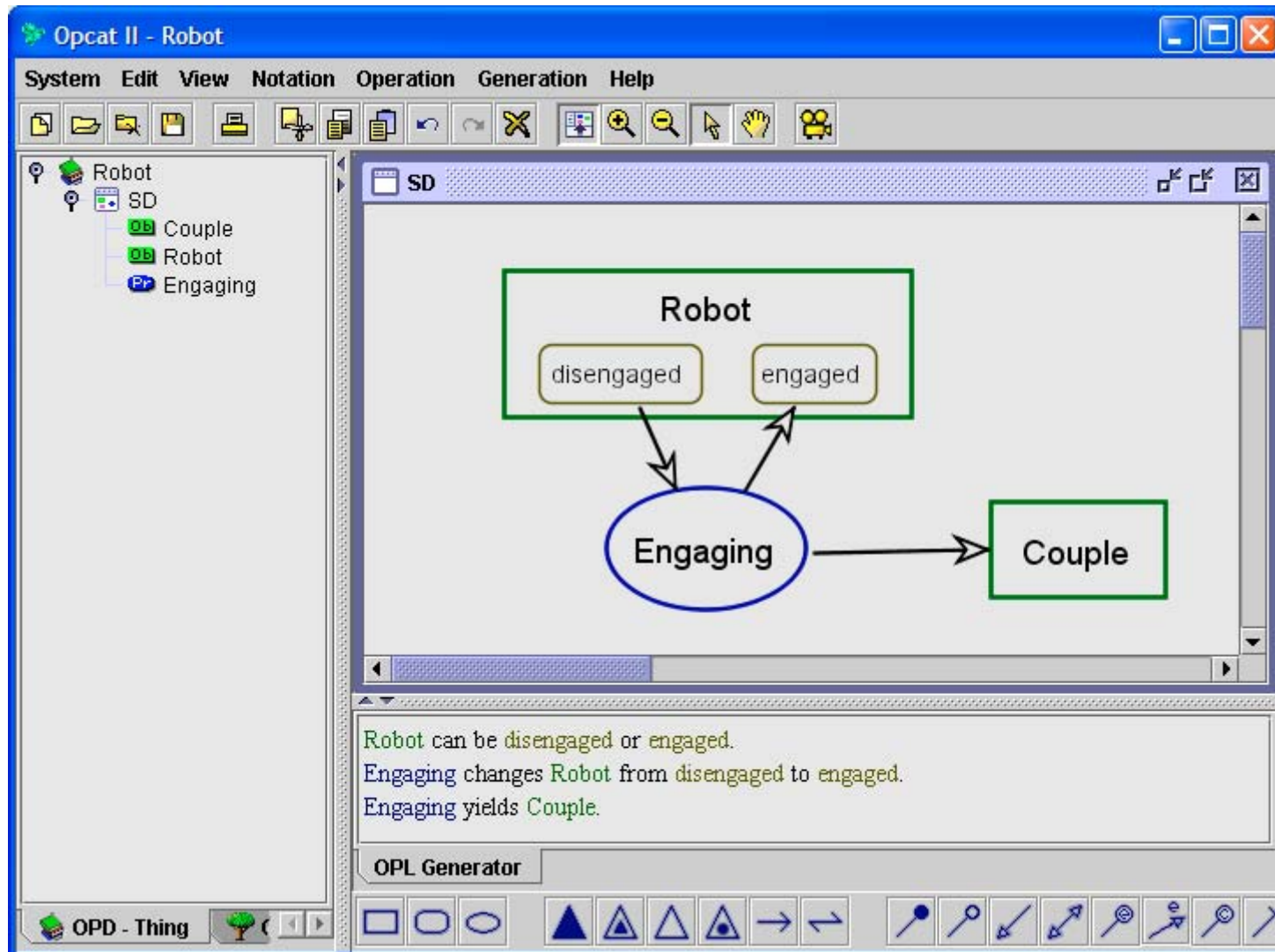
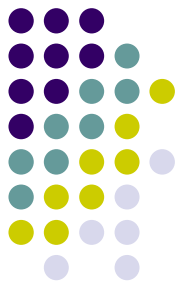
# States are situations of objects



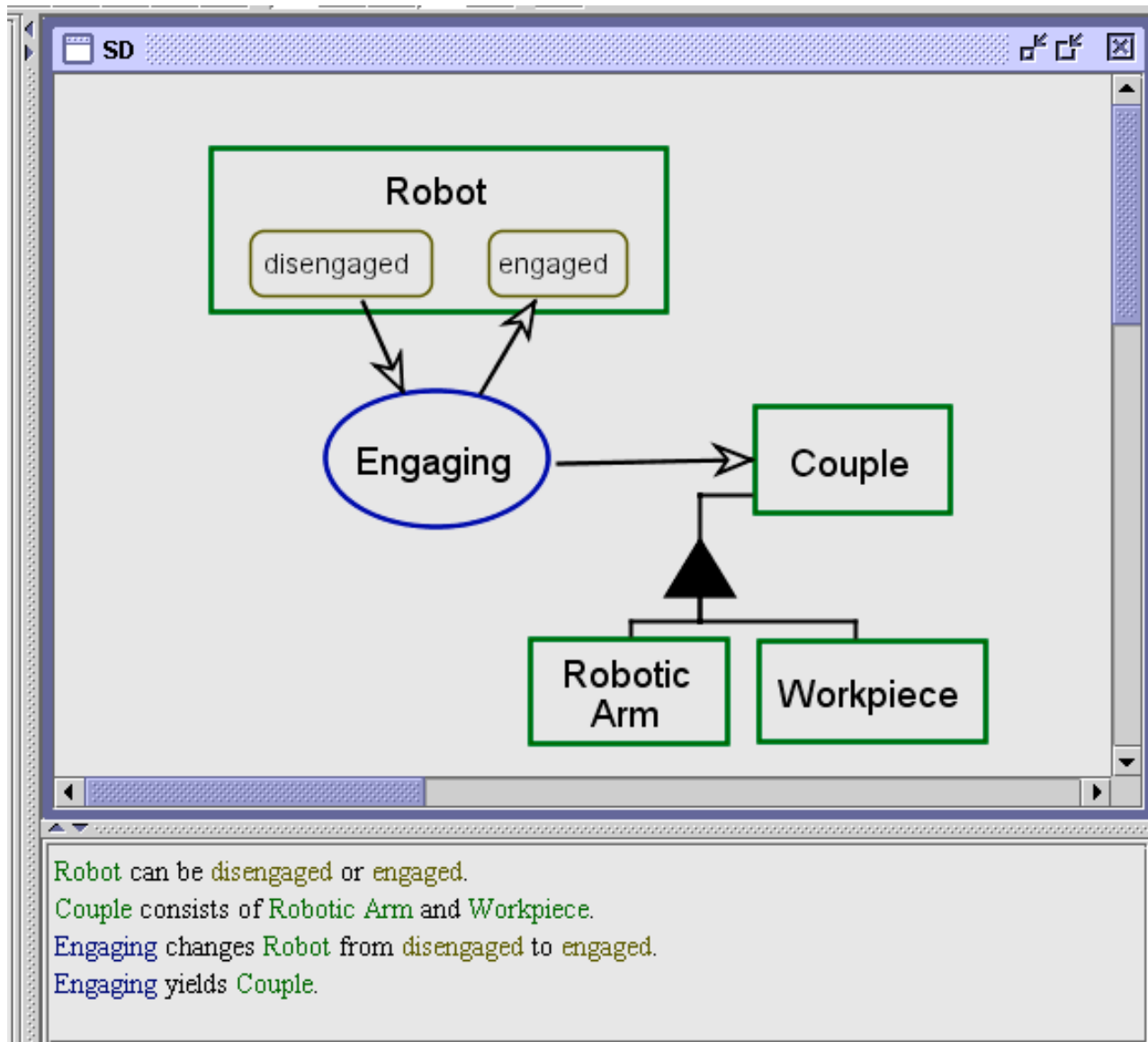
# A process changes an object state



# A process generates a new object



# Combining Behavior with Structure



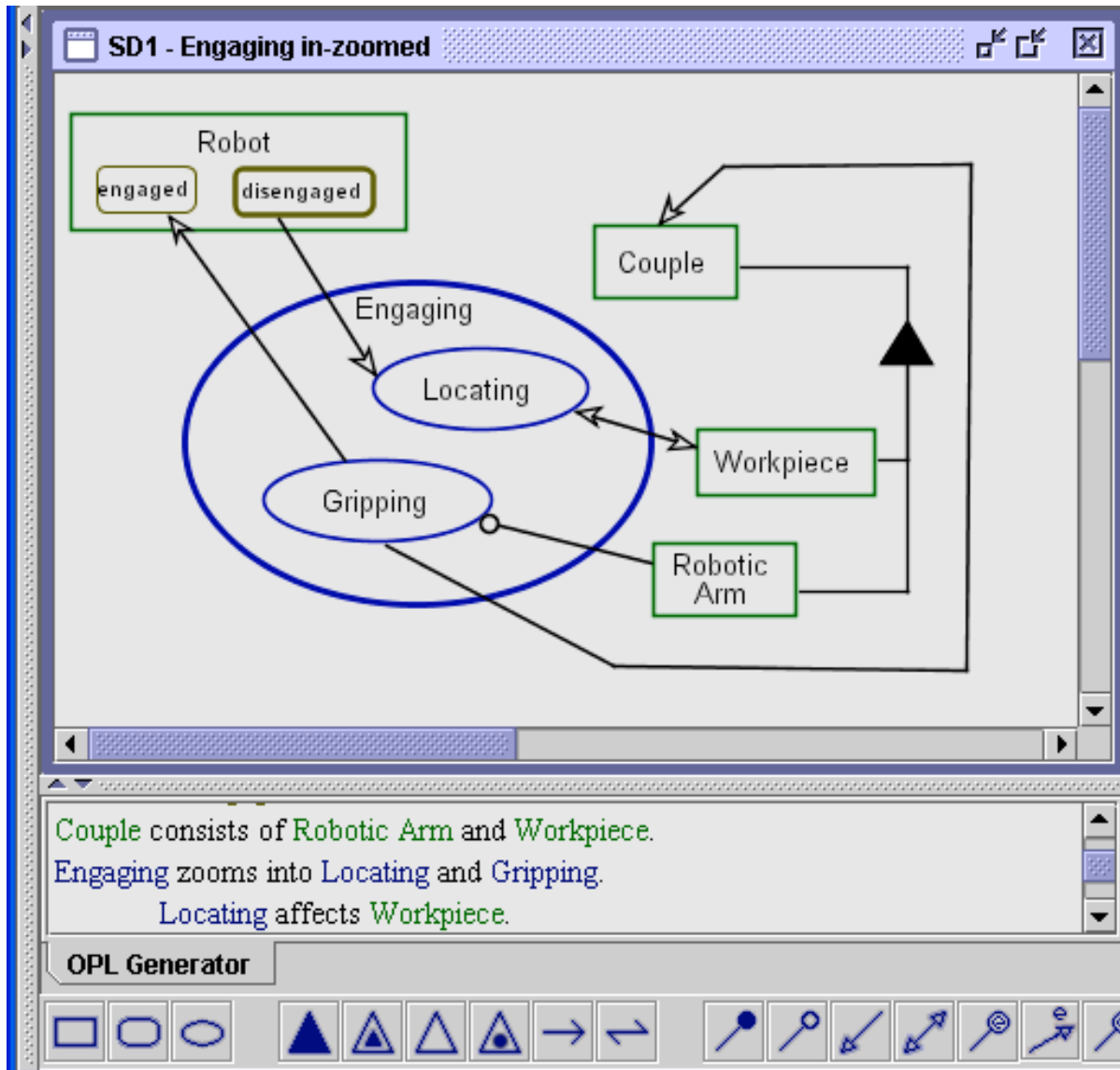
# Why Object-Process Methodology?



- As the inherent complexity and interdisciplinary nature of systems increases, the need for a universal modeling, engineering, and lifecycle support approach becomes ever more essential.
- The unnecessary complexity and software orientation of UML – the current standard language – calls for a simpler, formal, generic paradigm for systems development.



# Complexity Management



# Animated Simulation: Starting



Opcat II - Robot

Backward 1 Forward 1

Robot

- SD
- SD1 - Engaging

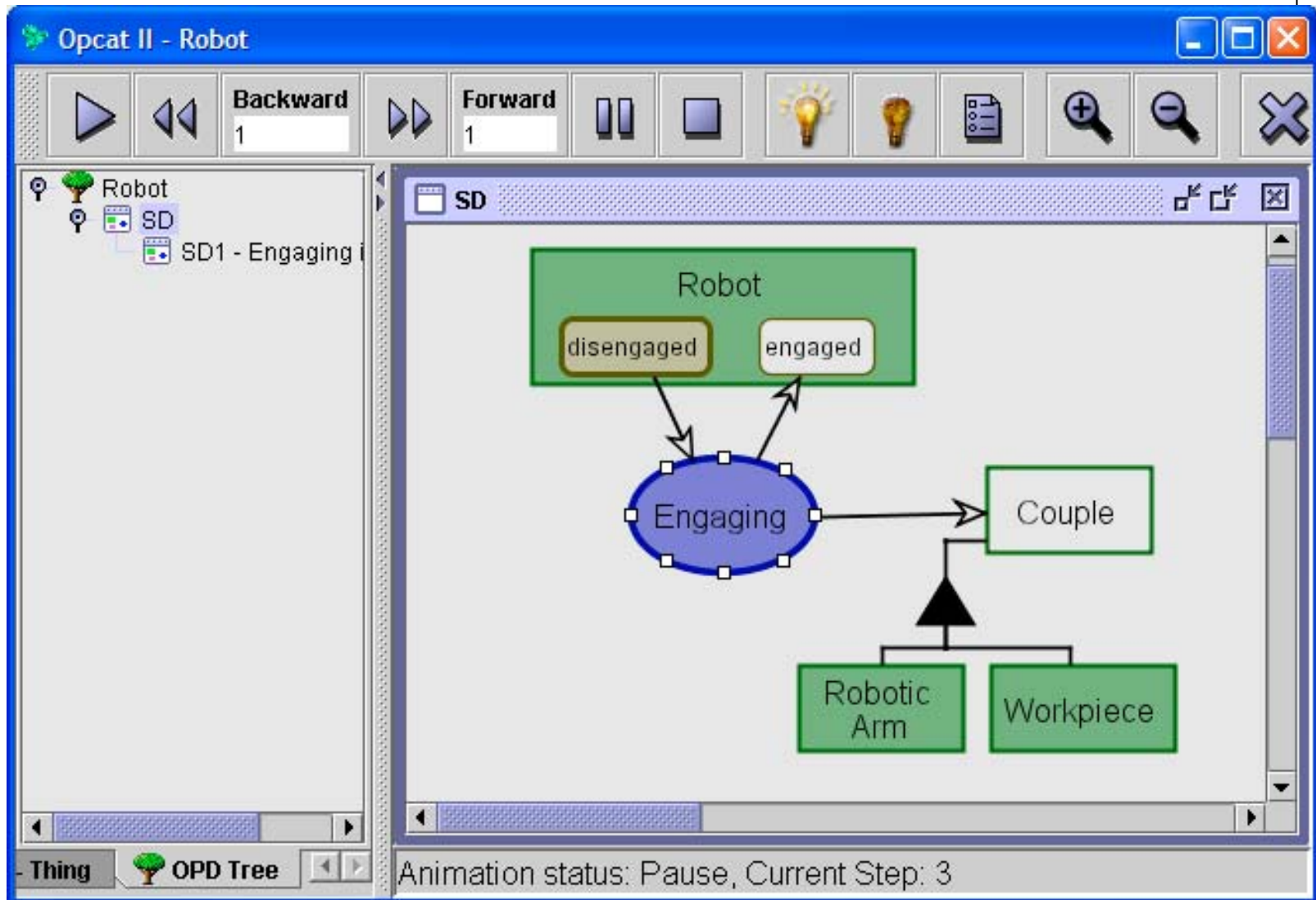
SD

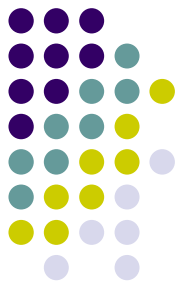
```
graph TD; Robot[Robot] --- disengaged[disengaged]; Robot --- engaged[engaged]; Engaging((Engaging)) --> disengaged; Engaging --> engaged; Engaging --> Couple[Couple]; Couple --> RA[Robotic Arm]; Couple --> WP[Workpiece];
```

Thing OPD Tree

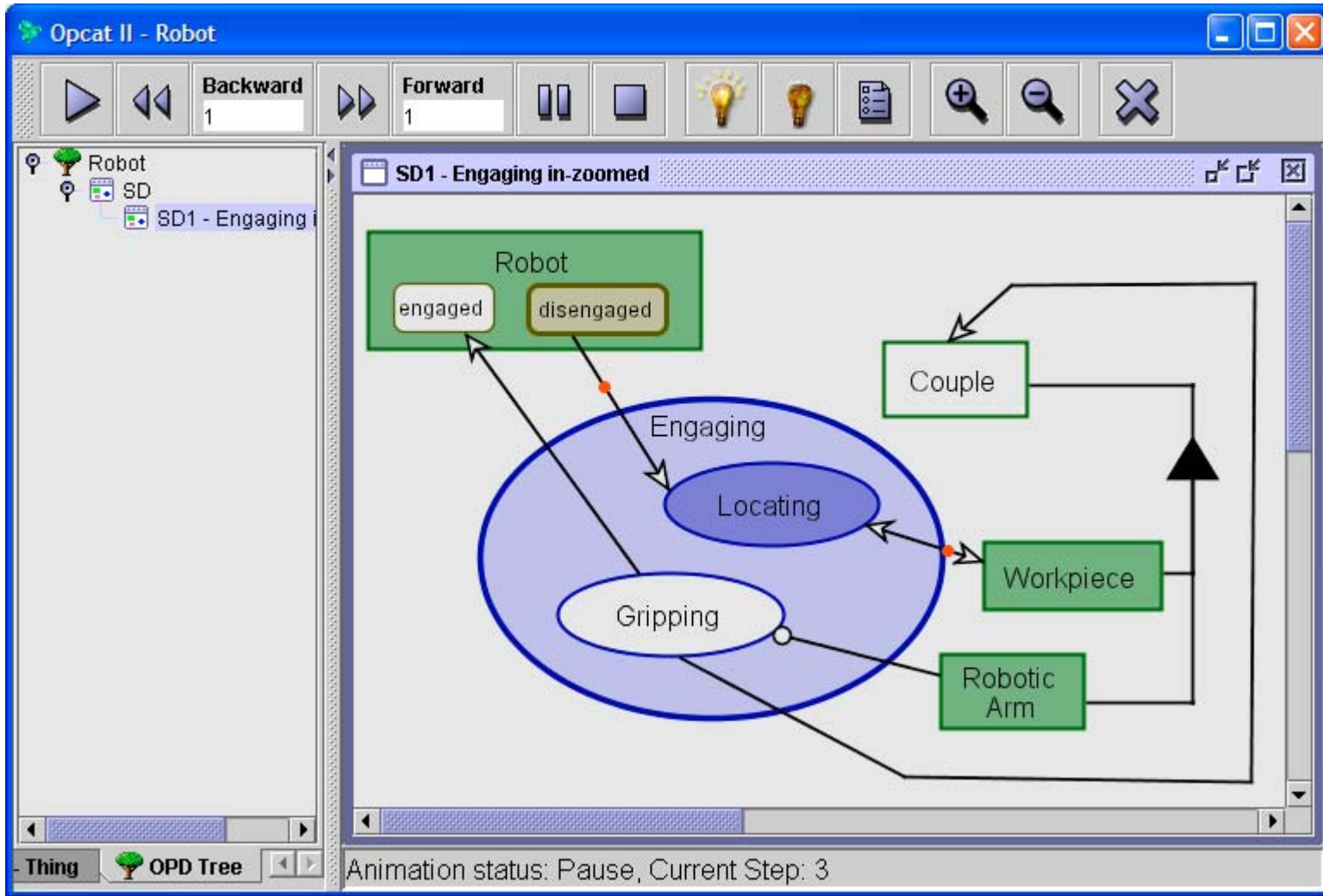
Animation status: Pause, Current Step: 2

# Animated Simulation: Engaging activated

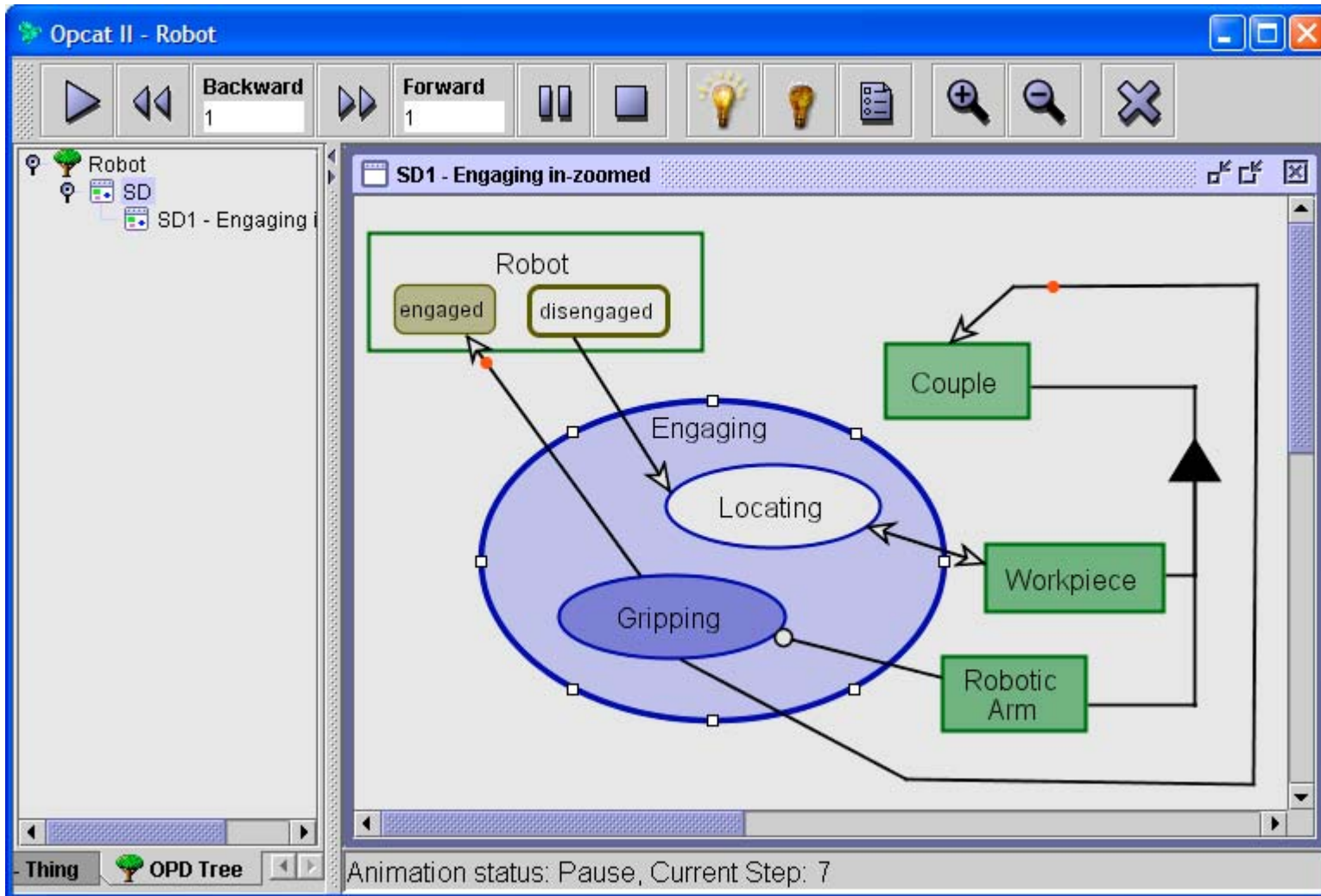




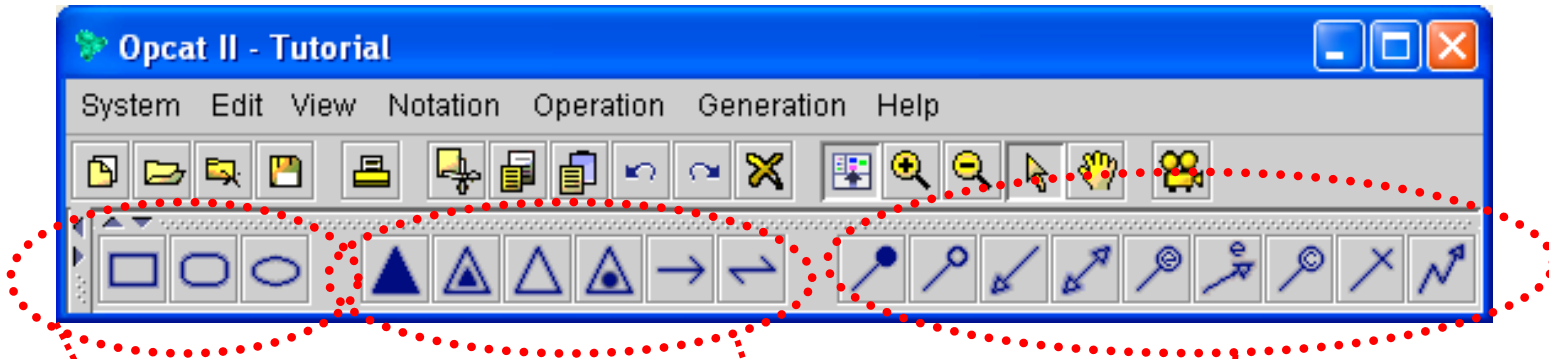
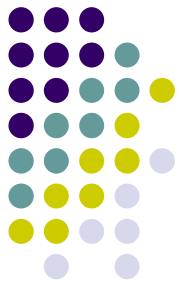
# Animated Simulation: Locating occurs, Workpiece affected



# Animated Simulation: Gripping occurs, Couple generated

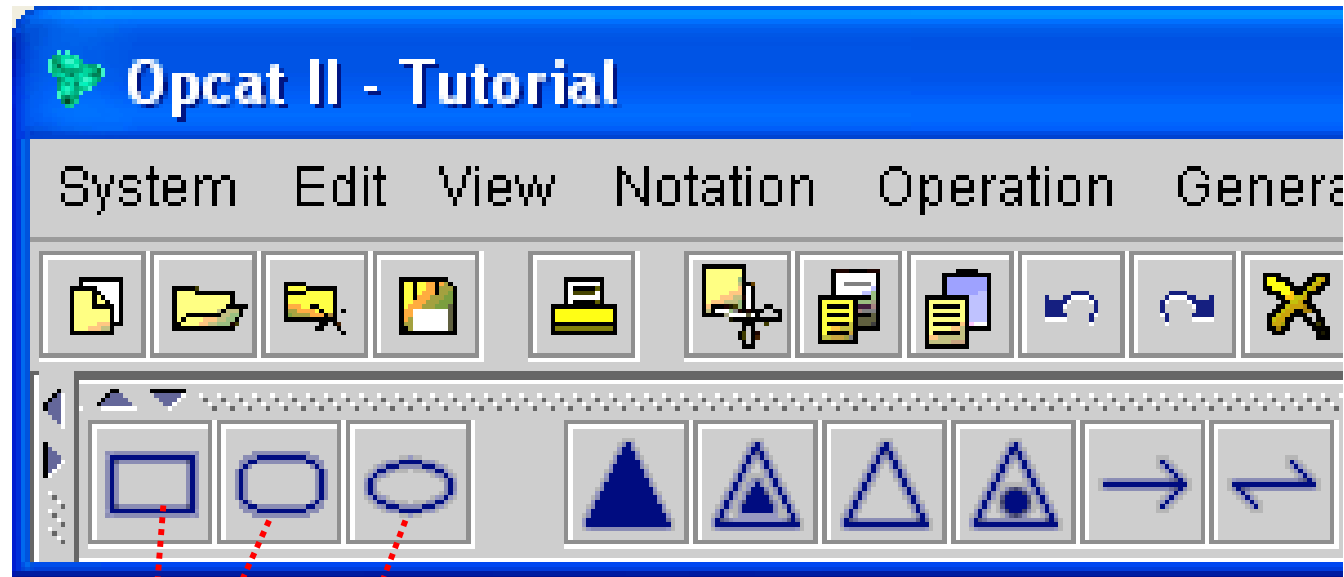


# OPM Elements: Entities and Links



- Entities:
  - Object
  - State
  - Process
- Links:
  - Structural link
  - Procedural link

# OPM Entities

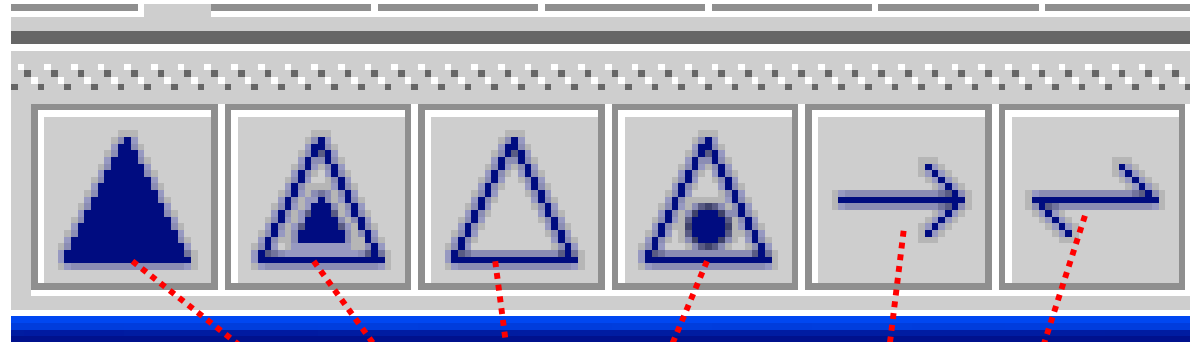


- **Object:** A thing that exists for some time
- **State:** A situation at which an object can be
- **Process:** A thing that transforms an object

# OPM Structural Links



Links denoting persistent relations between objects



Fundamental:

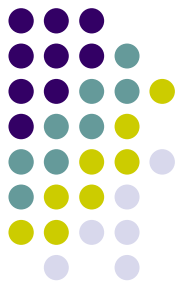
- Aggregation-participation
- Exhibition-characterization
- Generalization-specialization
- Classification-instantiation

General:

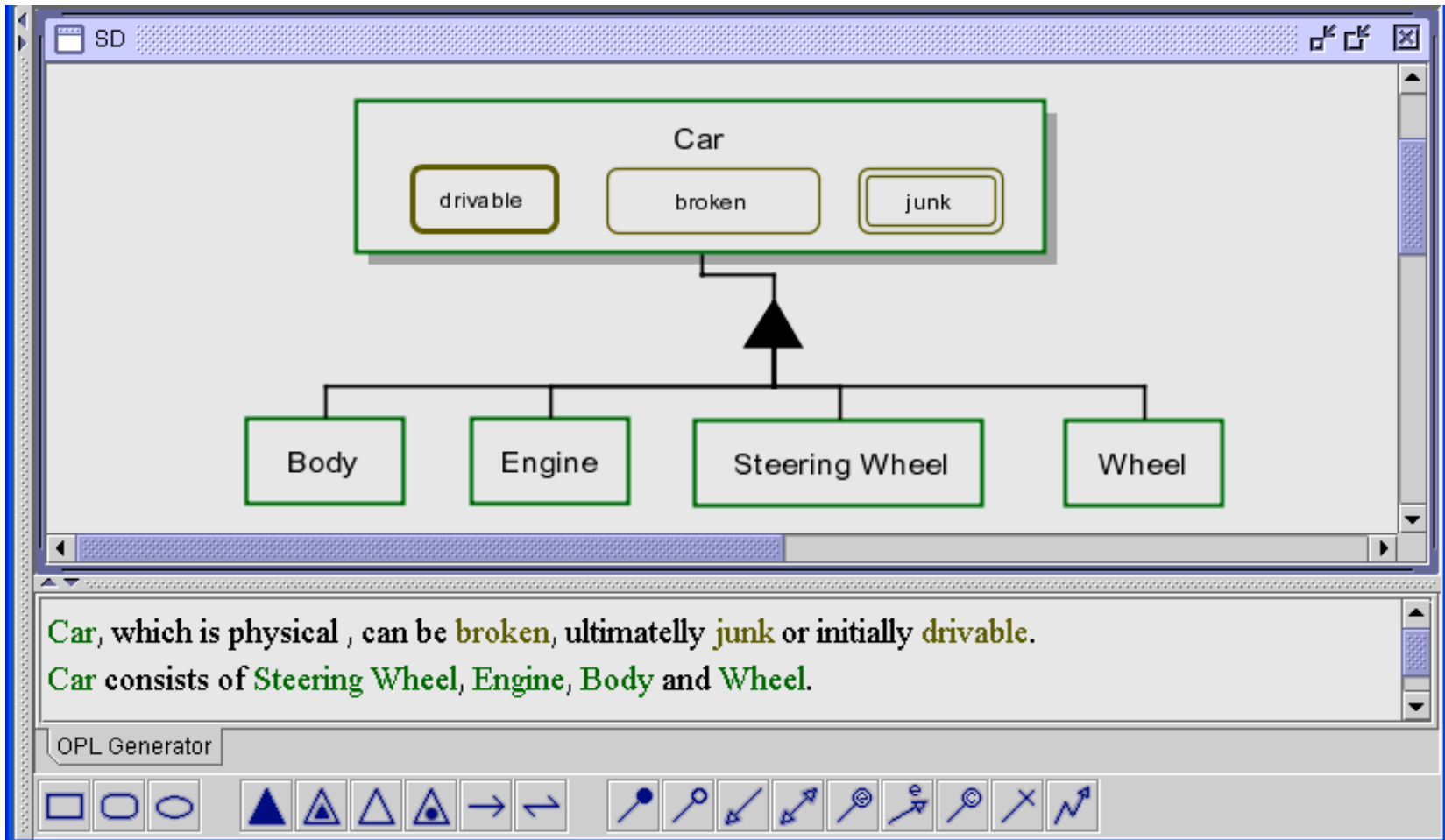
- Unidirectional tagged structural relation
- Bidirectional tagged structural relation



# Aggregation-participation



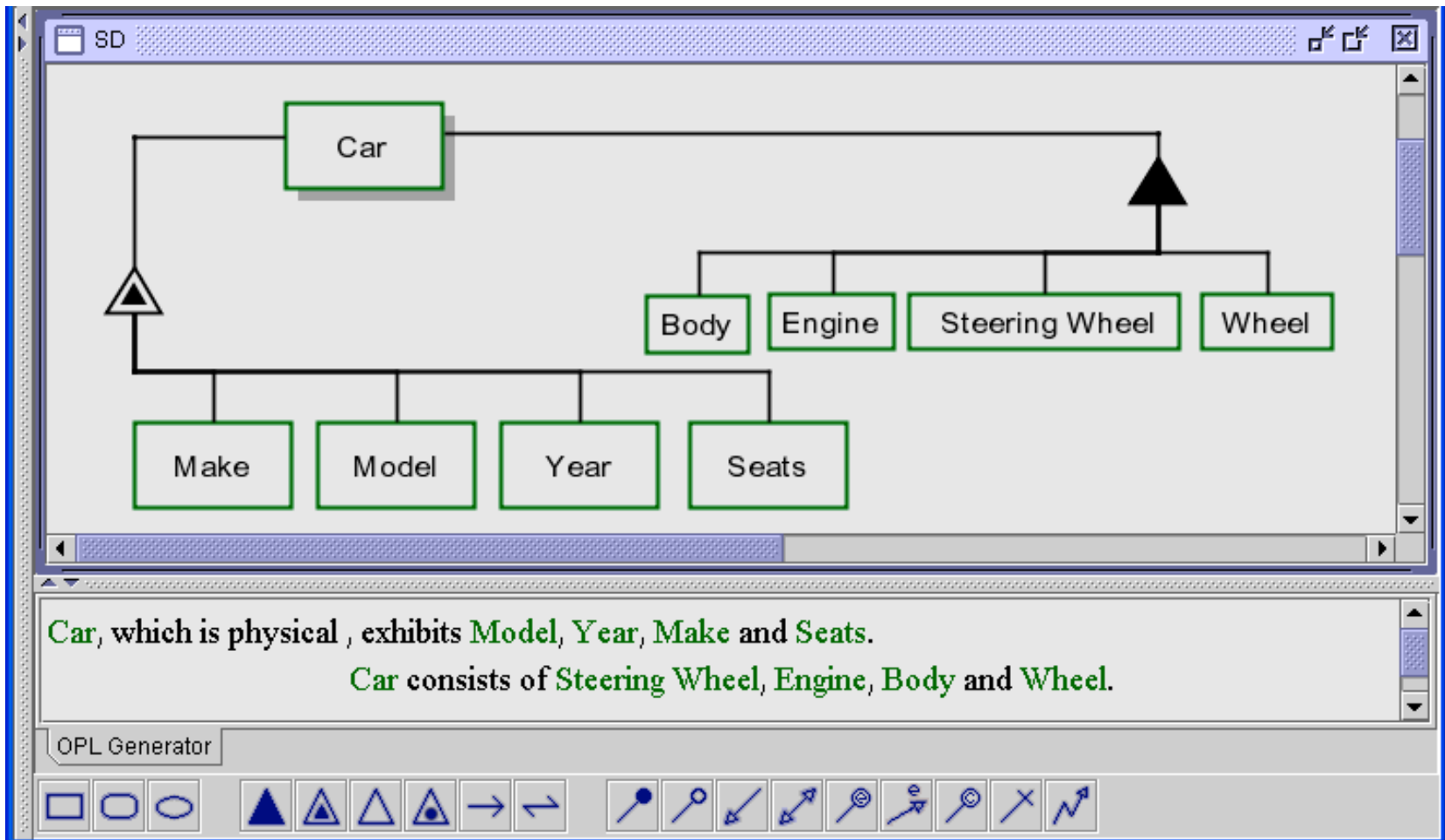
A structural relation between the whole and its parts



# Exhibition-characterization



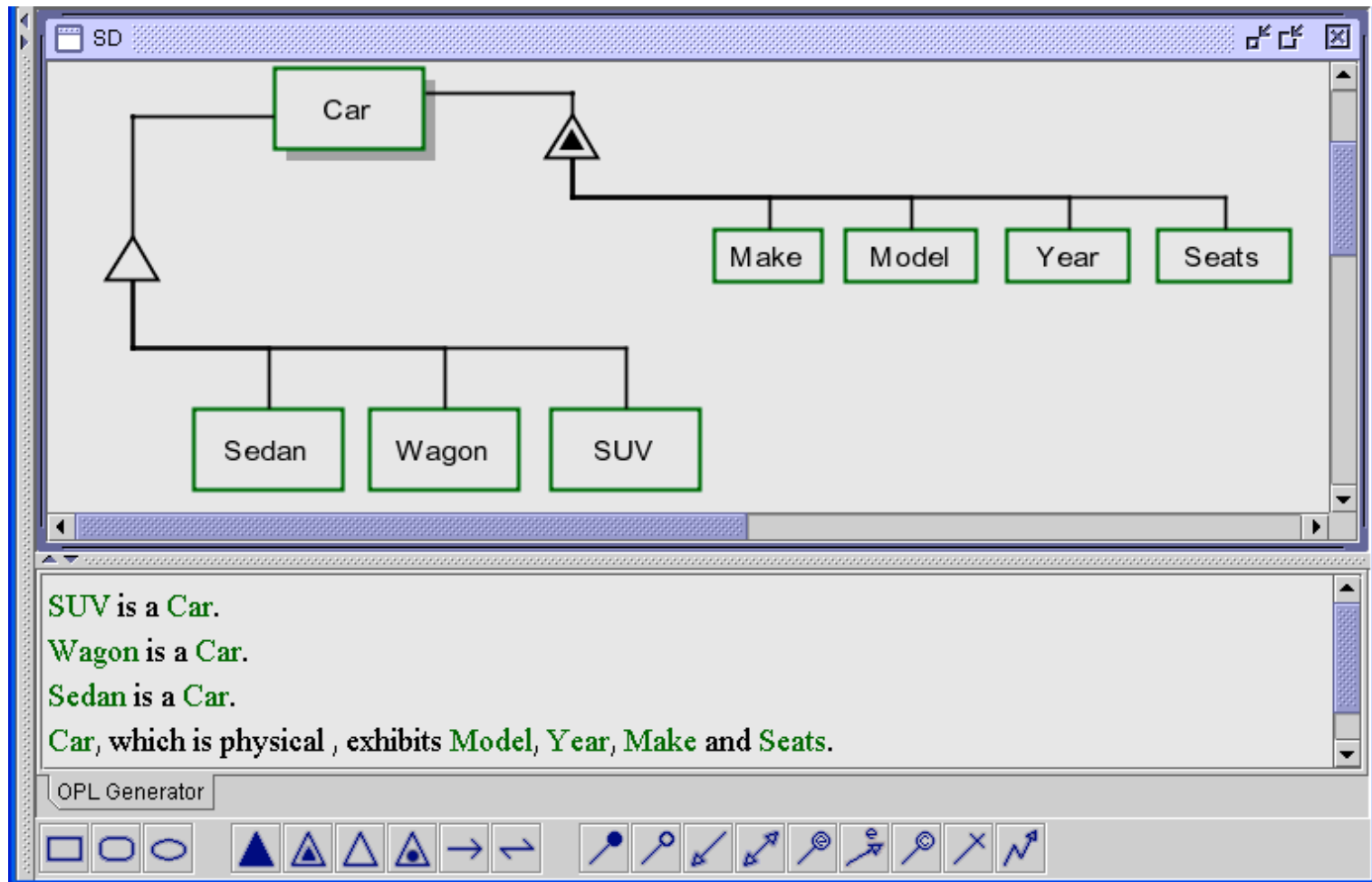
A structural relation between a thing and its features



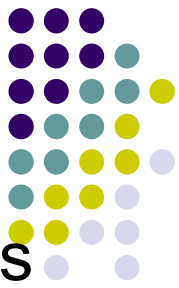
# Generalization-specialization



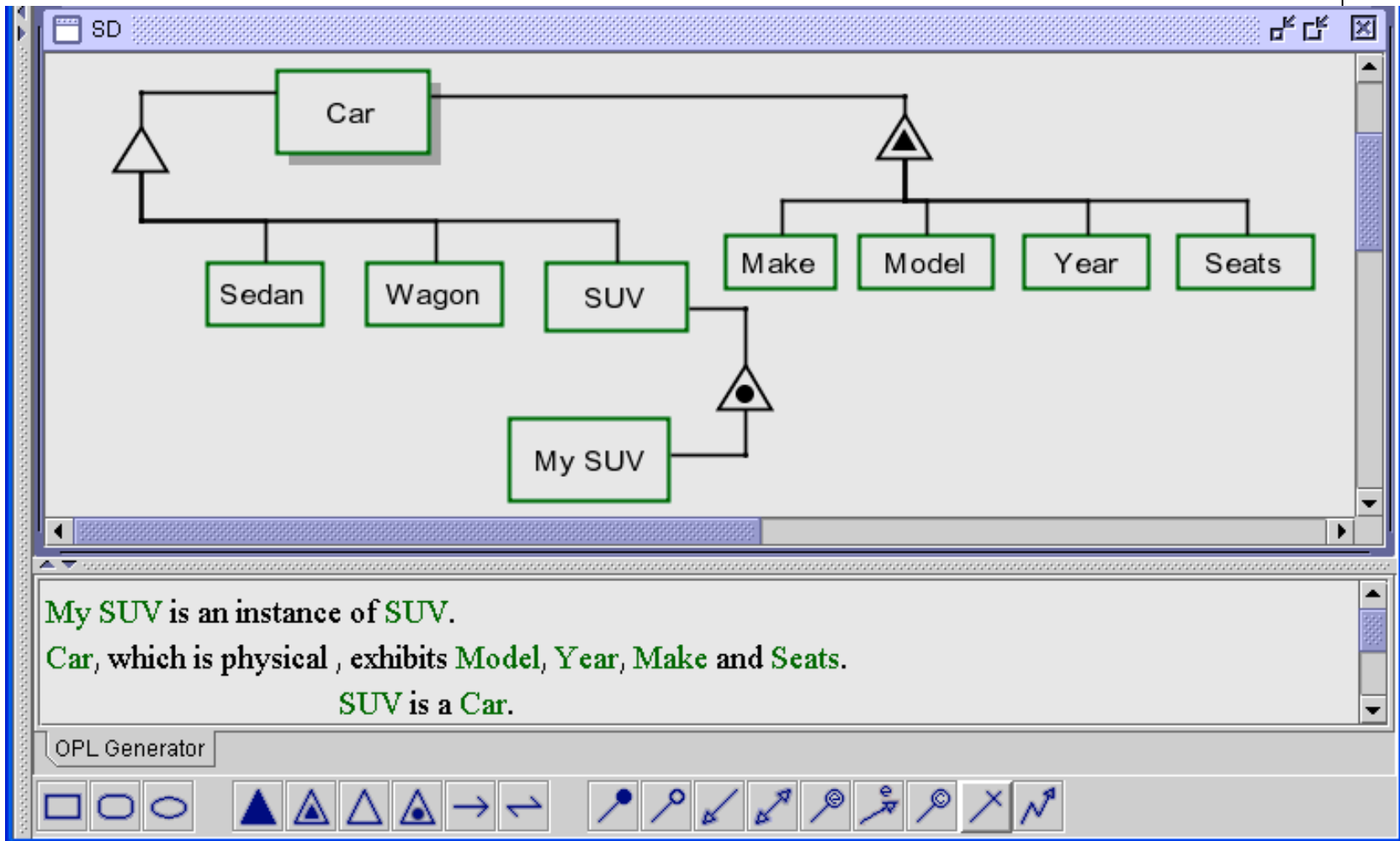
A structural relation between a thing and its specializations (known as the “is-a” relation)



# Classification-instantiation

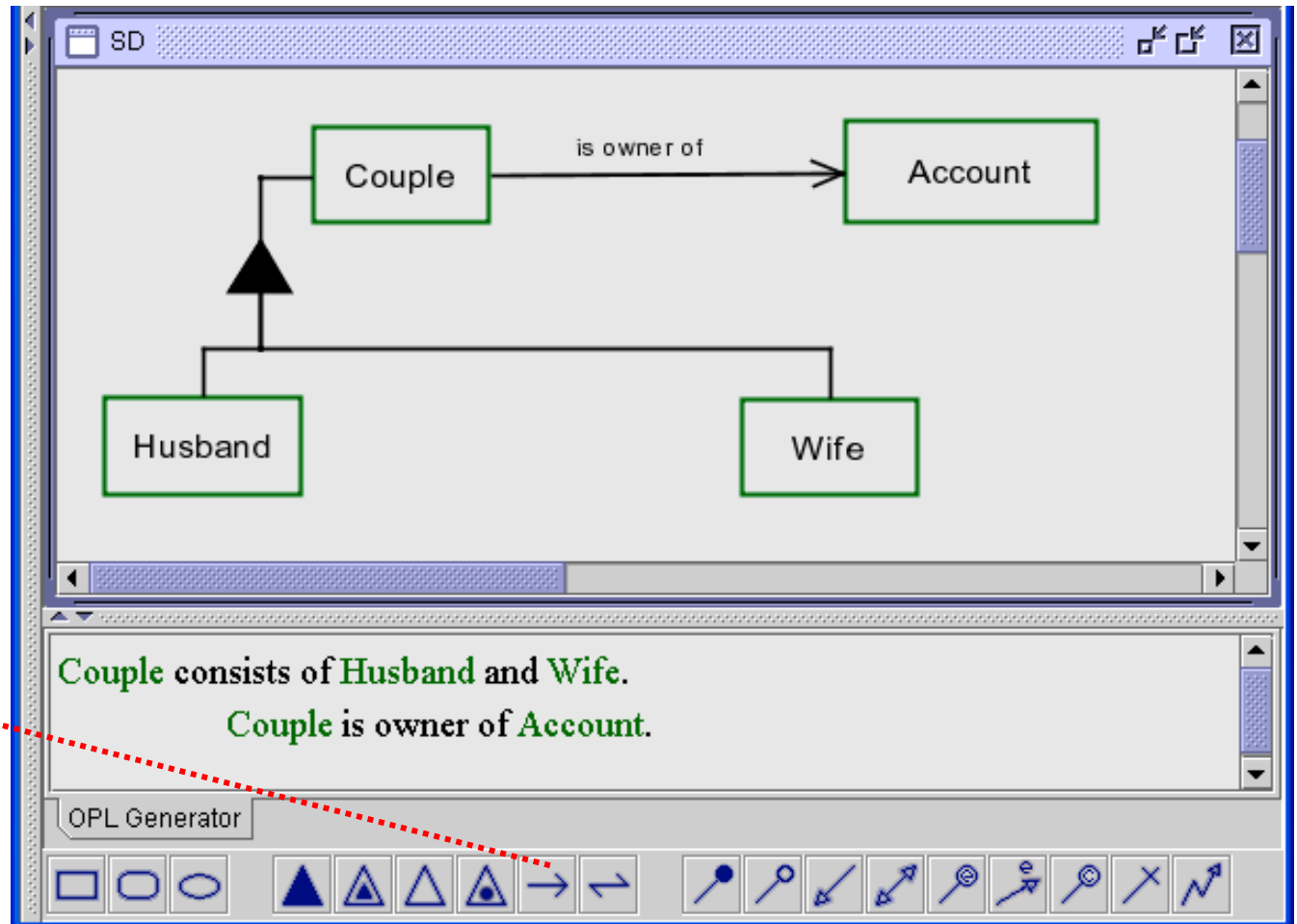
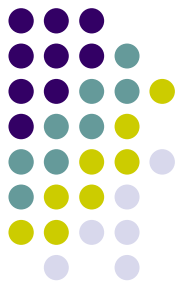


A structural relation between a thing and its instances



# General tagged structural link

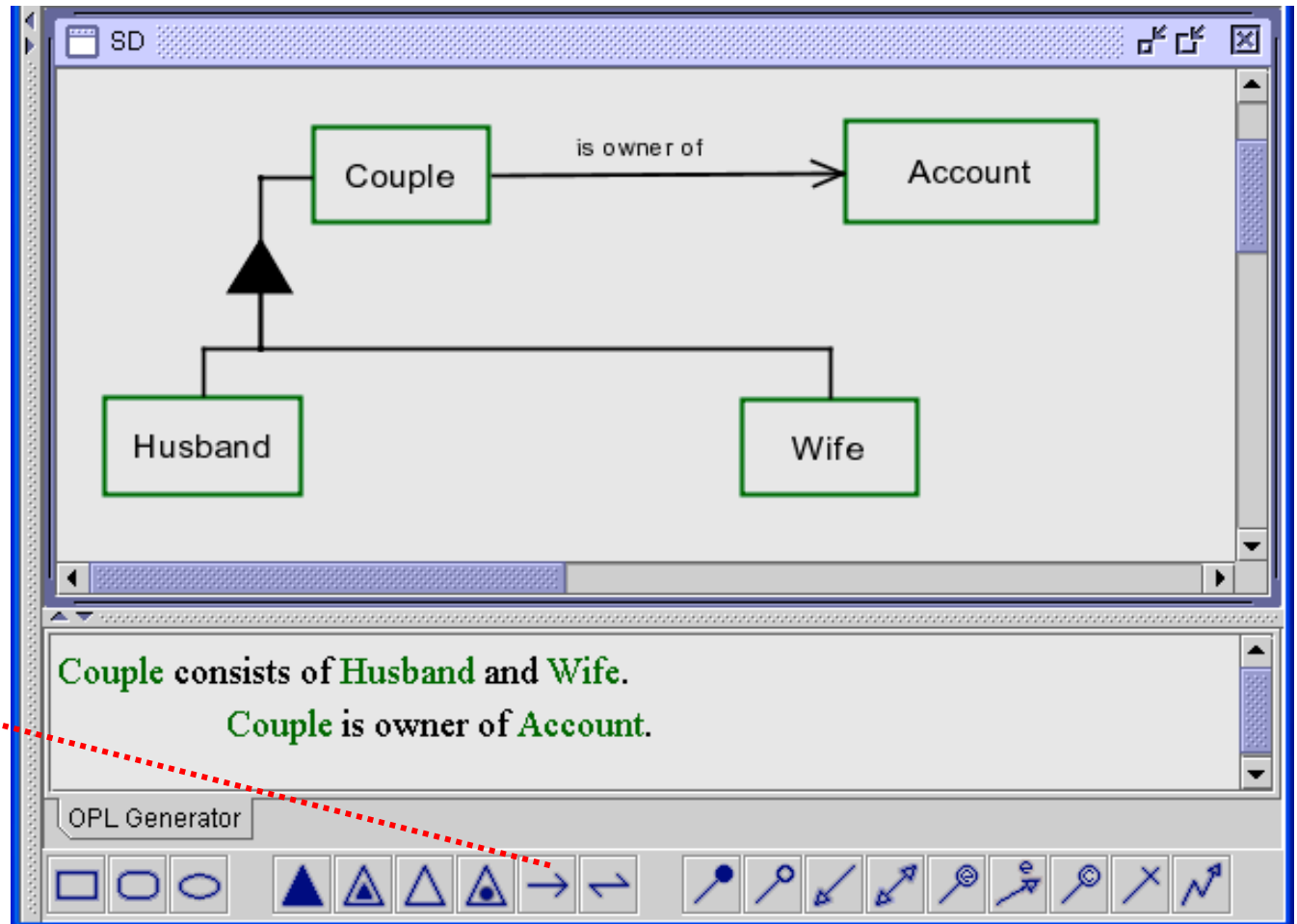
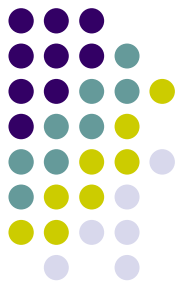
A structural relation between two things whose semantics is expressed through its tag



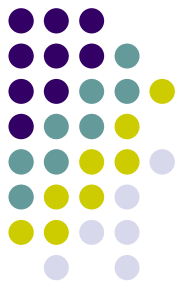
Unidirectional  
tagged  
structural link

# General tagged structural link

A structural relation between two things whose semantics is expressed through its tag

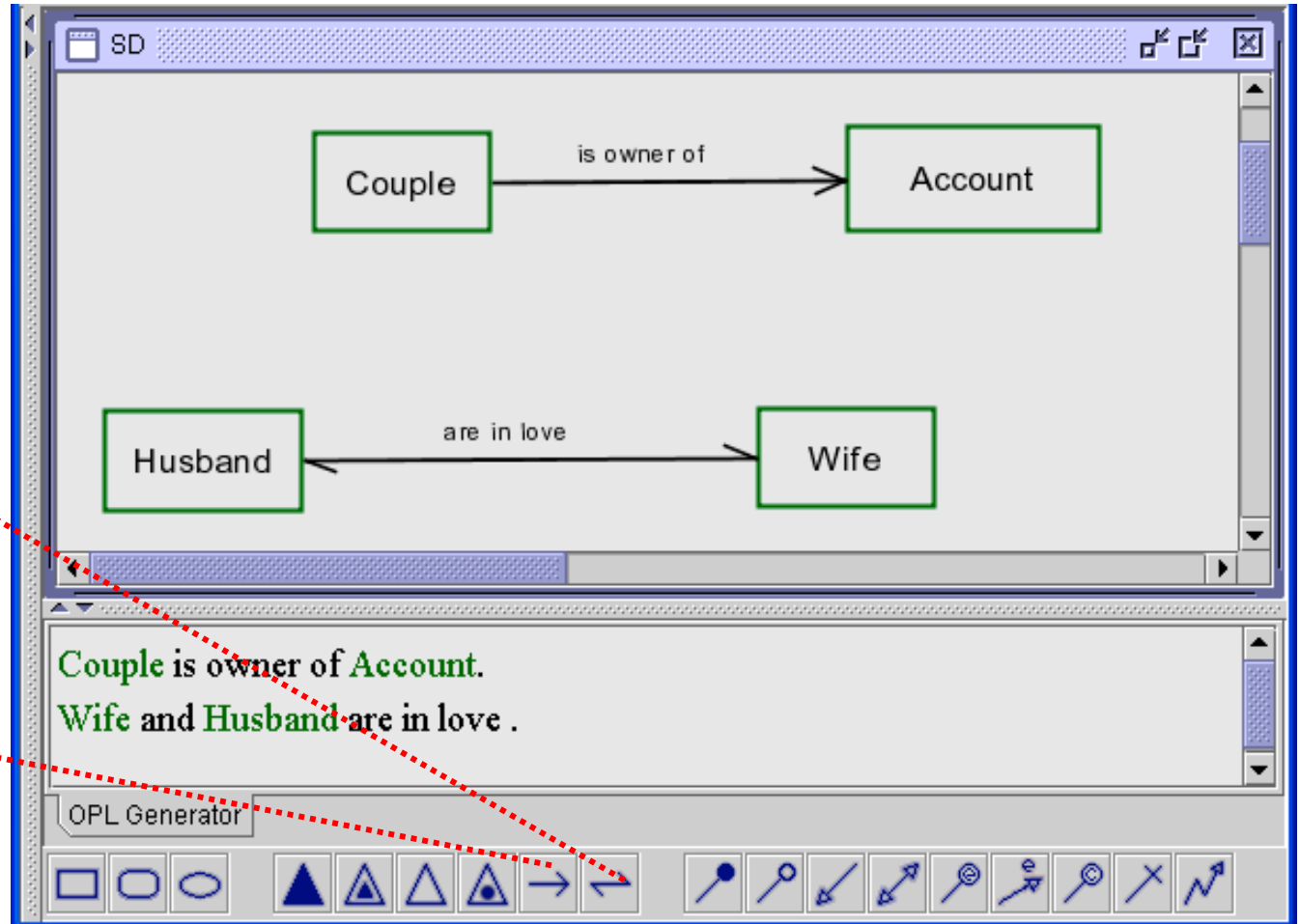


Unidirectional  
tagged  
structural link



# General tagged structural link

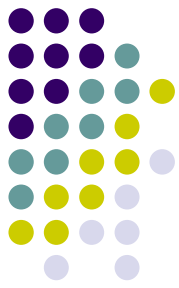
A structural relation between two things whose semantics is expressed through its tag



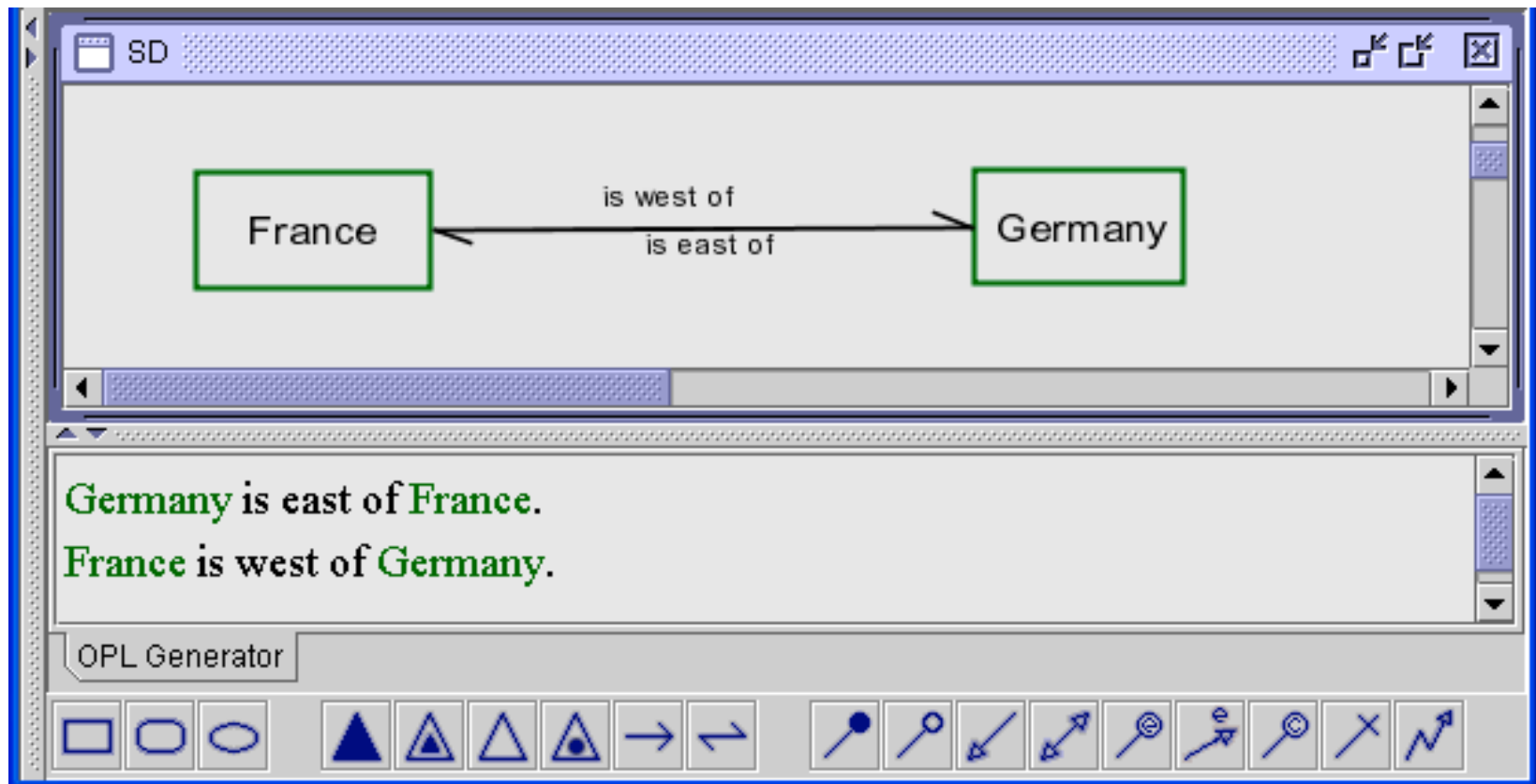
**Bidirectional  
tagged  
structural link**

**Unidirectional  
tagged  
structural link**

# Bi-directional tagged structural link with two tags



Each tag results in a separate sentence.

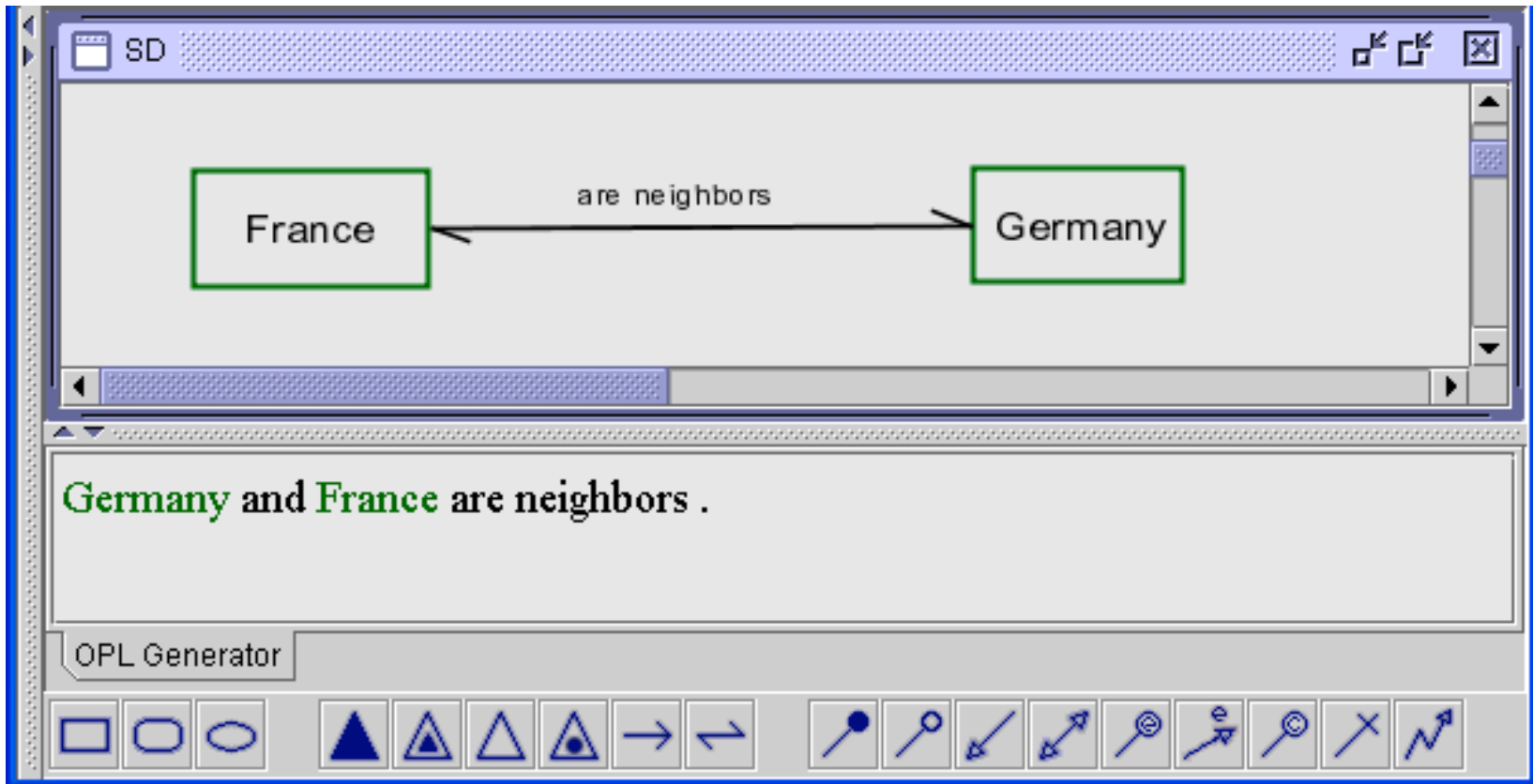




# Bi-directional tagged structural link with one tag

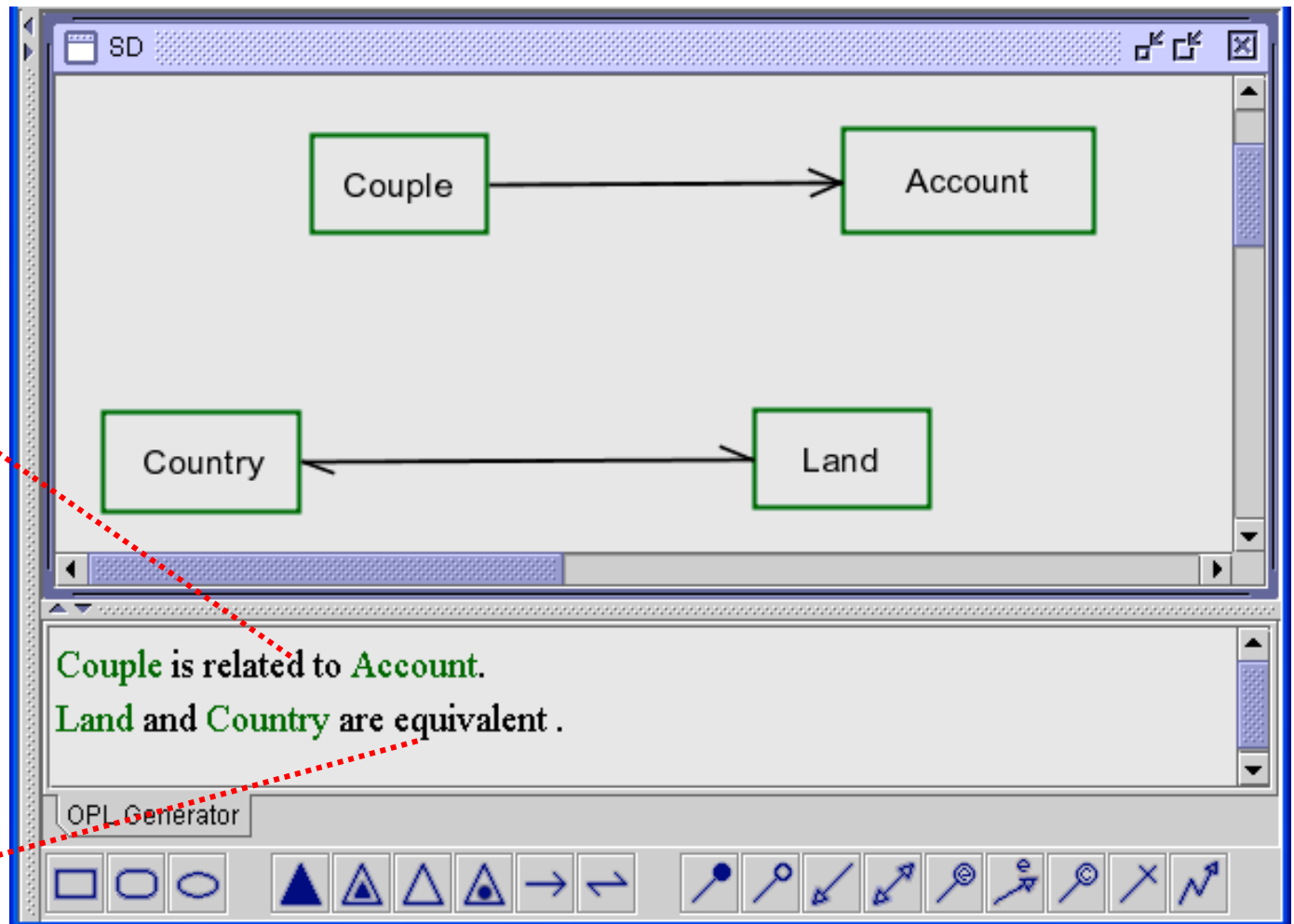
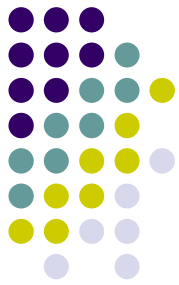


A structural relation between two things whose semantics is expressed through its tag



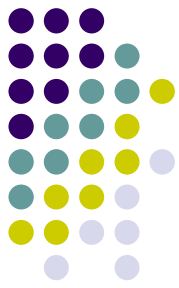
# Null tag structural link

A structural relation between two things with an empty tag



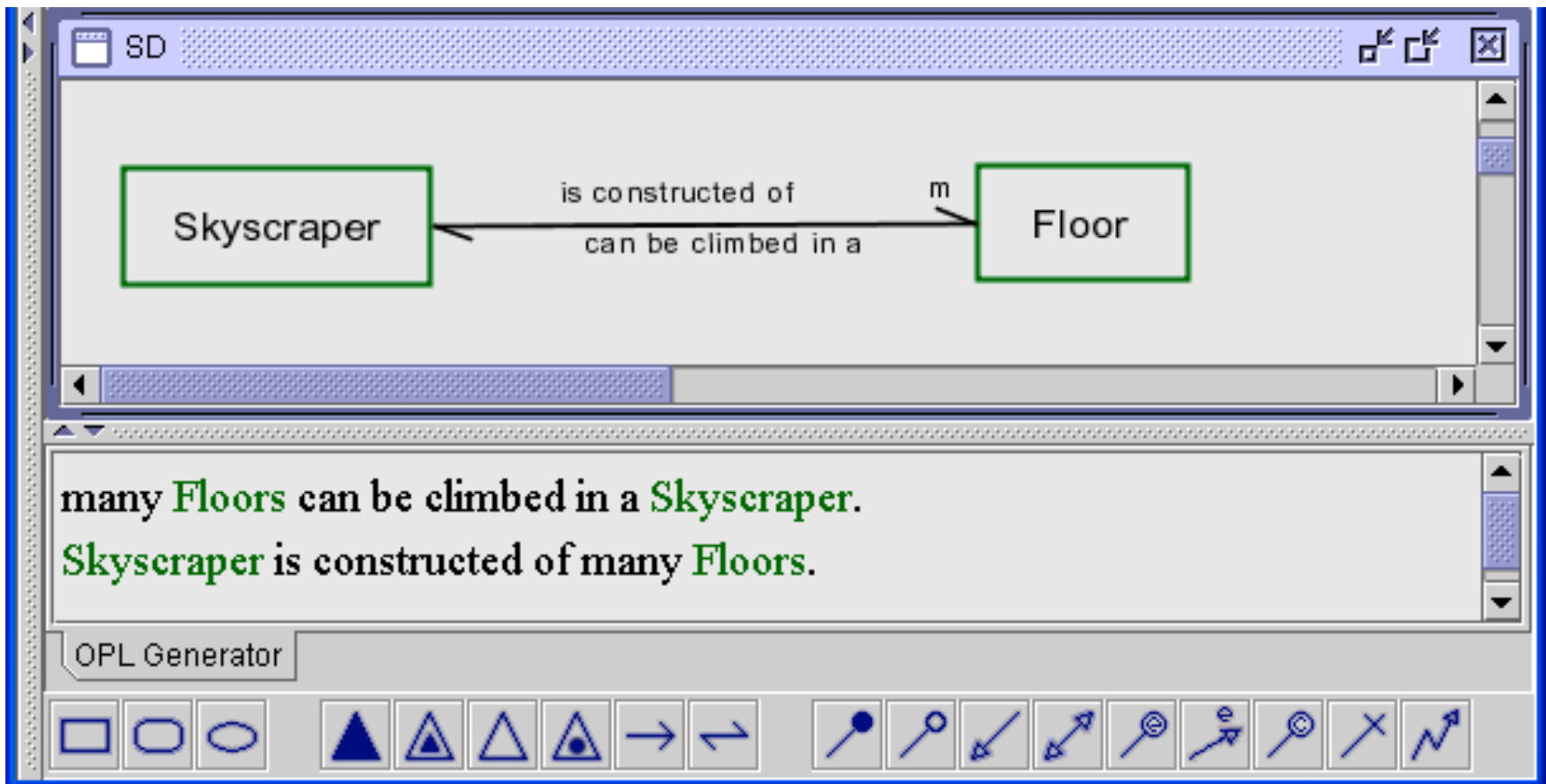
Unidirectional  
null tag OPL

Bidirectional  
null tag OPL



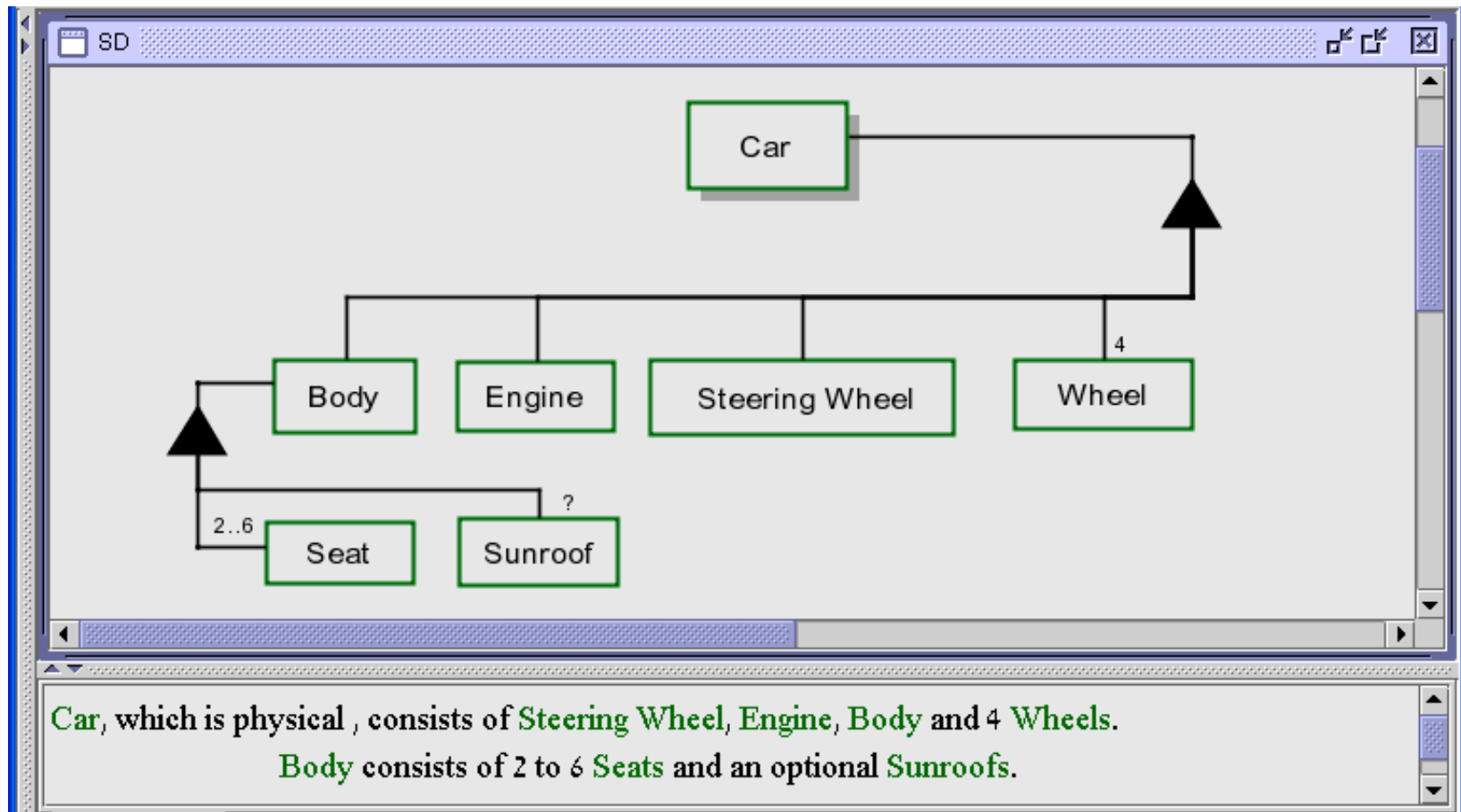
# Participation constraints

Structural relation can be associated with various quantities



# Participation constraints

The aggregation-participation link can have participation constraints on the parts



# Setting participation constraints

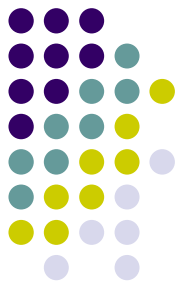


Each part in the aggregation-participation link can be set separately

The screenshot shows a dialog box titled "Aggregation-Participation Relation Properties" with a close button (X) in the top right corner. The dialog has two tabs: "General" and "Misc.". The "General" tab is active. It contains a "Source" section with a "Name" field set to "Body" and a "Cardinality" field set to "1". Below this is a "Destinations" section containing a table with three columns: "Name", "Participation Constr...", and "Value". The table has two rows: "Sunroof" and "Seat". The "Seat" row is selected, and a dropdown menu is open for its "Participation Constr..." field, showing options "1", "many", and "custom", with "custom" selected. The "Value" field for "Seat" contains "2..6". At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

Name	Participation Constr...	Value
Sunroof	custom	
Seat	custom	2..6

# Syntax and Semantics Consistency Checking



Trying to make a process part of an object results:

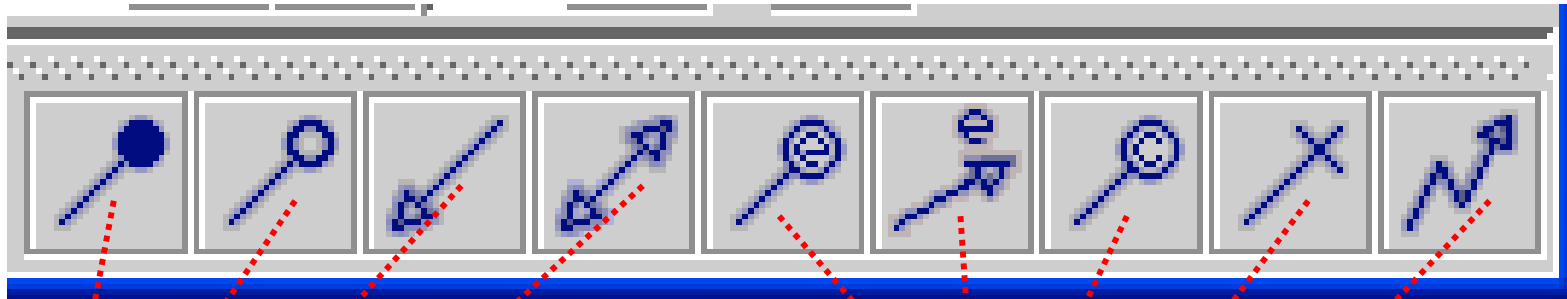
The screenshot shows a software development environment window titled "SD". Inside the window, a diagram illustrates a class hierarchy. A box labeled "PC" is at the top, with a solid black triangle pointing to it from a horizontal line below. This line connects to four boxes: "Box", "Monitor", "Keyborad", and "Mouse". To the right of this diagram is an oval labeled "Computing". Below the diagram, a text box contains the sentence: "PC consists of Box, Keyborad, Mouse and Monitor." At the bottom of the window, there is a toolbar with icons for shapes and a label "OPL Generator".

An error dialog box titled "Opcat2 - Error" is overlaid on the bottom right. It features a red octagonal stop sign icon and the following text: "An object and a process cannot be connected with an aggregation-participation relation." An "OK" button is located at the bottom of the dialog.

# OPM Procedural Links

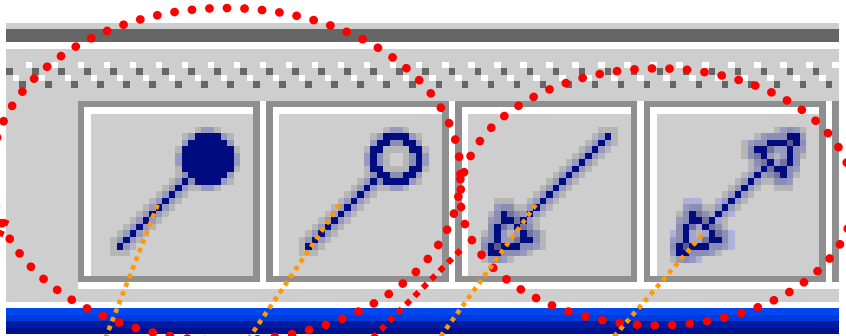


Links between a process and the object it transforms or a state of that object



- Agent link
- Instrument link
- Consumption/result link
- Effect link
- Event links
- Condition link
- Exception link
- Invocation link

# Two Procedural Link Types



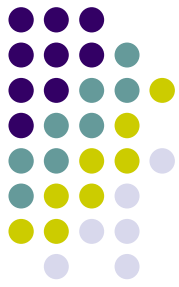
- Enabling Links

- Agent link
- Instrument link

- Transforming Links

- result/consumption/input/output link
- Effect link... more

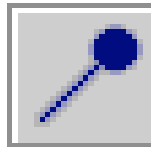




# Enabling Links

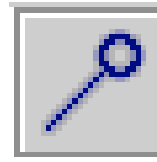
- Link objects that enable the process but are not transformed by it

- Agent link



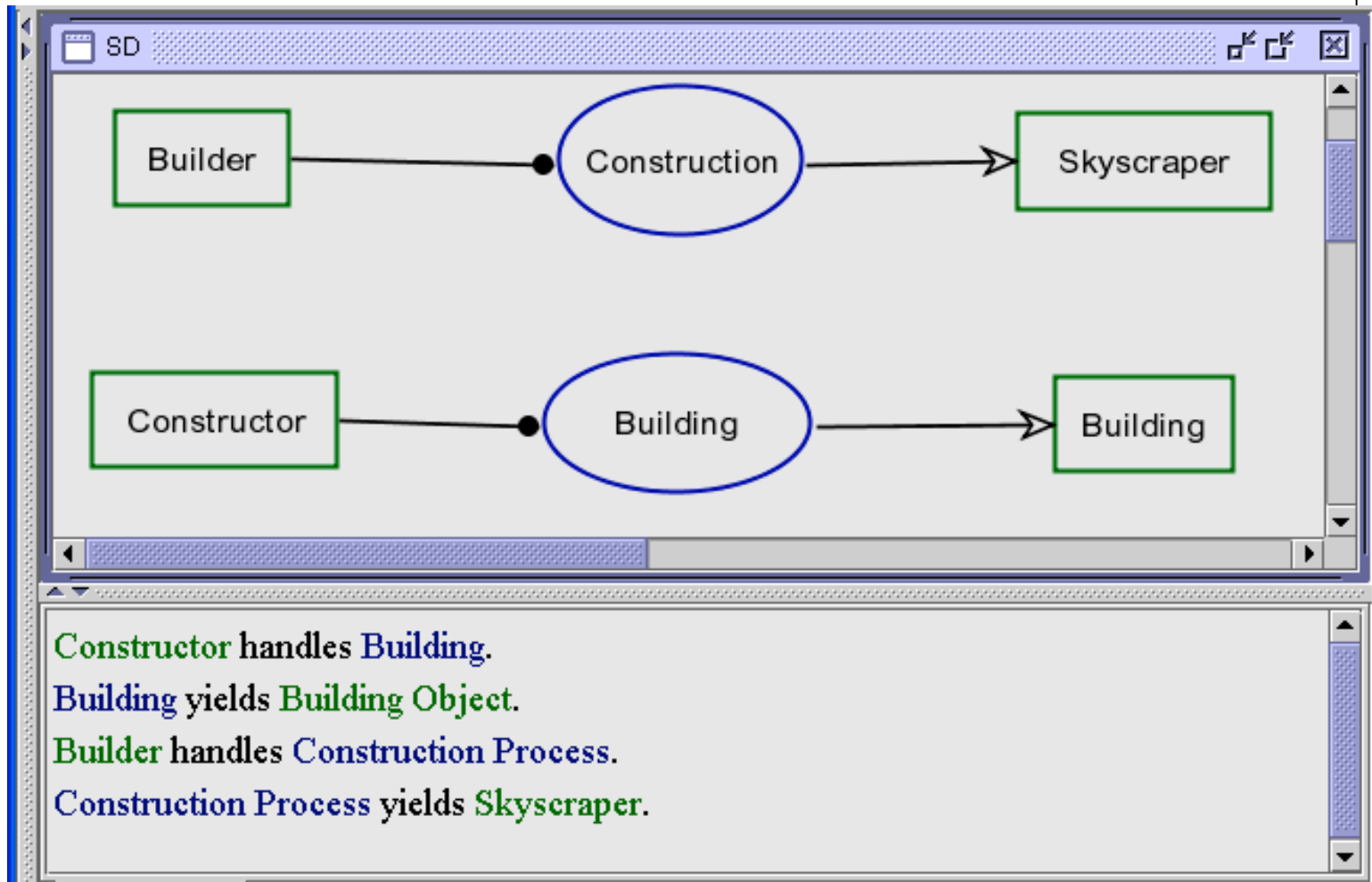
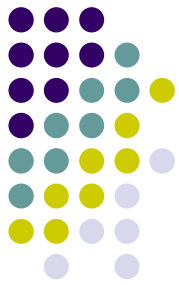
Links a human or a group of humans that trigger the process or participate in it

- Instrument link



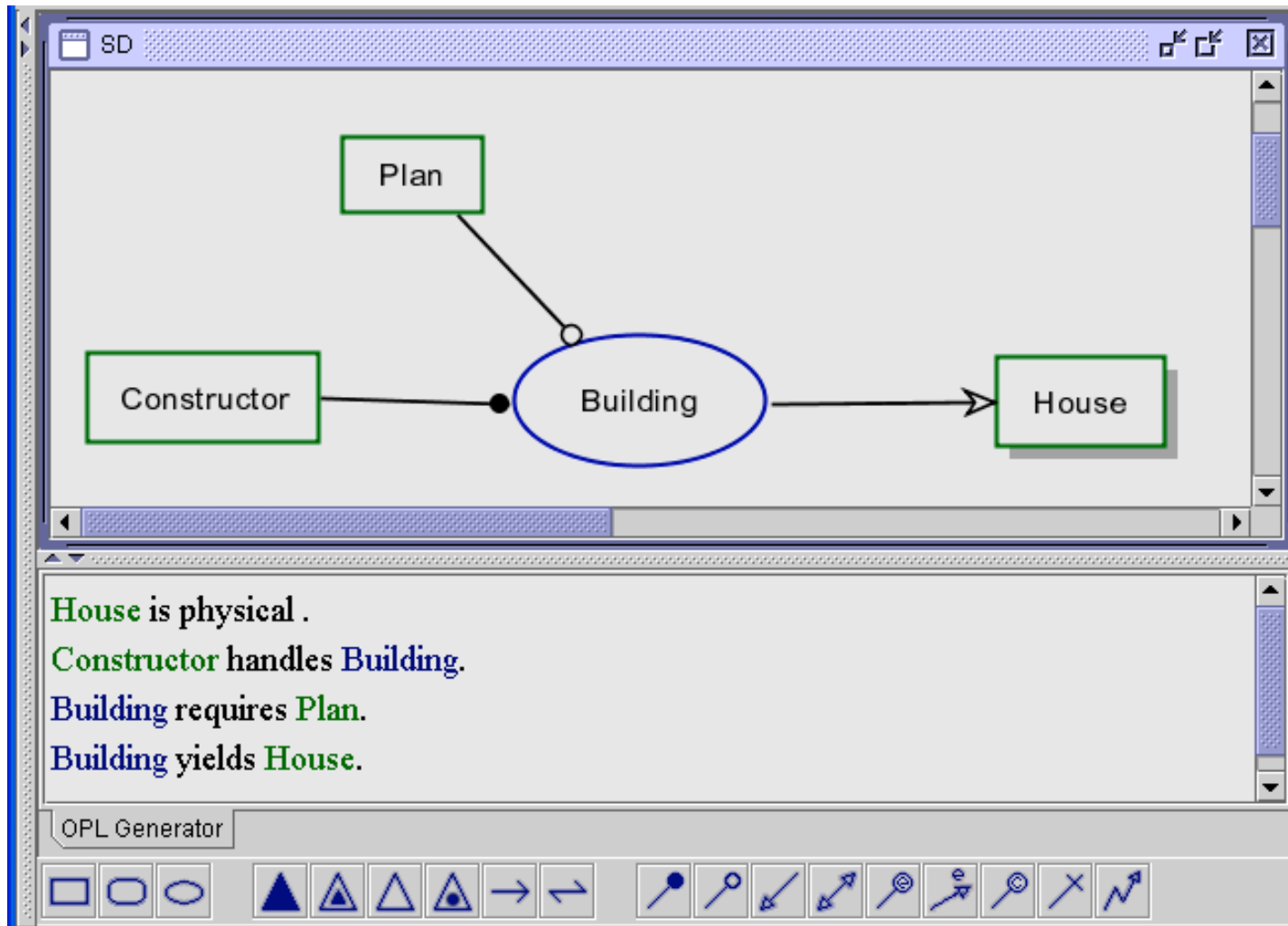
Links a non-human object that the process requires in order for it to occur or execute

# Agent Link

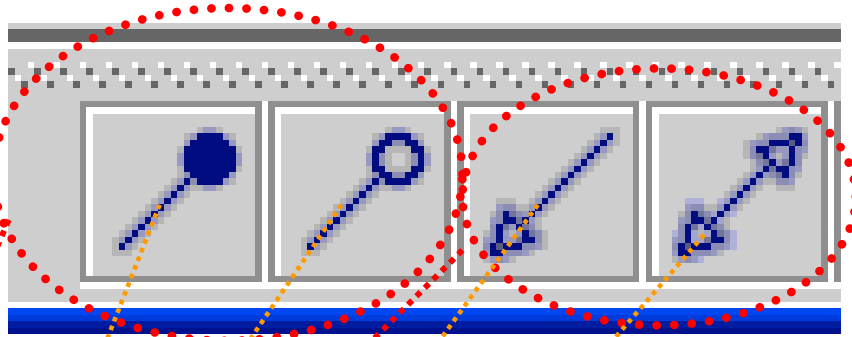




# Instrument Link



# Procedural Links



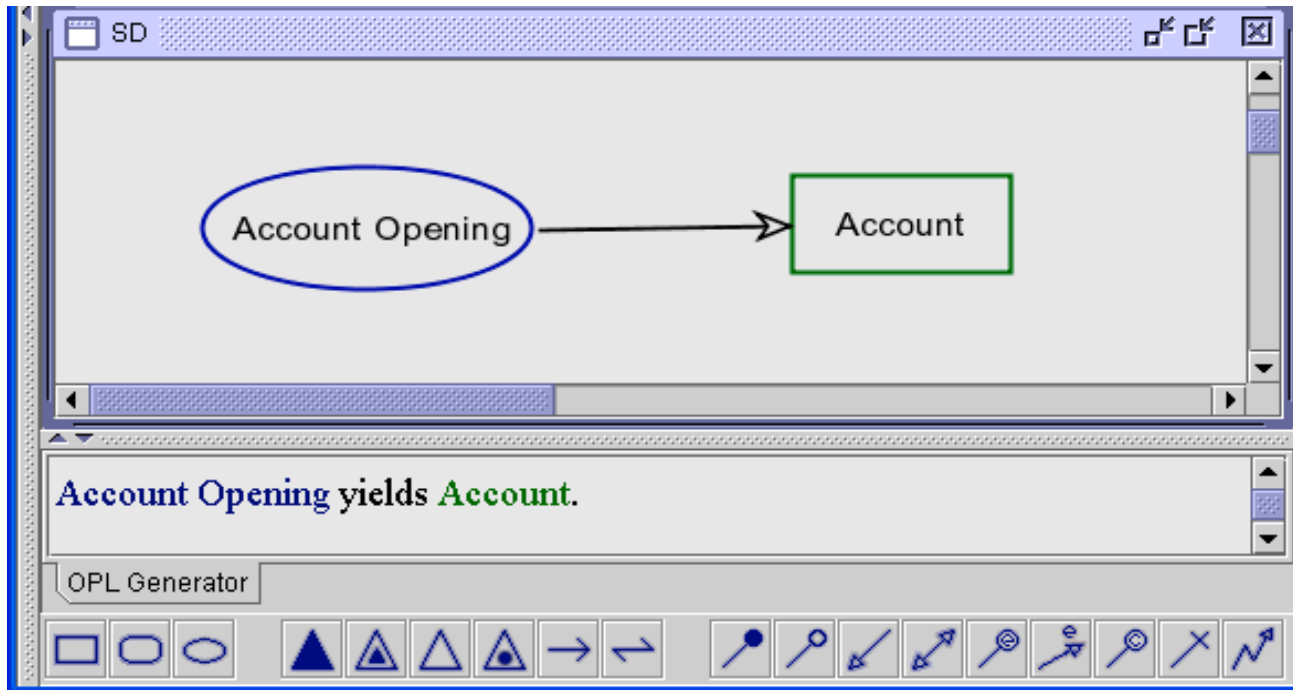
- Enabling Links

- Agent link
- Instrument link

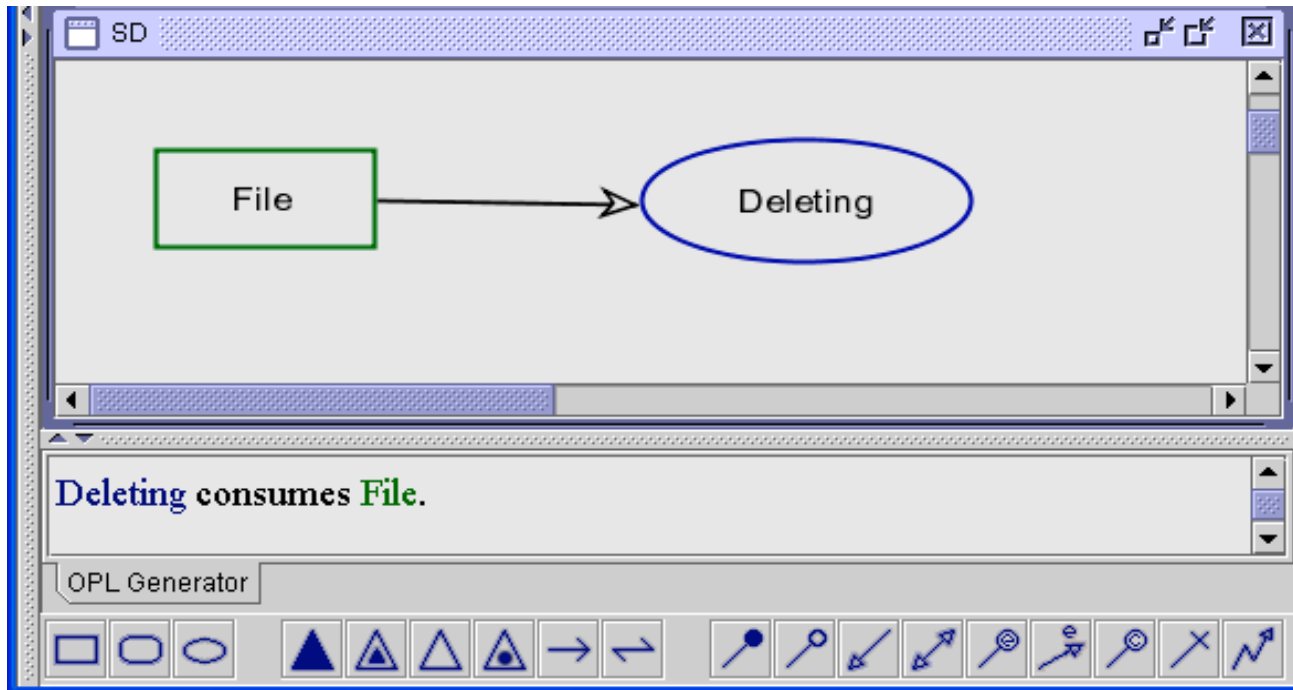
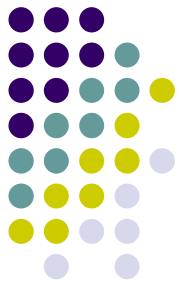
- Transforming Links

- result/consumption/input/output link
- Effect link

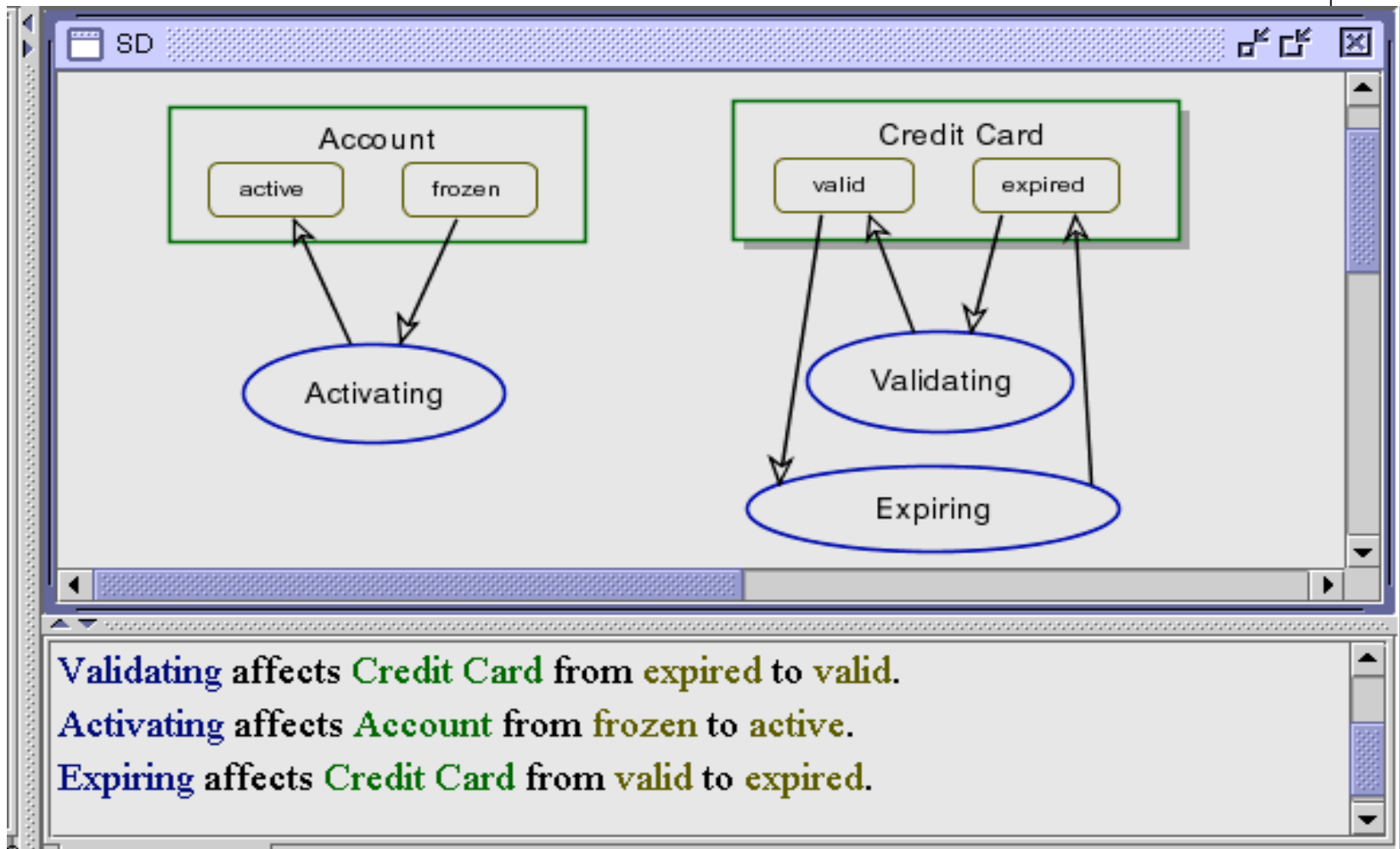
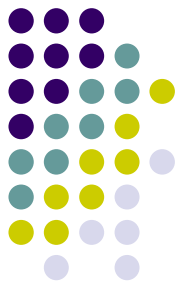
# Creating a new object: Result link



# Consuming an existing object: Consumption link

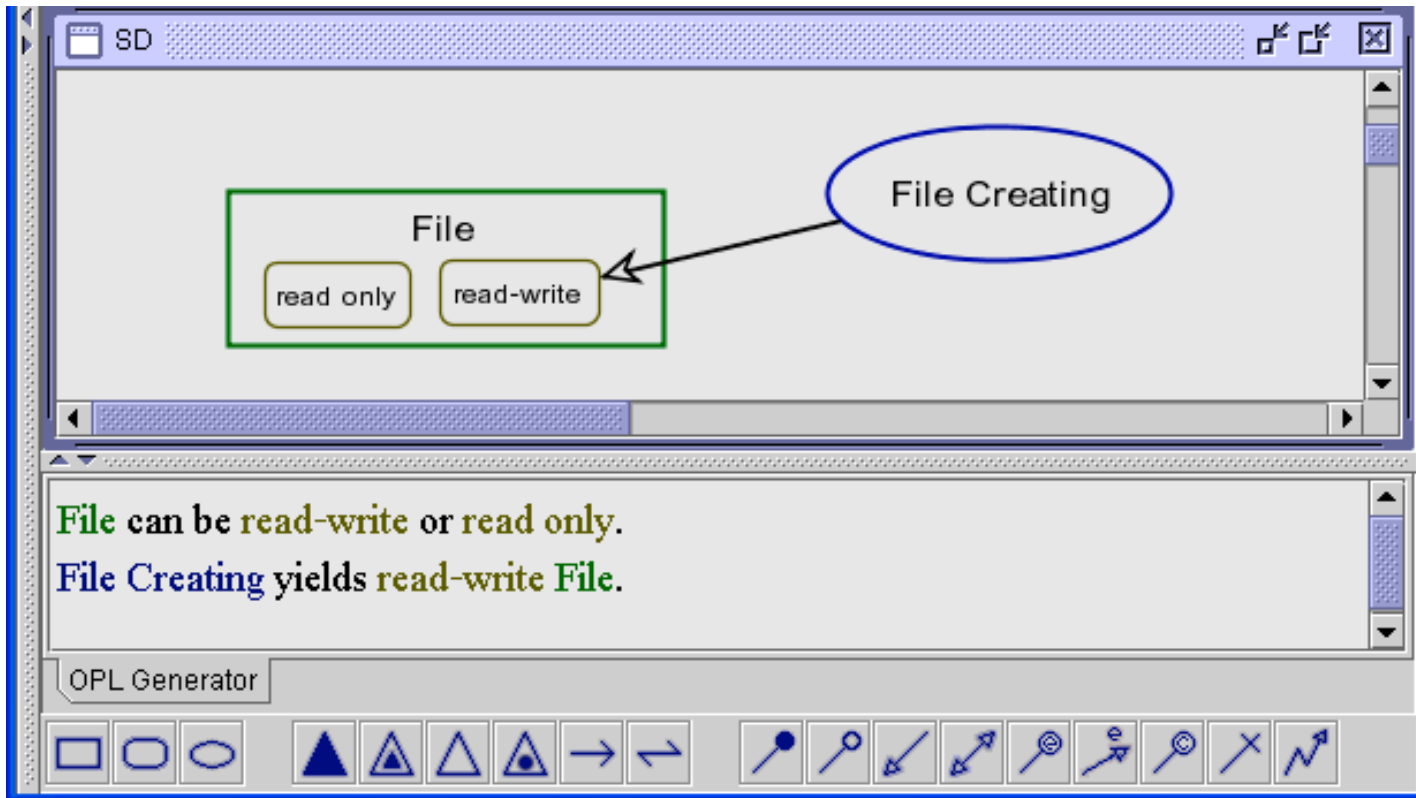
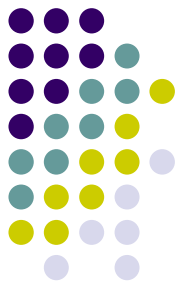


# Changing object state



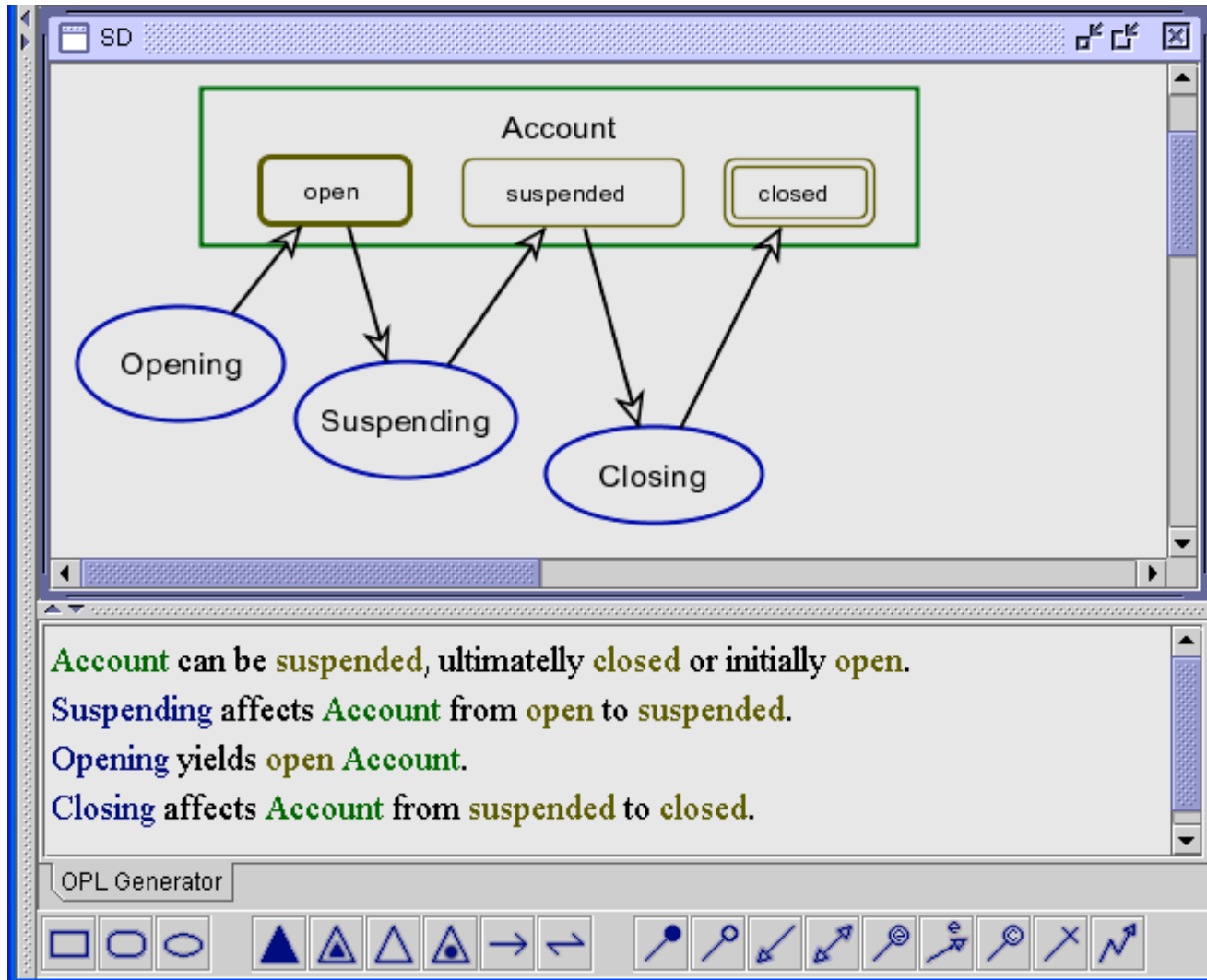
**Validating** affects **Credit Card** from **expired** to **valid**.  
**Activating** affects **Account** from **frozen** to **active**.  
**Expiring** affects **Credit Card** from **valid** to **expired**.

# State-specified object generation





# States can be initial or final





# Object Essence

Can be informatical (default) or physical

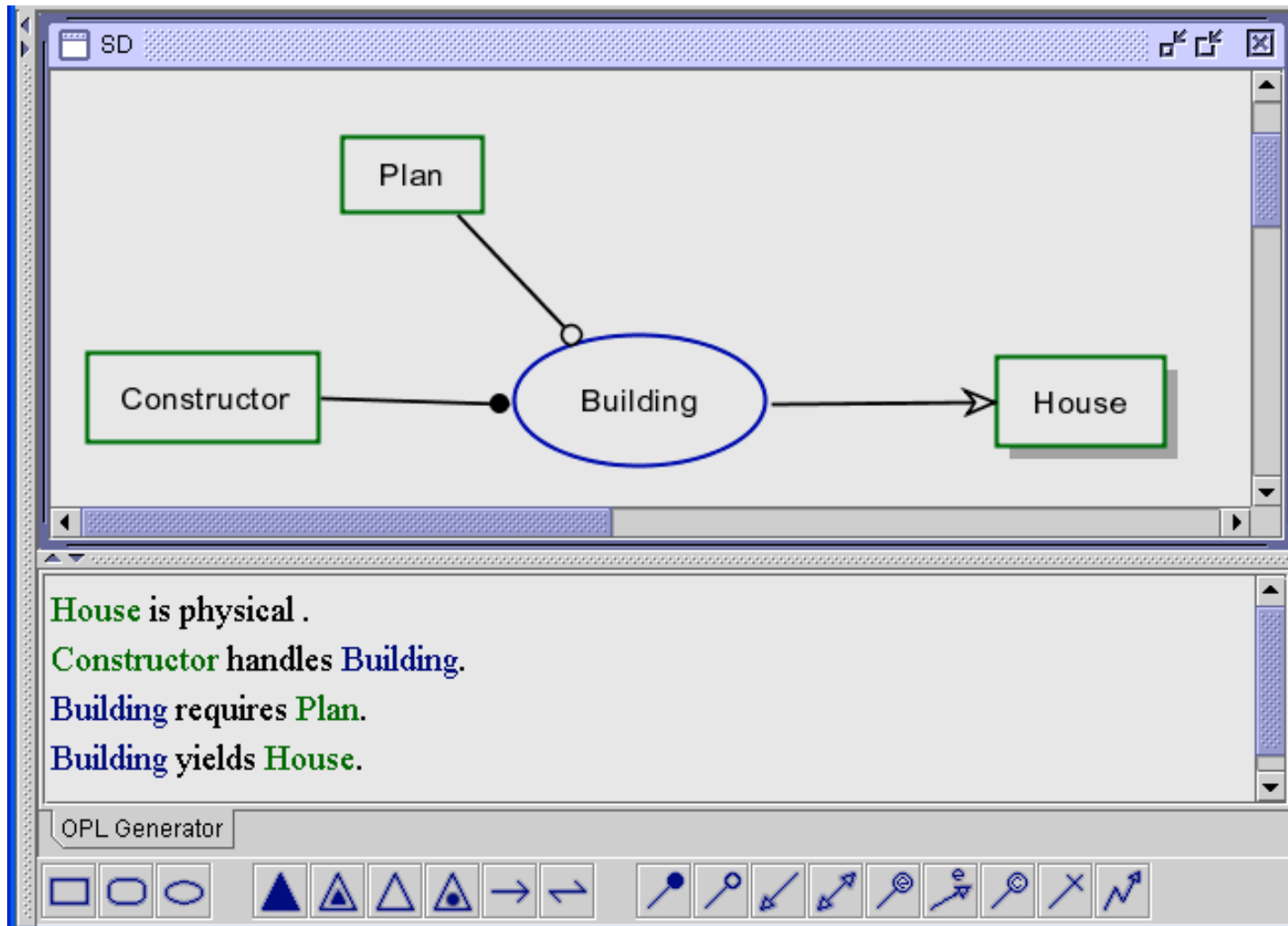
The screenshot shows a software window titled "SD" with a toolbar and a scroll bar. The main area contains two rectangular boxes labeled "Account" and "Credit Card". A yellow callout box points to the "Credit Card" box with the text "Object-Process Diagram (OPD)". Below the diagram area, a text field contains the sentence "Credit Card is physical ." in green text. A yellow callout box points to this text with the text "Object-Process Language (OPL) sentence(s)". At the bottom, there is a toolbar labeled "OPL Generator" with various icons for creating shapes and lines.



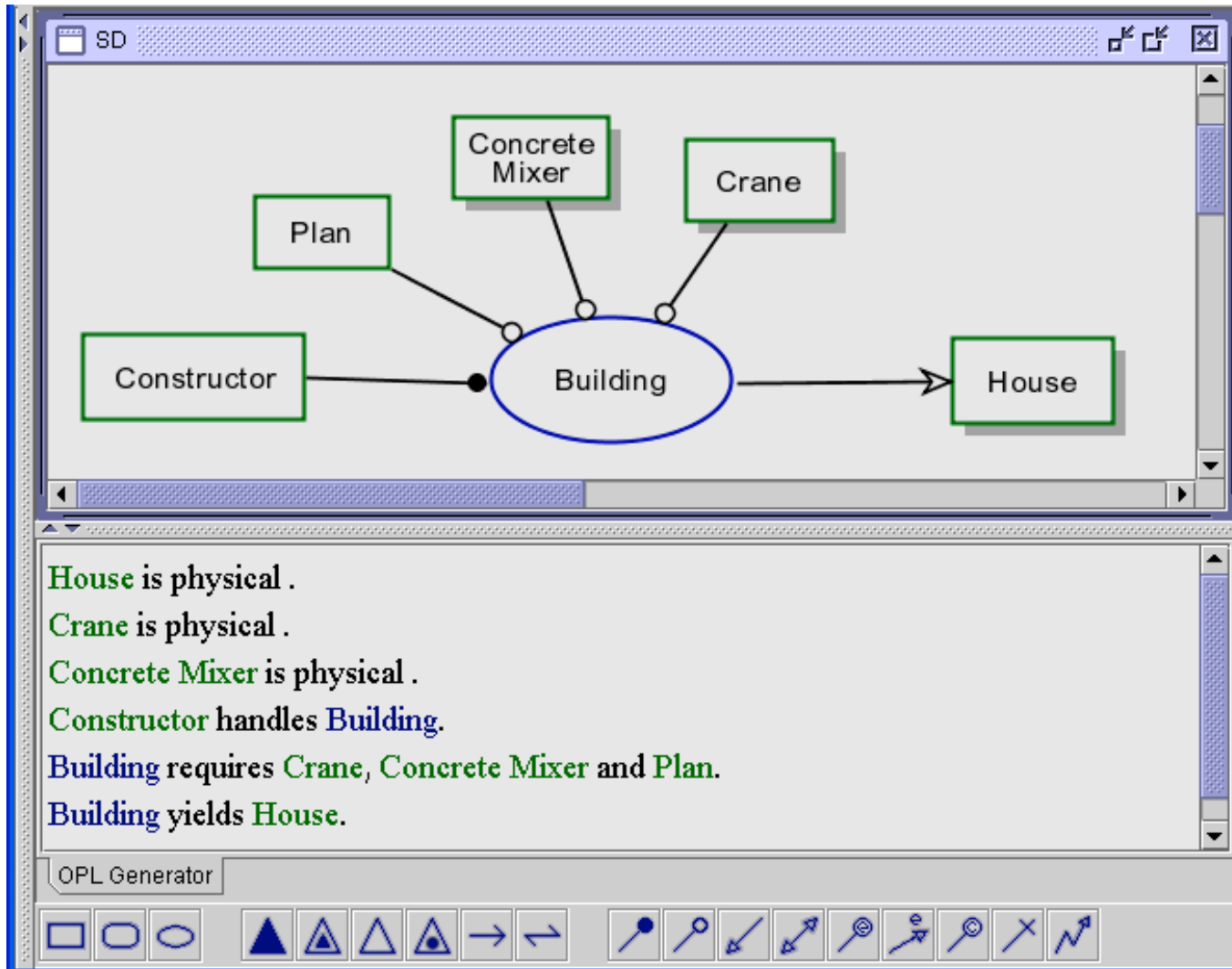
# Processes transform objects

- Three transformation options:
  - Change an object state
  - Create (generate) a new object
  - Consume (destroy) an existing object

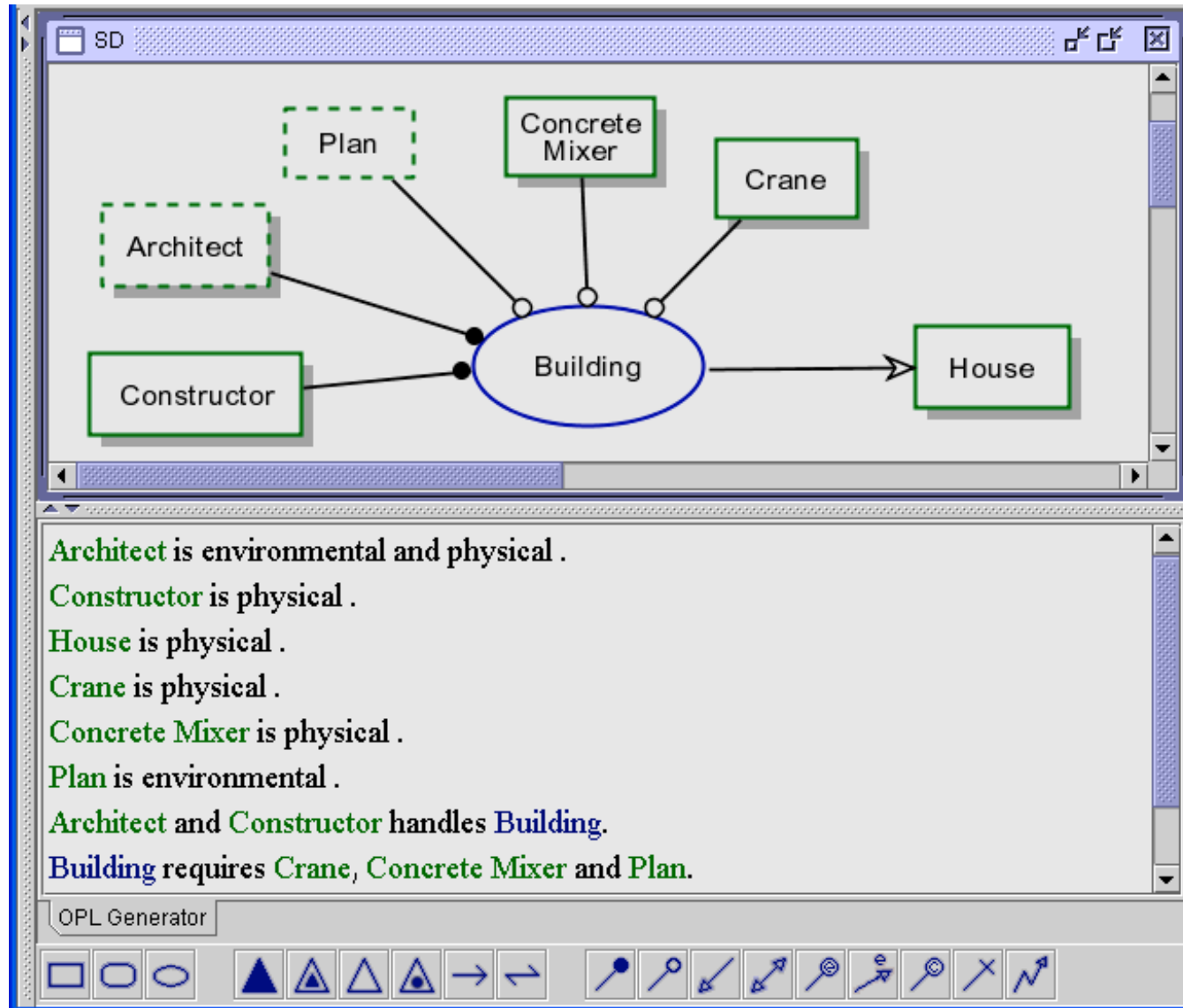
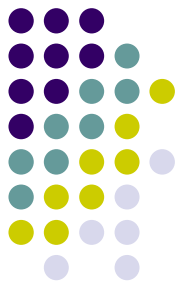
# Enabling Links: Agent and Instrument



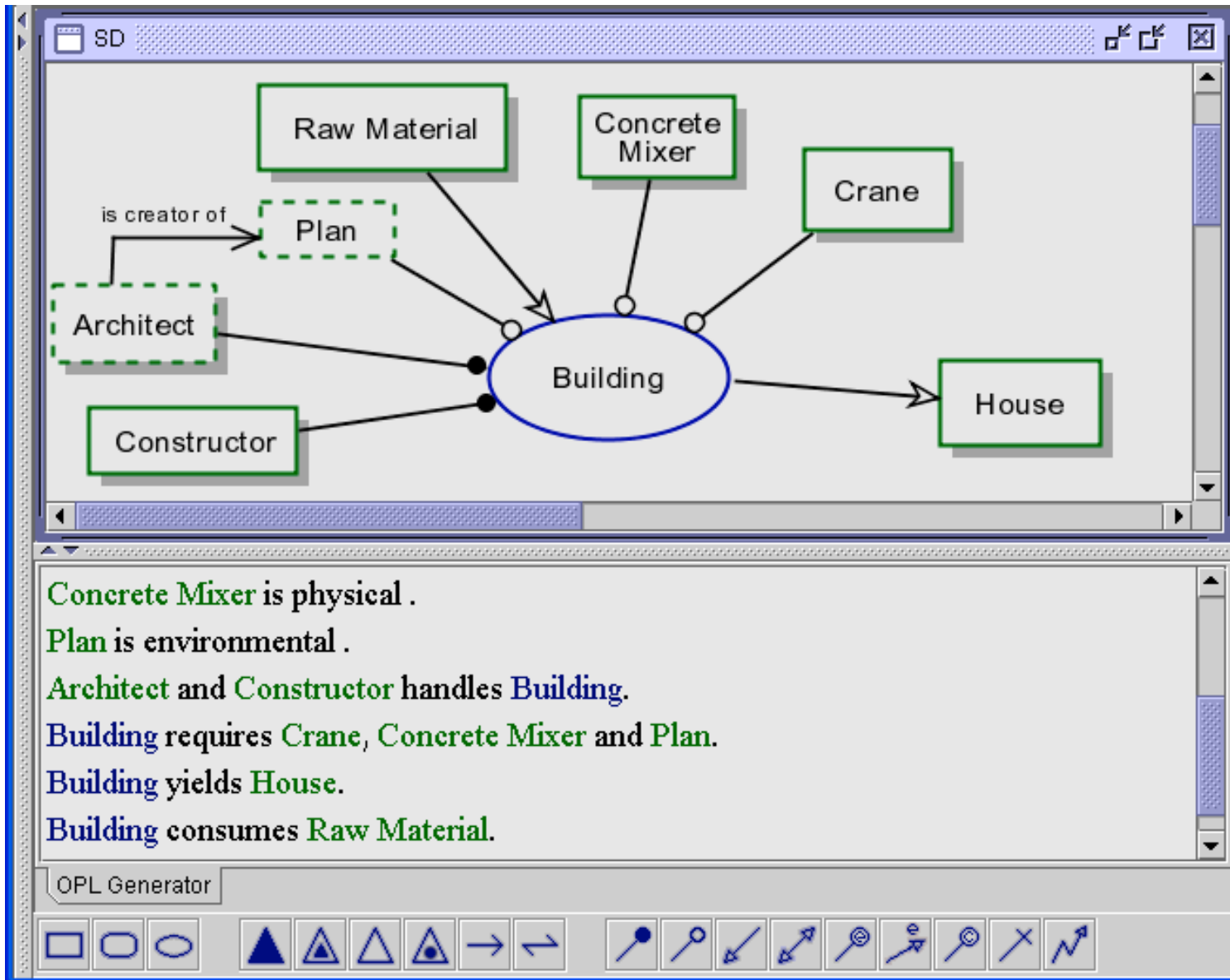
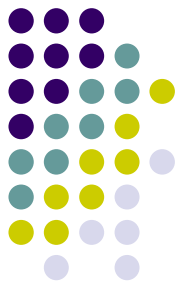
# Adding physical Instruments



# Systemic vs. environmental objects



# Raw Material is consumed, House created



# Complexity Management in OPM



Three **refinement/abstraction**  
mechanisms:

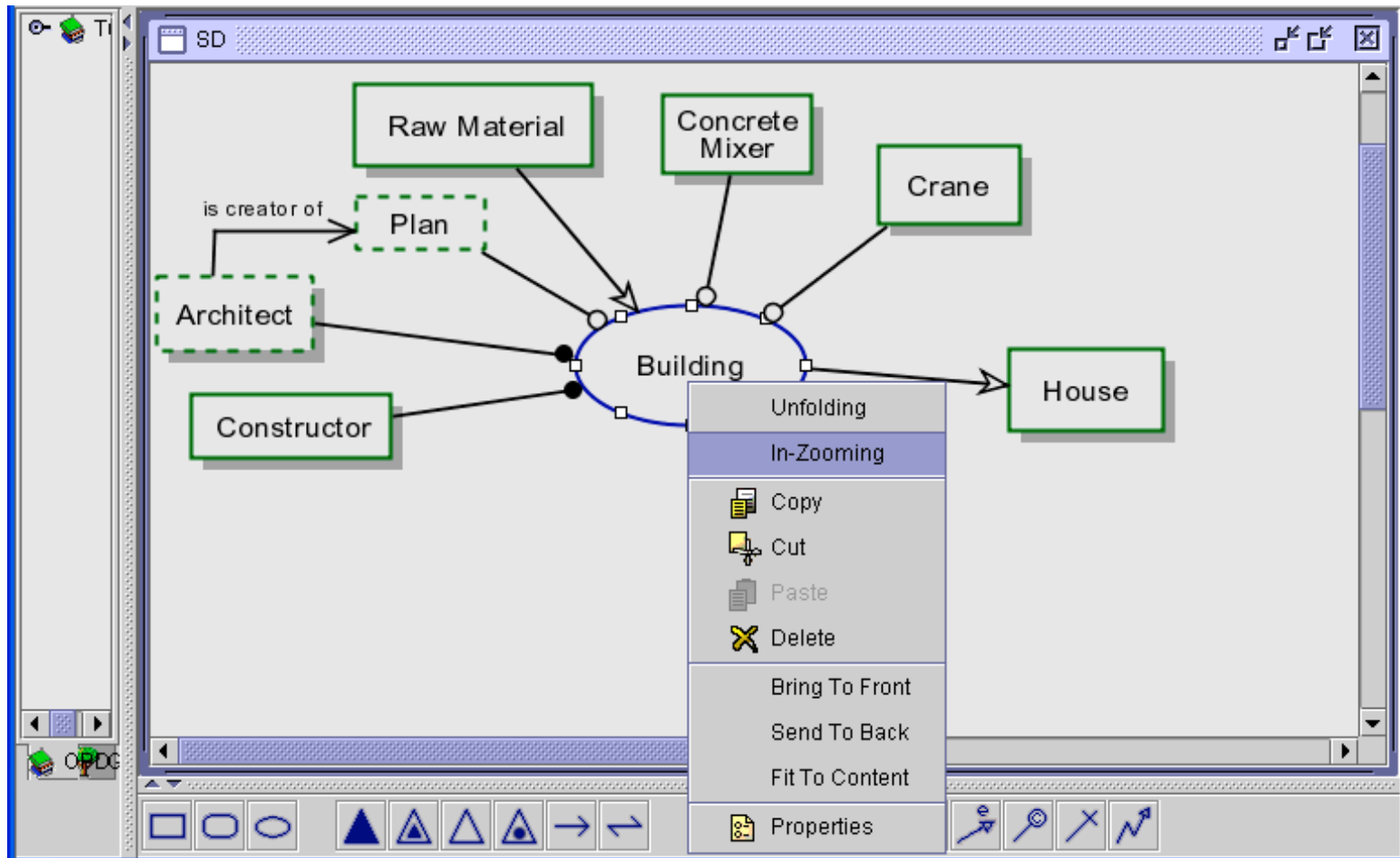
- **In-zooming/out-zooming**
- **Unfolding/folding**
- **State expression/state suppression**



# In-zooming

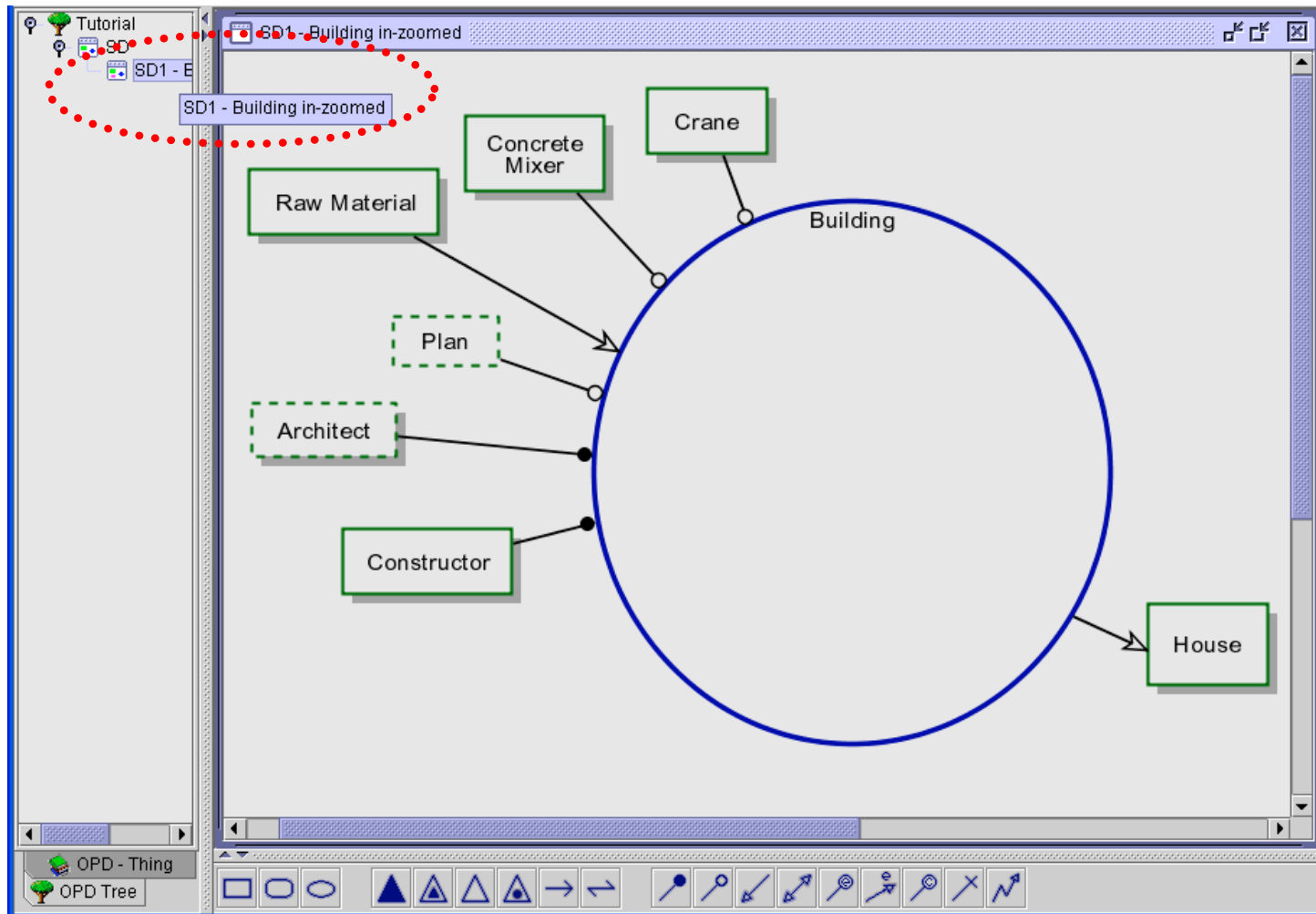


Exposing the details inside a complex process



# In-zooming

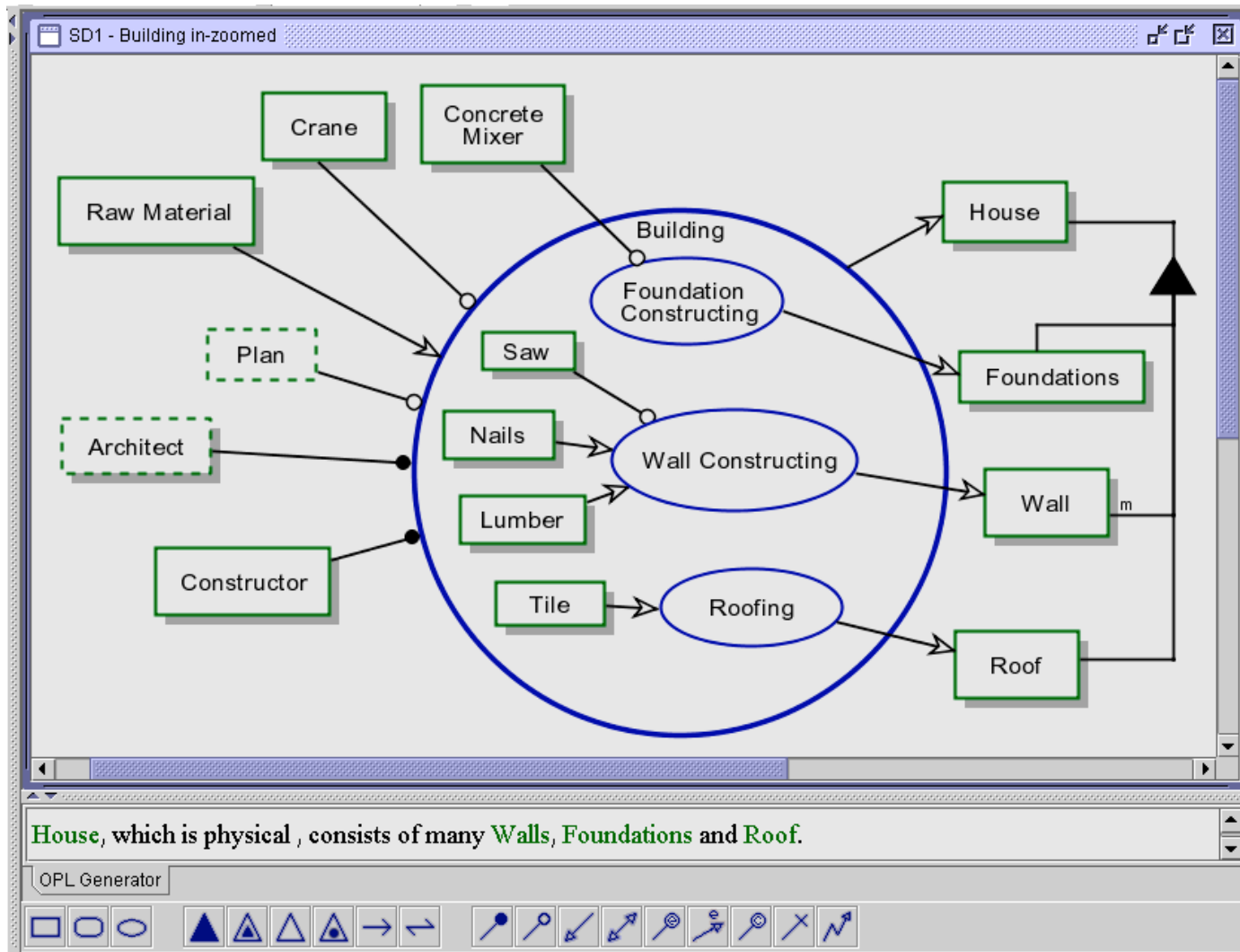
A new OPD is created with Building enlarged



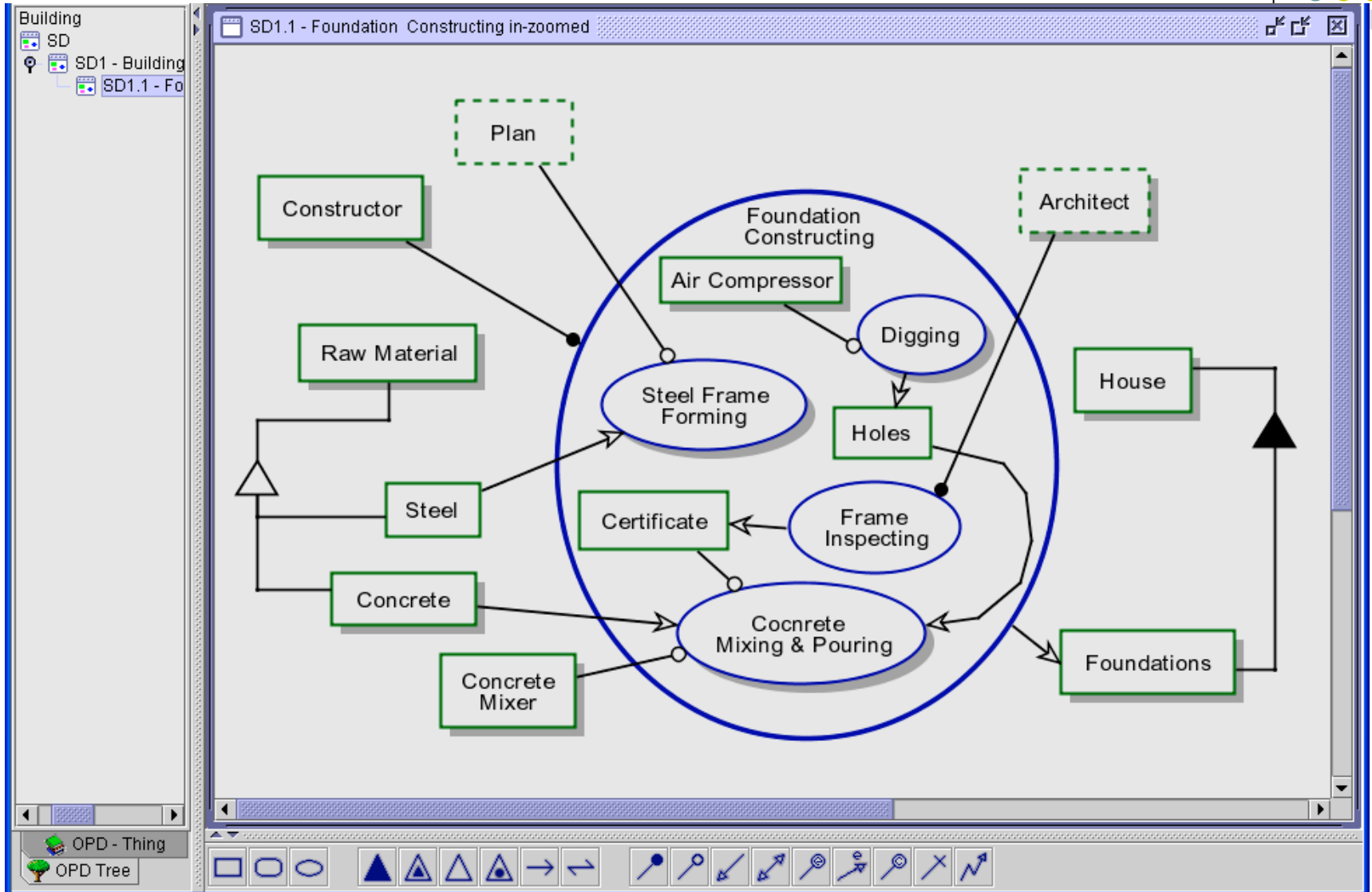
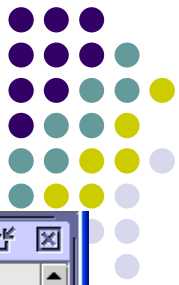
# In-zooming



The sub-processes of the Building process are depicted inside the in-zoomed process ellipse



# In-zooming is recursive



# The OPL paragraph:



The screenshot shows the Opcat II - Building software interface. The main window displays an OPL paragraph with the following text:

House, which is physical , consists of Foundations.  
Foundations is physical .  
Steel is a Raw Material.  
Concrete, which is physical , is a Raw Material.  
Architect is environmental and physical .  
Raw Material is physical .  
Concrete Mixer is physical .  
Plan is environmental .  
Air Compressor is physical .  
Constructor is physical .  
Steel Frame Forming, which is physical, requires Plan.  
Steel Frame Forming consumes Steel.  
Digging, which is physical, requires Air Compressor.  
Digging yields Holes.  
Cocnrete Mixing & Pouring, which is physical, requires Certificate and Concrete Mixer.  
Cocnrete Mixing & Pouring consumes Holes and Concrete.  
Constructor handles Foundation Constructing.  
Foundation Constructing yields Foundations of House.  
Architect handles Frame Inspecting.  
Frame Inspecting yields Certificate.

The interface includes a menu bar (System, Edit, View, Notation, Operation, Generation, Help), a toolbar with various icons, a left-hand tree view showing the project structure (Building, SD, SD1 - Building in-zoomed, SD1.1 - Foundation Constructing in-zoomed), and a bottom toolbar with additional icons. The title bar reads "Opcat II - Building".

# Simulation by animation



**Opcat II - Building**

System Edit View Notation Operation Generation Help

```
graph TD; Architect[Architect] -- "is creator of" --> Plan[Plan]; RawMaterial[Raw Material 11] --> Building((Building 2)); ConcreteMixer[Concrete Mixer] --> Building; Crane[Crane] --> Building; Constructor[Constructor] --> Building; Building --> House[House];
```

Building

- SD
- SD1 - Building in-zc
- SD1.1 - Founda

Raw Material 11

Concrete Mixer

Crane

Architect

Plan

Constructor

Building 2

House

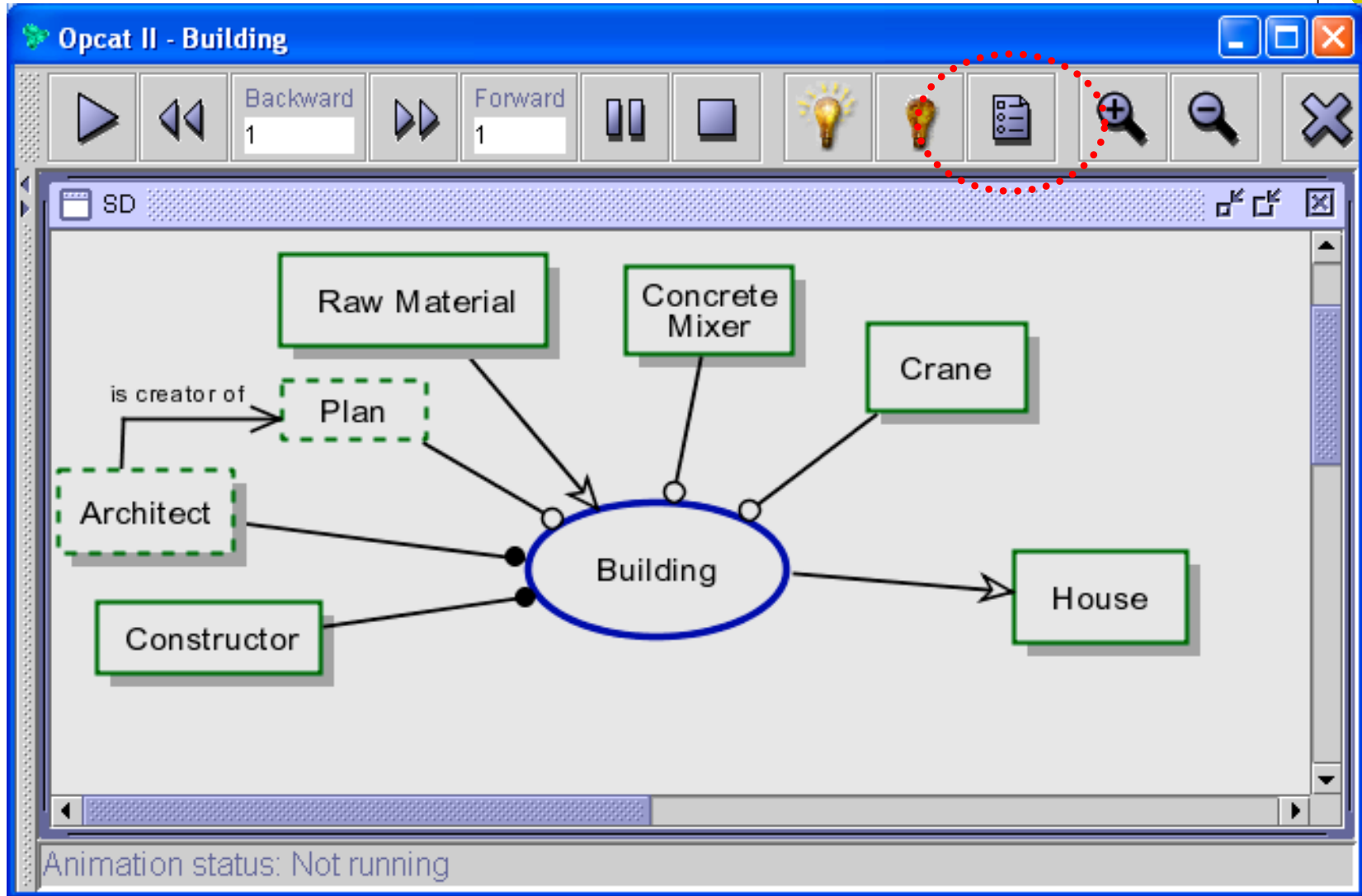
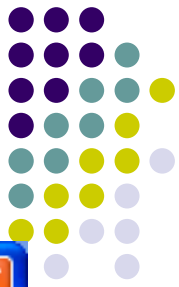
Raw Material is physical .

Constructor is physical .

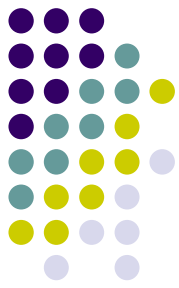
Architect, which is environmental and physical , is creator of Plan.

OPL Generator

# Simulation setup



# Animation parameter setup



**Animation Settings** [X]

Step Duration:  msec

Process Duration:  Fixed  steps  
 Random  -  steps

Reaction Time:  Fixed  steps  
 Random  -  steps

Default Object Instances:  One  
 Many

Use Automatic Initiation:

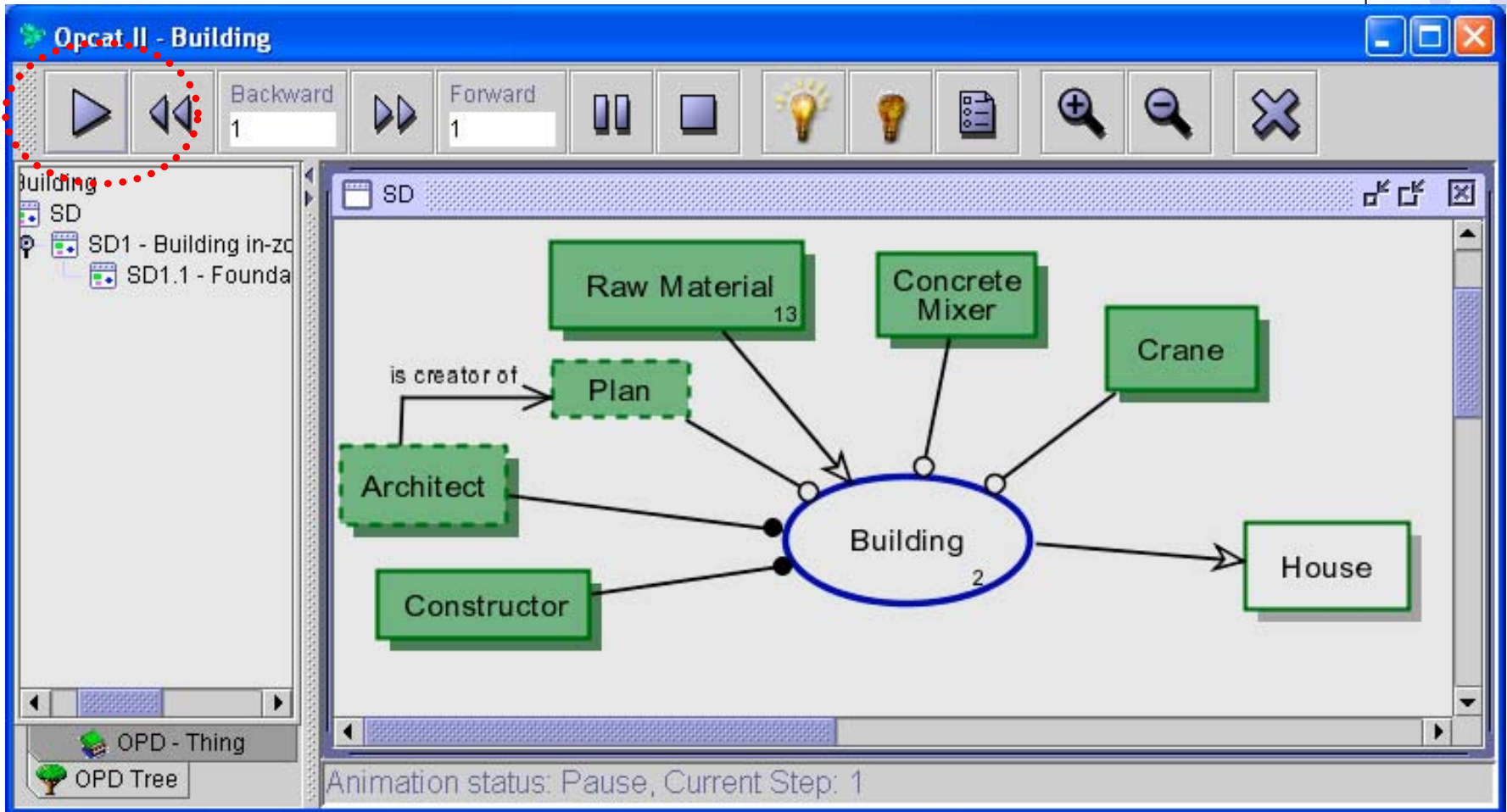
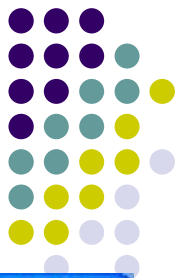
Automatic Move Between OPD:

Animation Mode:  Continuous  
 Step by Step

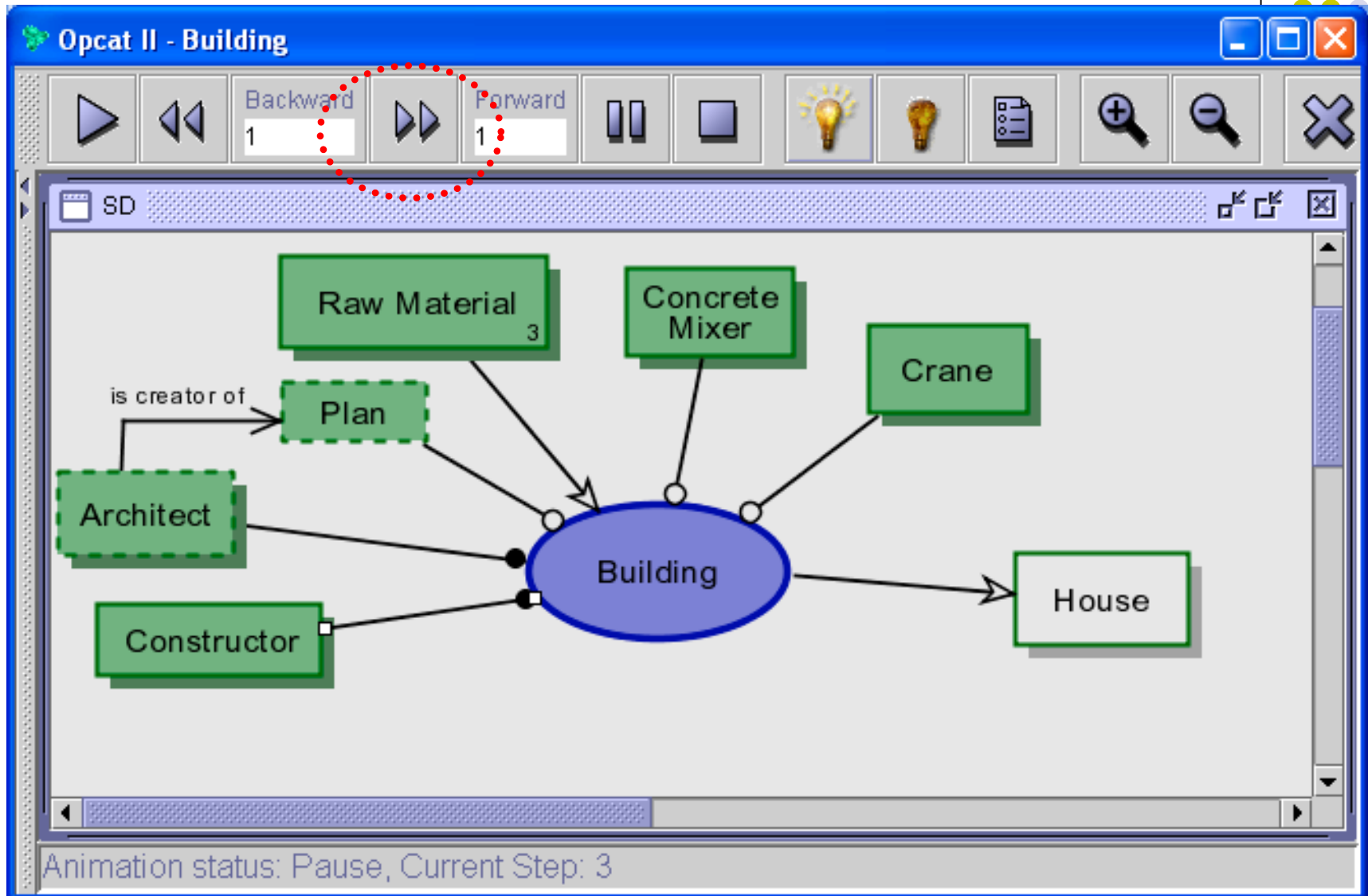
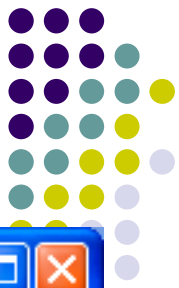
Random State Selection



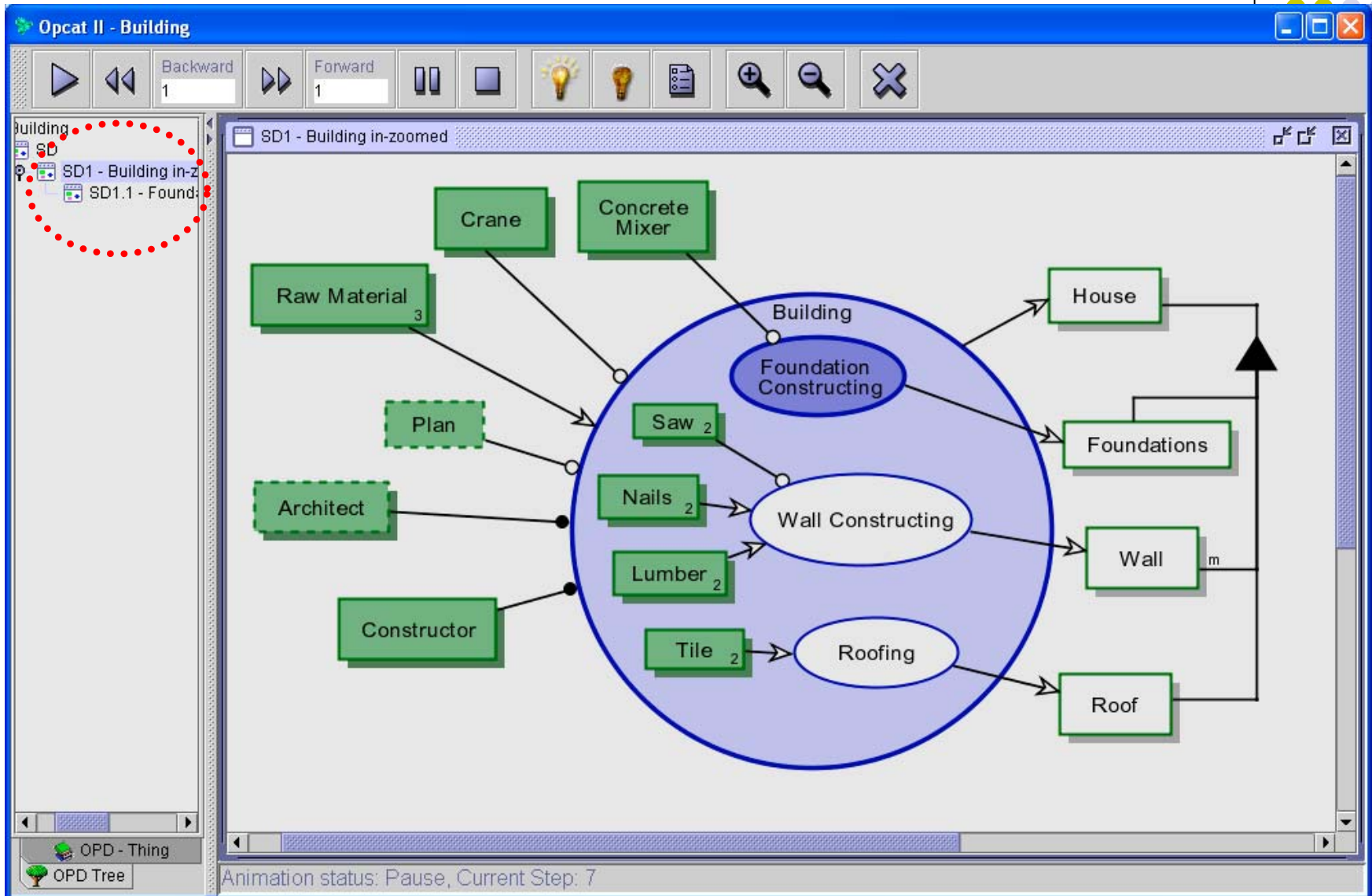
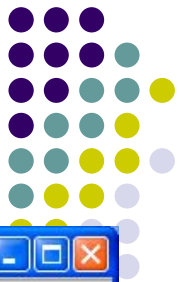
# Green objects already exist



# Building started

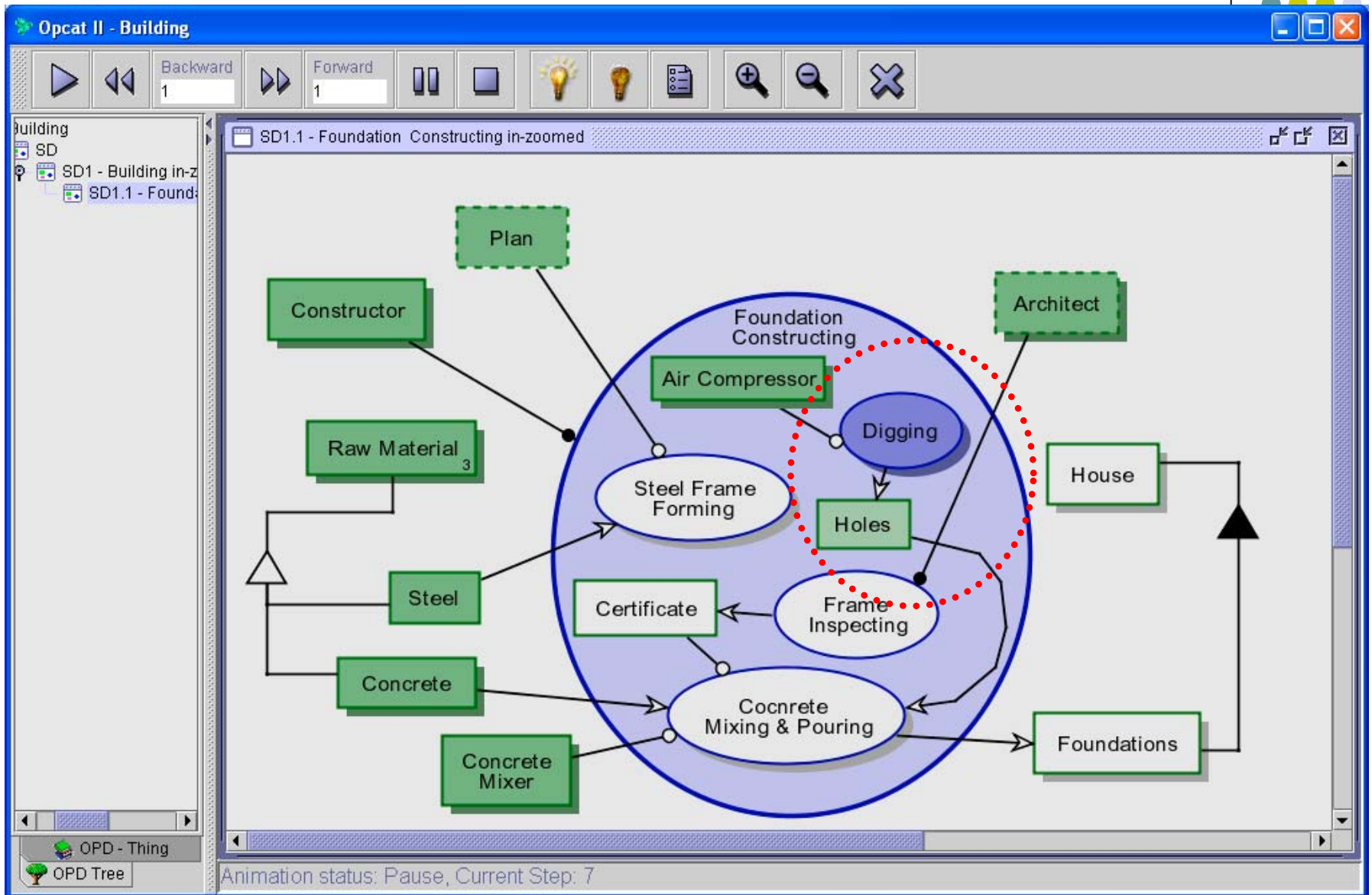


# Building in-zoomed

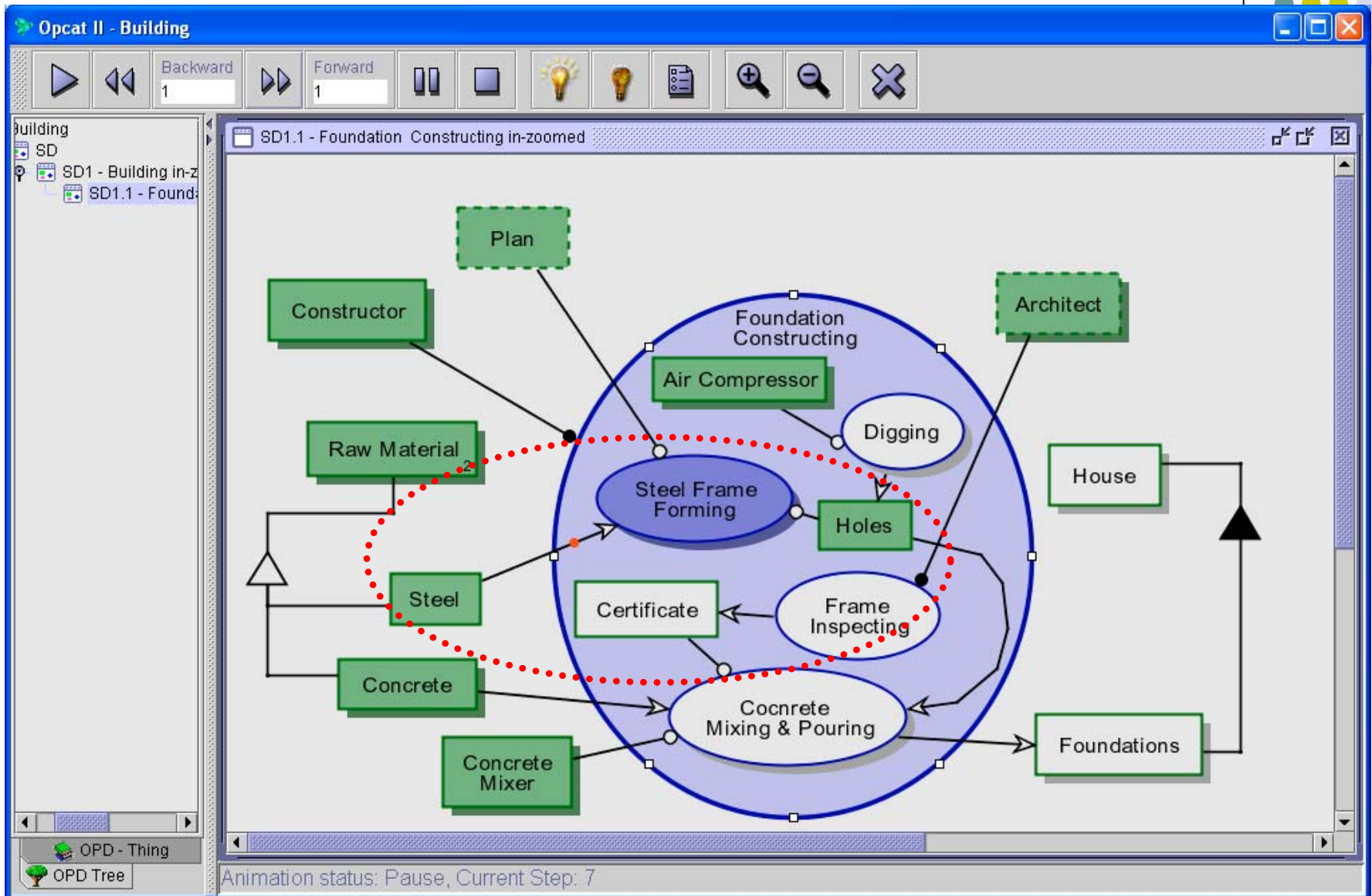


Animation status: Pause, Current Step: 7

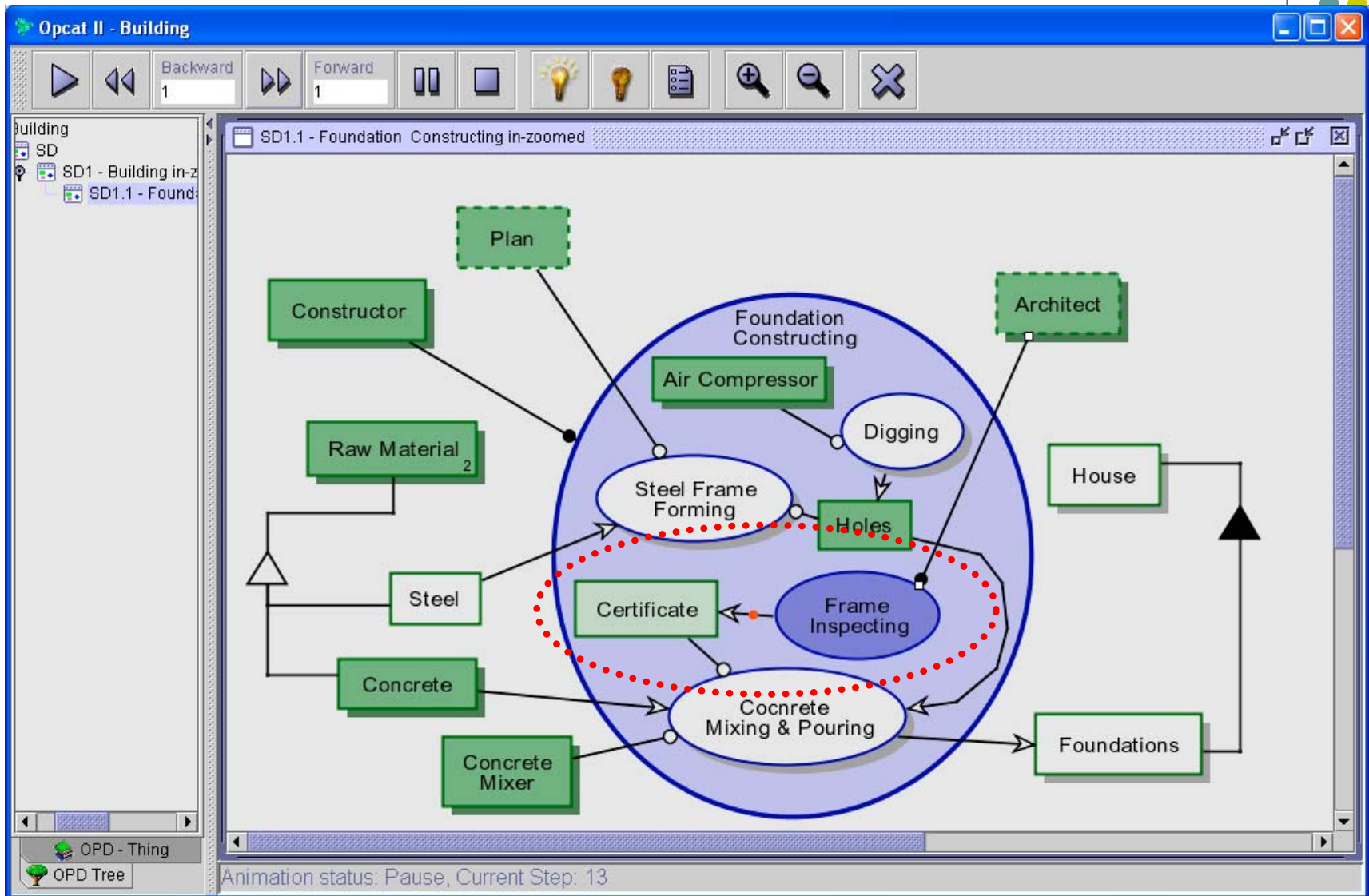
# Foundation Constructing in-zoomed



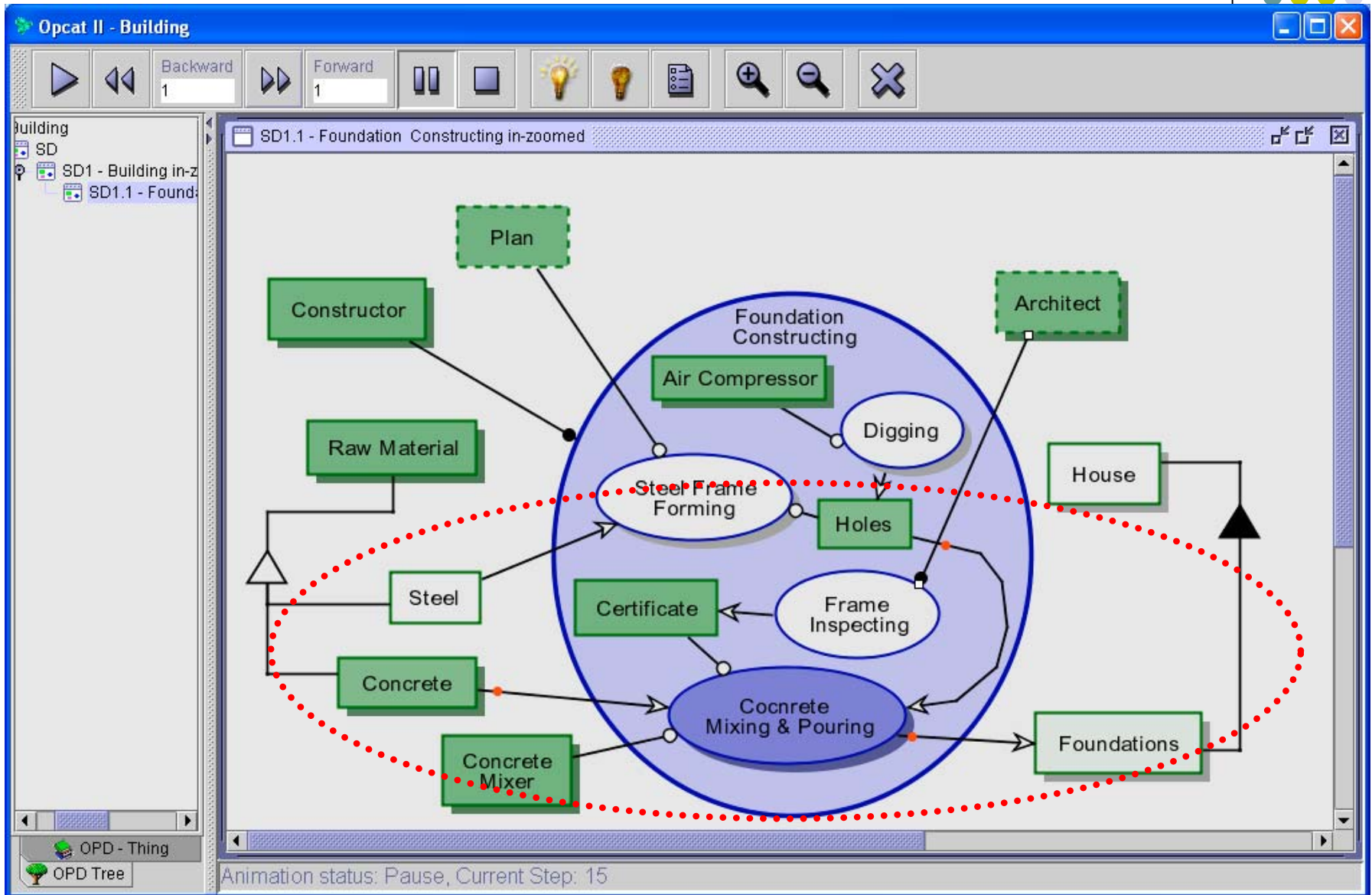
# Foundation Constructing in-zoomed



# Certificate is being created



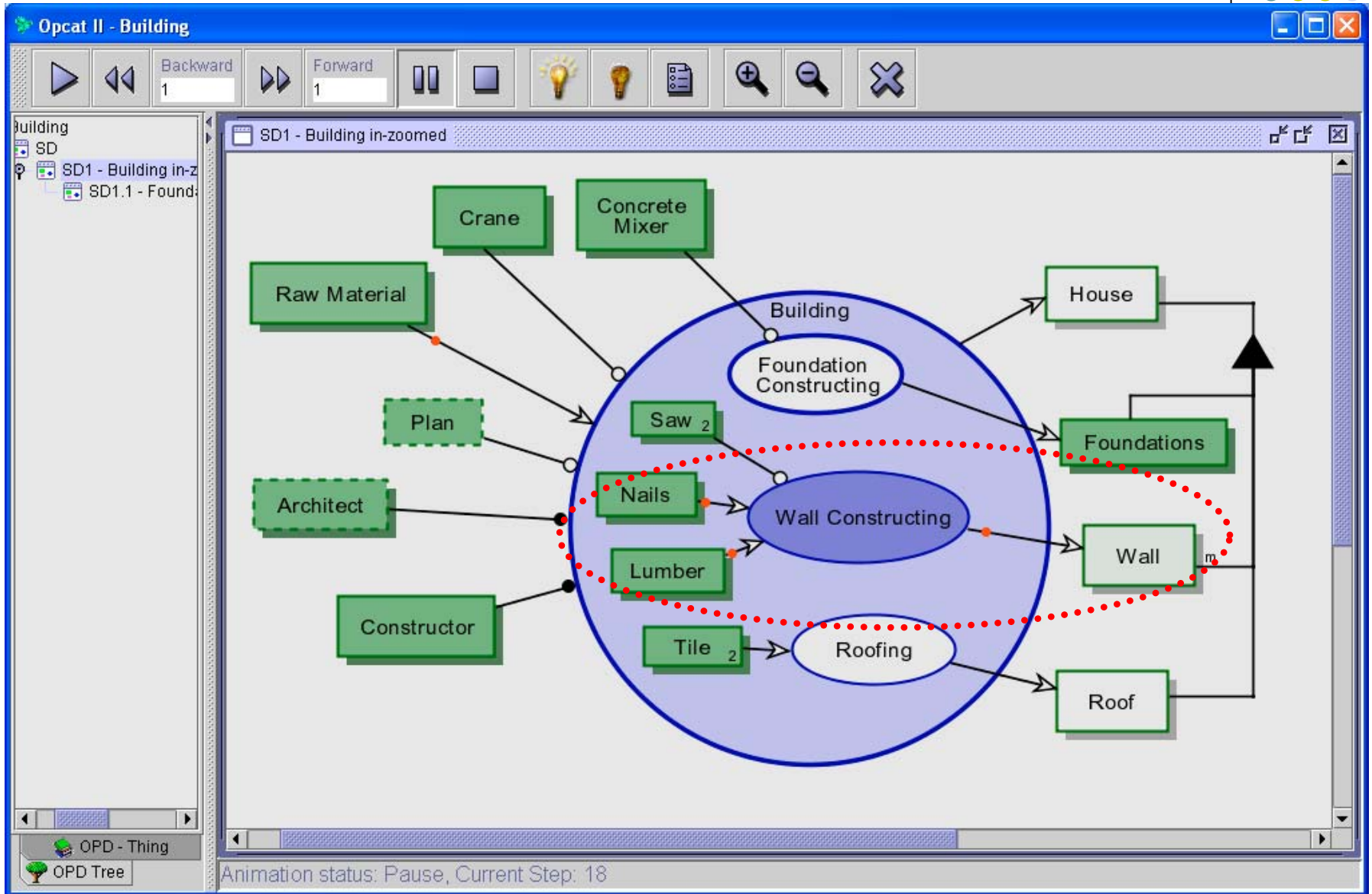
# Foundations are being created, Holes and Concrete consumed



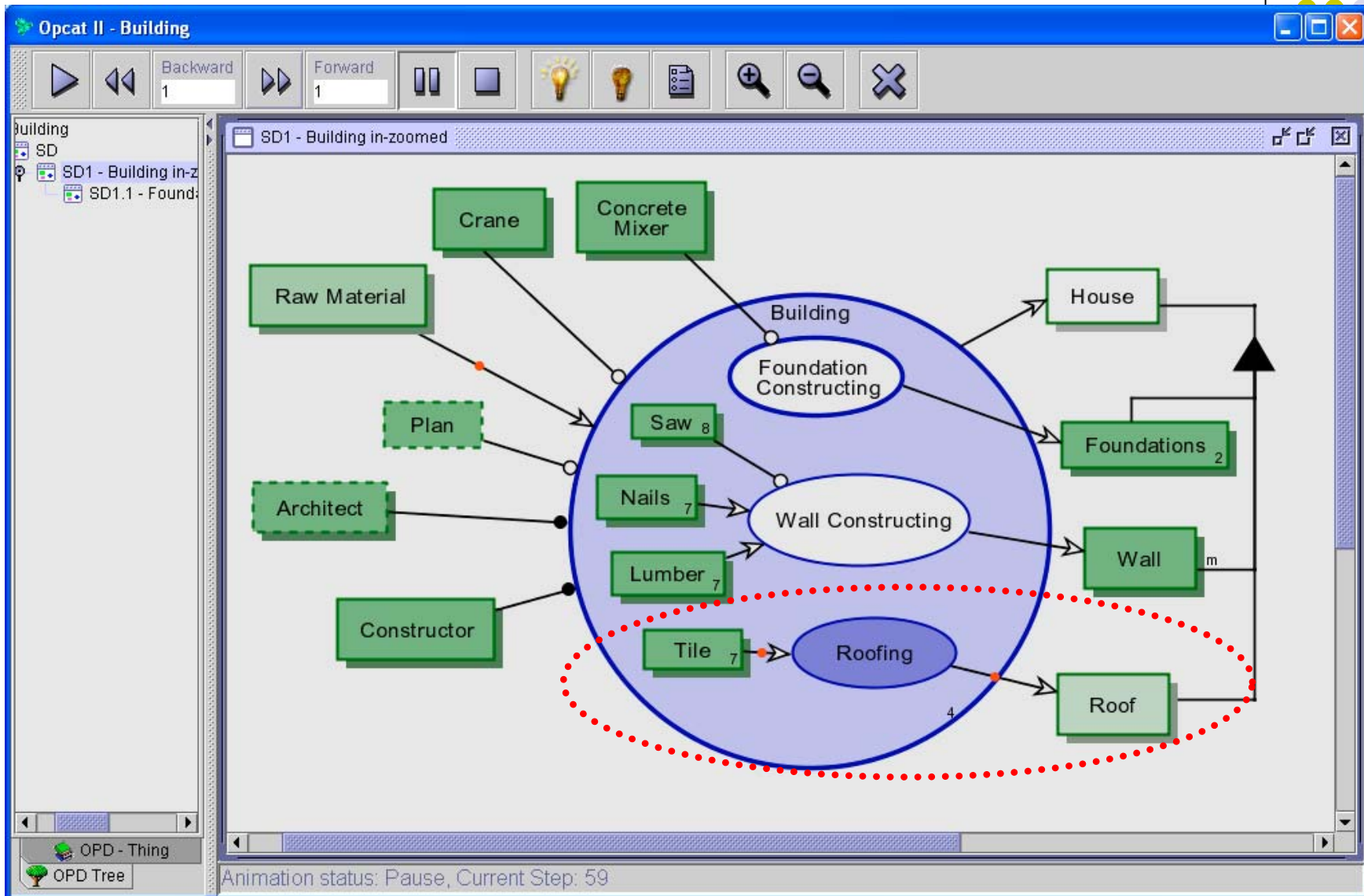
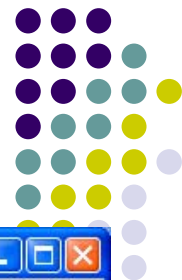




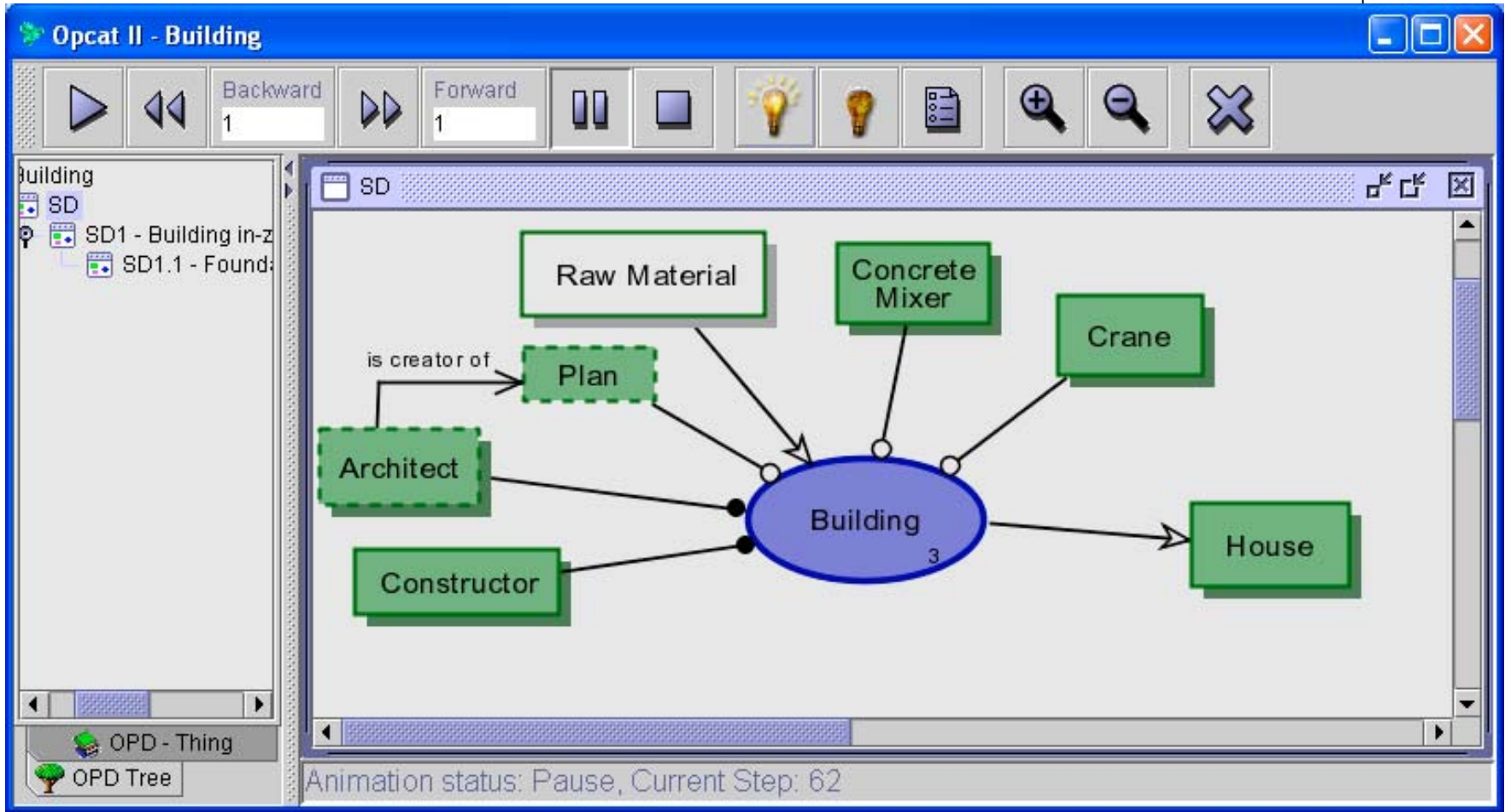
# Walls are being constructed, Nails and Lumber consumed



# Roof is being constructed, Tiles consumed



# House ready, Raw Material gone



# Downloading OPCAT 2: [www.ObjectProcess.org](http://www.ObjectProcess.org)



OPM - Official Web Site ... - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address <http://dori.technion.ac.il/opm/downloads/default.asp?up=indir&sec=8> Go

Google Delugach Search Web Search Site News New! PageRank Page Info Up Highlight Delugach

Links Customize Links Windows DEBKAfile

# OPM

Object Process Methodology

Wed 14 May, 2003 08:34:50 User: Dov Dori

[Events] [Publications] [Forums] [FAQs] [Methodology] [Exercises] [Members] [Links] [Downloads] [Search] [Contact Us]

## OPM - Downloads

If you wish to join our member list to get news and updates please press here.

### OPCAT 2.09 - Object-Process CASE Tool Version 2.09

**Download Now** ↗

**Version:** OPCAT 2.09  
**Released:** 16/02/2003  
**File Size:** (1.898 MBytes)

**Description:**

This version requires [Java SDK 1.4](#).

This version contains the animation module, as well as a UML generator, a document generator, and an OPL to OPD convertor.

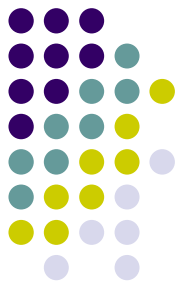
### OPCAT 2.07 - Object-Process CASE Tool Version 2.07

**Download Now** ↗

**Version:** OPCAT 2.07  
**Released:** 23/01/2003  
**File Size:**

**Description:** This version contains the animation module and runs in J2SE SDK 1.4 and 1.3.  
The developers are not responsible for any direct or indirect damage as a result of using the software.

# Translating to UML Diagrams



**Use Case Diagram**

**OPM to UML**

**Use Case Diagram**

Root Level Only

All OPD Levels

Until Level

None

**Sequence Diagrams**

Yes

No

Order Paying And Sup

**Statechart Diagrams**

Yes

No

Inventory Empty

Product

Order

**Class Diagram**

Yes

No

**Deployment Diagram**

Yes

No

**Activity Diagrams**

None

From Top Level down  levels

By Processes

Order Paying And Supplying

Product Handling

All levels down

levels down

Generate Cancel

**Class Diagram**

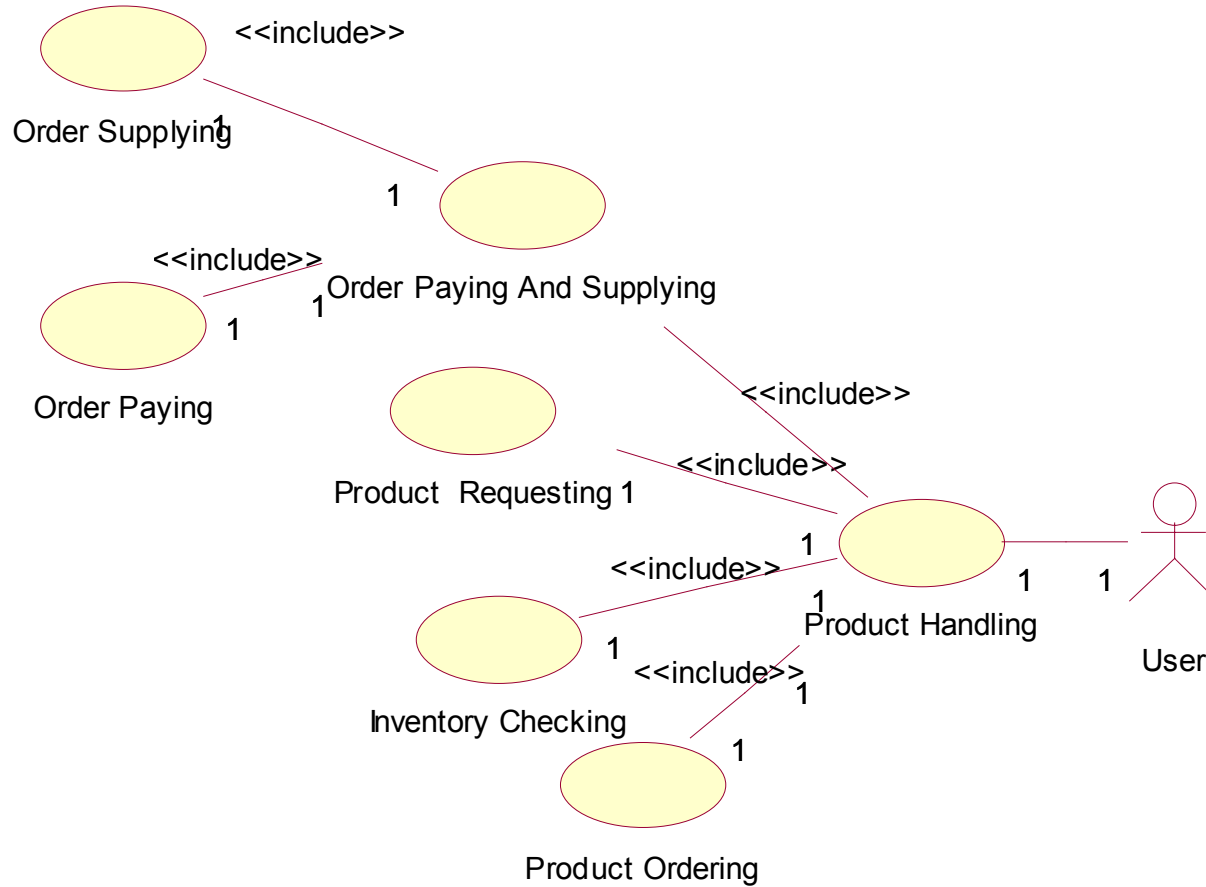
**Deployment Diagram**

**Activity Diagram**

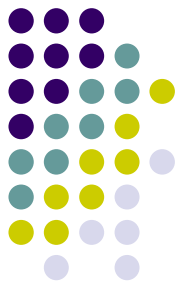
**Sequence Diagram**

**Statechart**

# UML Use Case Diagram



# Example: Defining a JAVA Condition



**Edit Operation**

Translation to: java

Operation Number: 0

Condition:

\*\*\*Complex Condition\*\*\*

Complex condition

Action:

insertAtLocation

path

file: \$\$subjectAggregationFatherName\$.java

location: BEFORE\_ENDIND\_TAG

Translation:

```
<Method name="getthe$$subjectThingName$$">
  public $$subjectThingName$$ getthe$$subjectThingName$$() {
    return the$$subjectThingName$$;
  }
</Method>
<Method name="setthe$$subjectThingName$$">
  public void setthe$$subjectThingName$$ ($$subjectThingName$$
new$$subjectThingName$$) {
    the$$subjectThingName$$= new$$subjectThingName$$;
  }
</Method>
```

Preview

Template's Elements:

- ObjectEnvironmentalPhysicalSe
- ObjectInheritanceSentence
- ObjectStateSentence
- ObjectExhibitionSentenceSet
- ObjectAggregationSentenceSe
- ObjectUniDirectionalRelationSe

Ok

# Example of a generated JAVA complex process



```
// File OrderPayingAndSupplying.java representing the complex process OrderPayingAndSupplying

package OrderSystem;

import opmTypes.*;

public class OrderPayingAndSupplying extends opmProcess {
    public OrderPayingAndSupplying () {
    }
    public boolean preconditionHolds (Boolean theInventoryEmpty) {
        boolean check = true;
        if (! ( (theInventoryEmpty.booleanValue() == false) ||
                (theInventoryEmpty.booleanValue() == true)))
            check = false;
        return check;
    }
    public void run (Boolean theInventoryEmpty, Order theOrder, Product theProduct,
                    Receipt theReceipt) {
        if (preconditionHolds (theInventoryEmpty)) {
            // Effect theOrder
            // Effect theProduct
            // Yield theReceipt
            theReceipt = new Receipt();
            OrderPaying theOrderPaying = new OrderPaying();
            theOrderPaying.run(theOrder, theReceipt);
            OrderSupplying theOrderSupplying = new OrderSupplying();
            theOrderSupplying.run(theProduct, theOrder);
        }
    }
}
```



# ViSWeb – The Visual Semantic Web



## Objectives:

- Unifying human and machine knowledge representations with Object-Process Methodology (OPM)
- Enhancing the current Semantic Web technology
- Representing knowledge over the Web in a unified way that caters to human perceptions while also being machine-processable.

# The Human-Machine Language Orientation Dilemma

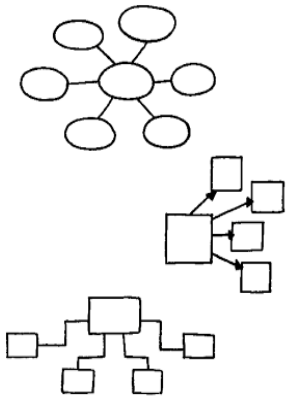


## Semantic Web major assumption:

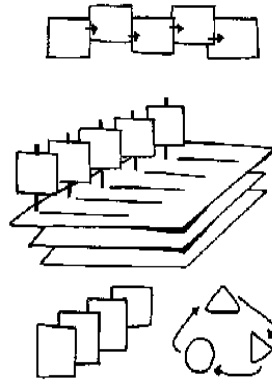
- Humans and machines must each use a different format of knowledge representation.
- **OWL Introduction:** “*...computational agents require machine-readable descriptions of the content and capabilities of web accessible resources. These descriptions must be in addition to the human-readable versions of that information.*” – NOT TRUE!



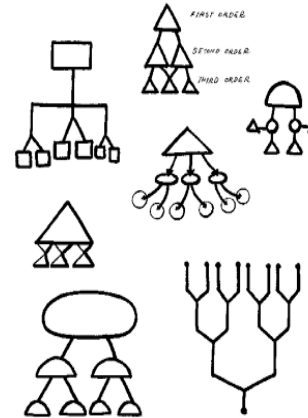
# Graphic Knowledge Representations: Concept Maps



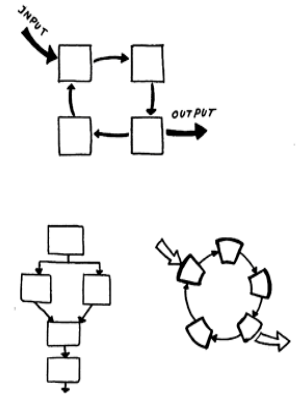
Spider



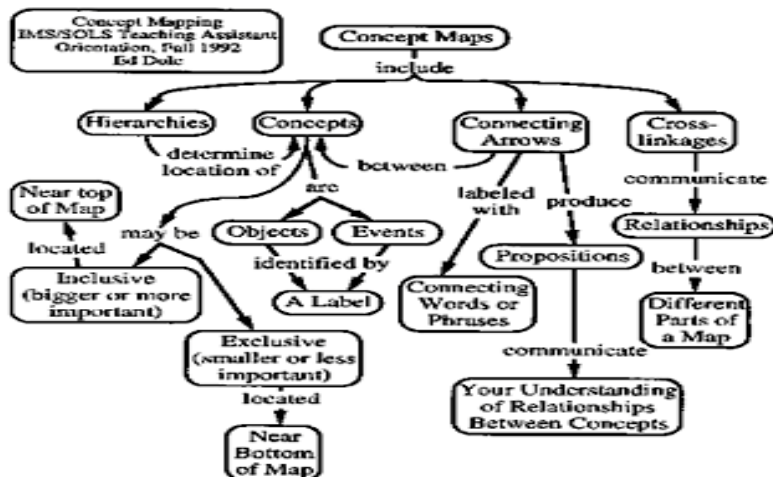
Flowchart



Hierarchy

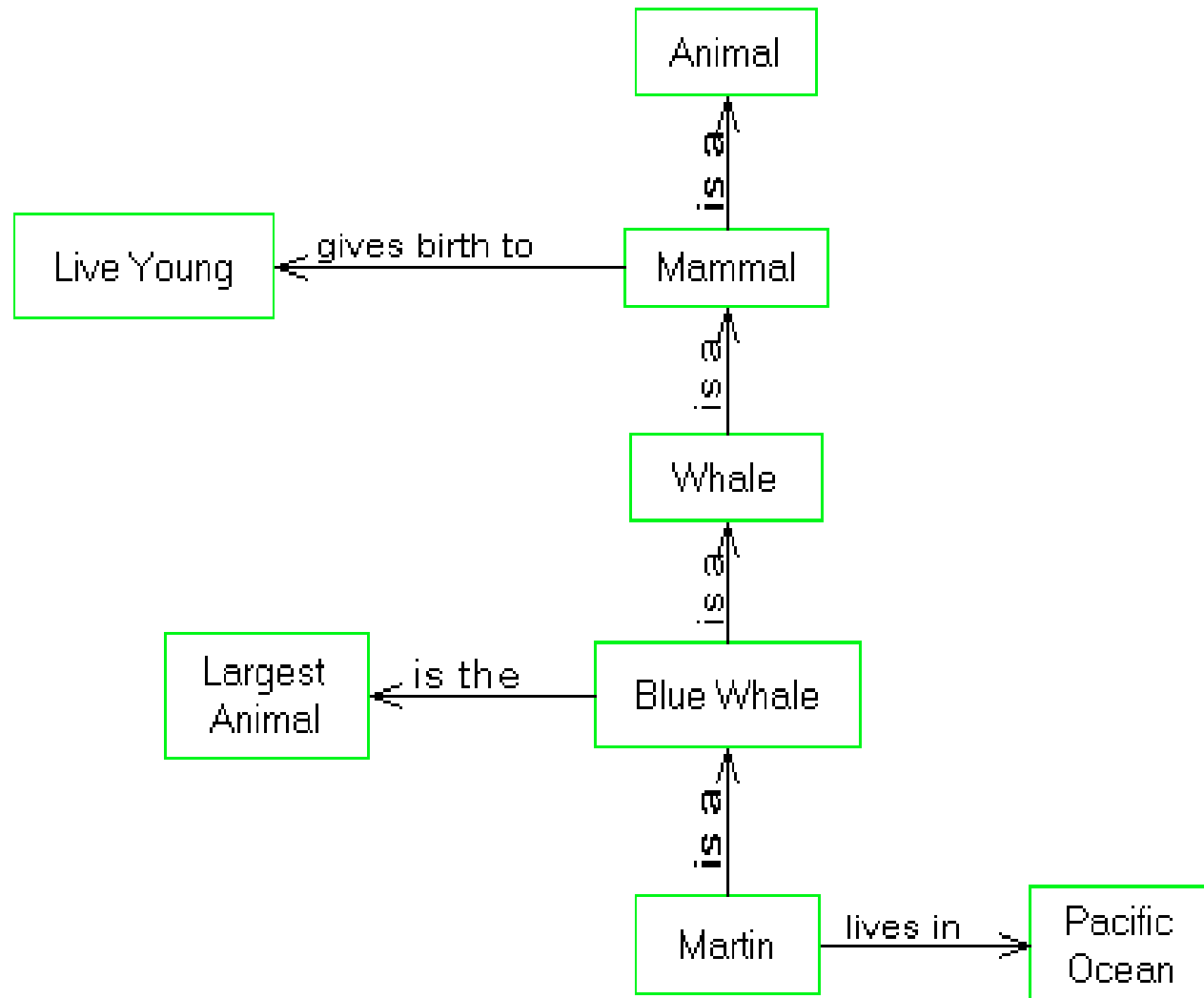


System

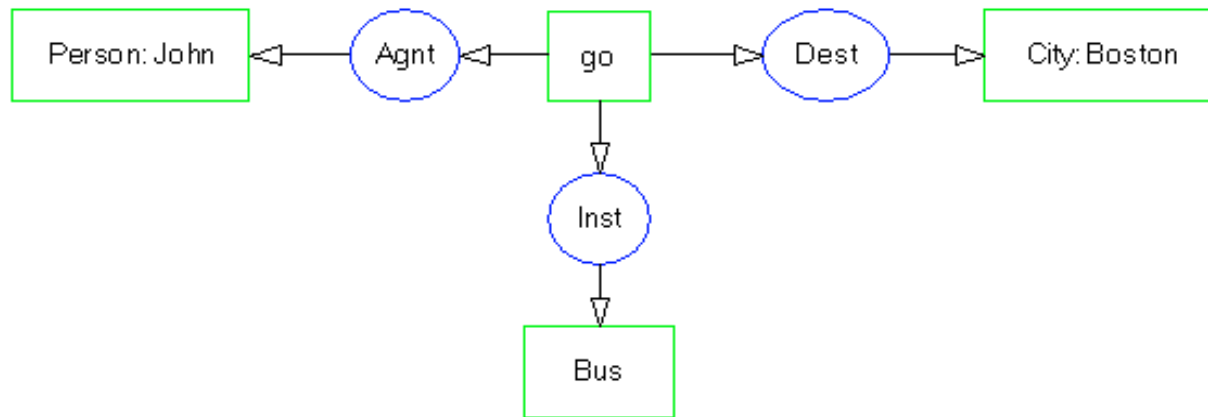


Concept map of concept map (Adapted from <http://classes.aces.uiuc.edu/ACES100/Mind/CMap.html>)

# Graphic Knowledge Representations: Semantic Network



# Graphic Knowledge Representations: Conceptual Graphs

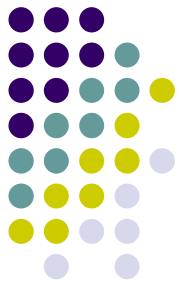


CG Display Form (DF) for "*John is going to Boston by bus.*"

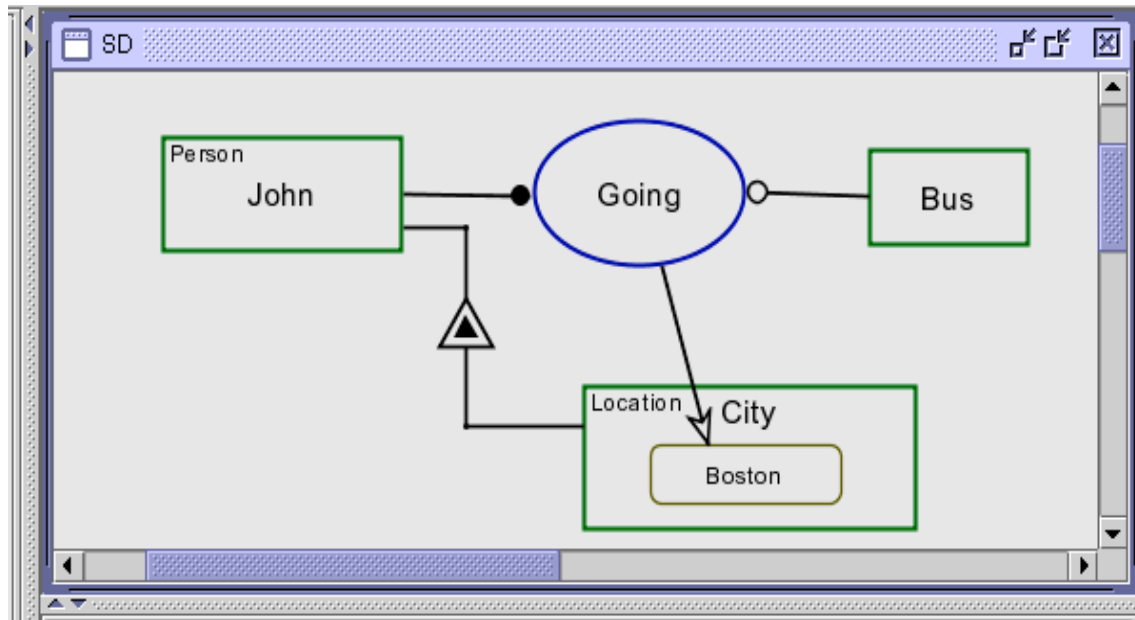
```
[Go] -  
(Agnt) -> [Person: John]  
(Dest) -> [City: Boston]  
(Inst) -> [Bus].
```

CG Linear Form (LF) for "*John is going to Boston by bus.*"

# Graphic Knowledge Representations: The OPM equivalent



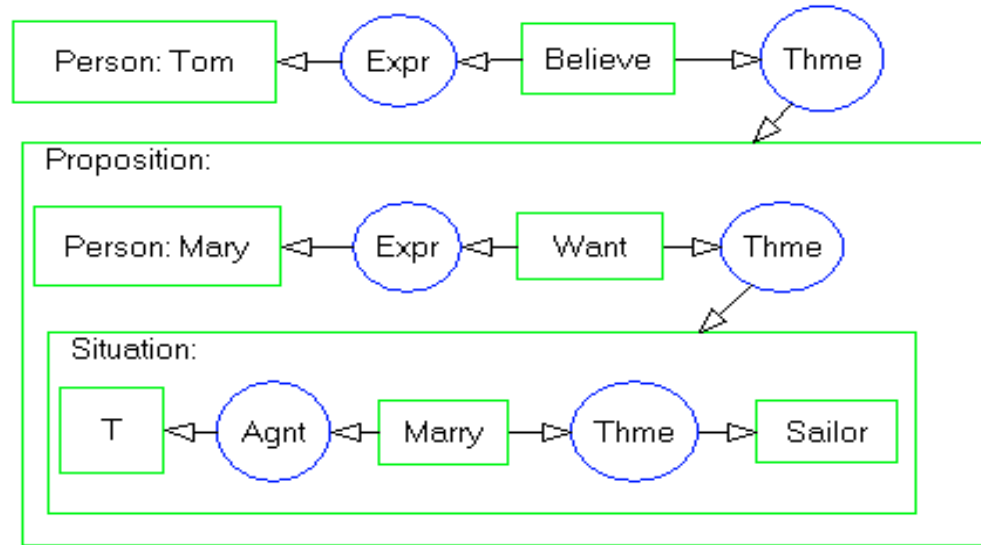
OPD:



OPL:

The **Person John** exhibits the **Location City**.  
**City** is **Boston**.  
**John** handles **Going**.  
**Going** requires **Bus**.  
**Going** changes **City** to **Boston**.

# Graphic Knowledge Representations: Conceptual Graphs: Context



CG Display Form (DF) for "*Tom believes that Mary wants to marry a sailor*".

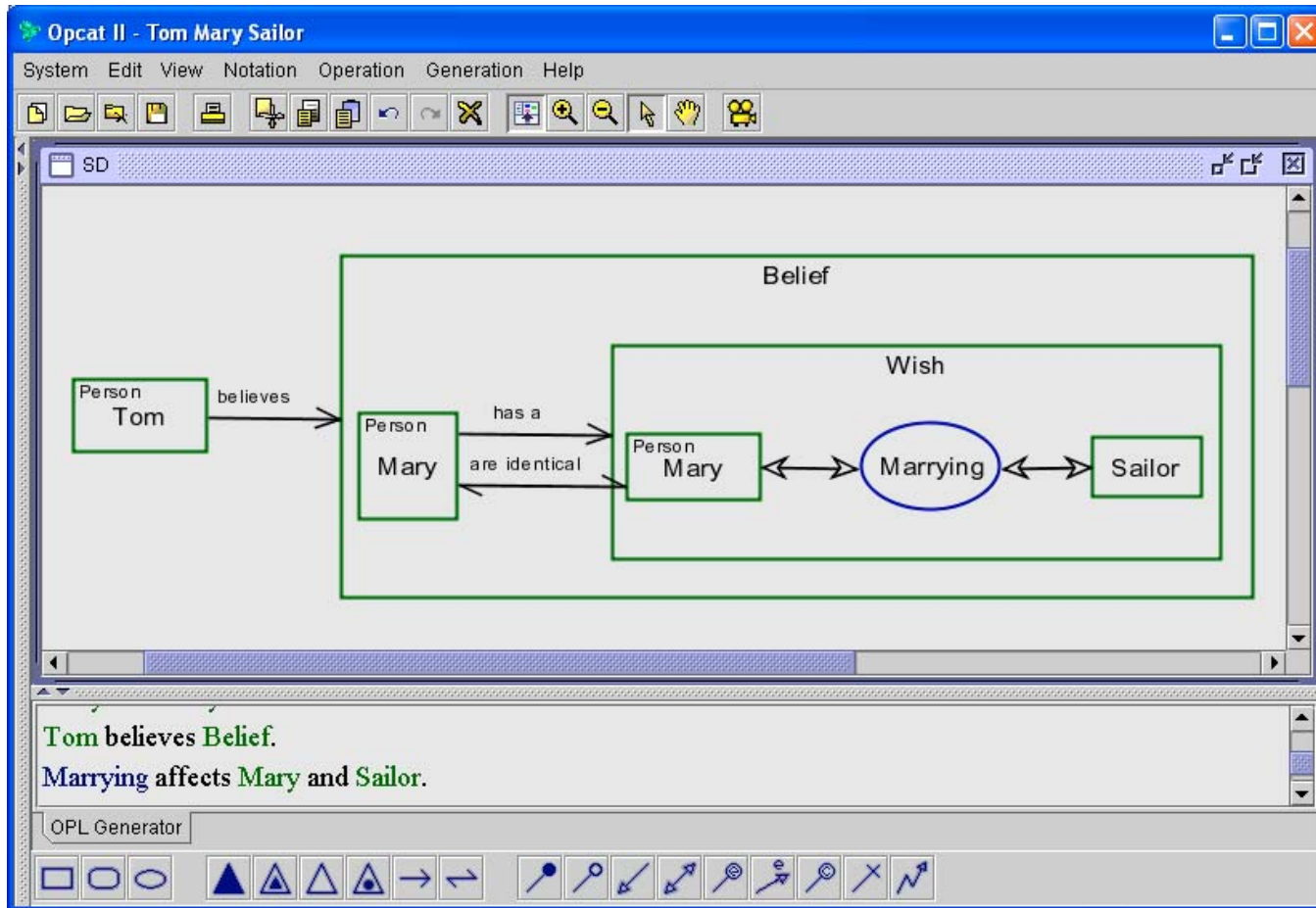
```
[Person: Tom] <- (Expr) -> [Believe] -> (Thme) -
[Proposition: [Person: Mary *x] <- (Expr) <- [Want] -> (Thme) -
[Situation: [?x] <- (Agnt) <- [Marry] -> (Thme) -> [Sailor]]].
```

CG Linear Form (LF) for "*Tom believes that Mary wants to marry a sailor*".

# Graphic Knowledge Representations: The OPM equivalent

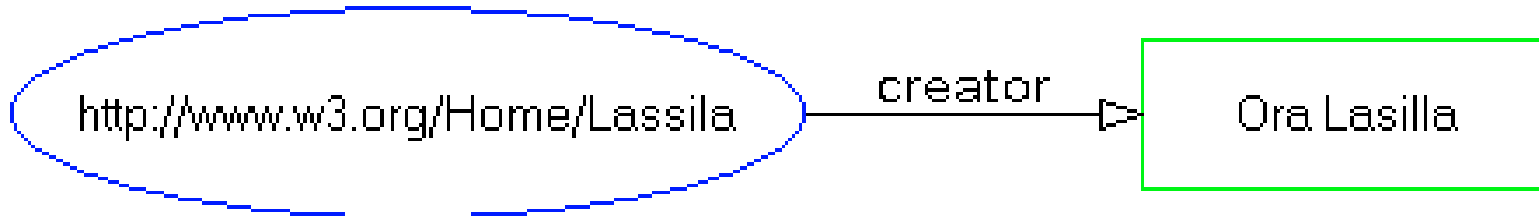


OPD:





# Graphic Knowledge Representations: RDF



*"http://www.w3.org/Home/Lassila has creator Ora Lassila",*

and in general

*"<subject> HAS <predicate> <object>".*

# RDF – more realistic with

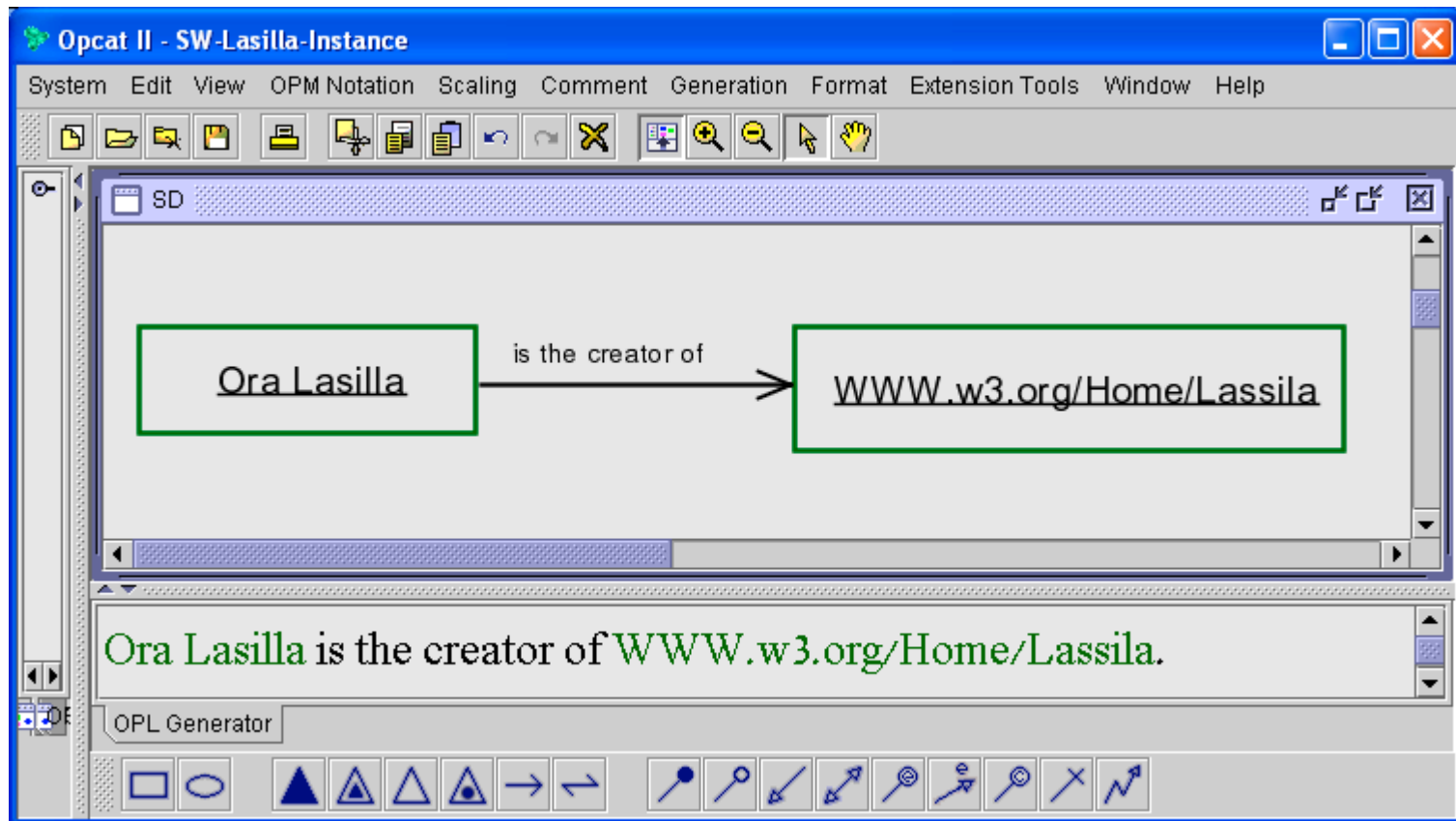
RDF/XML Validation Service

<http://www.w3.org/RDF/Validator/>



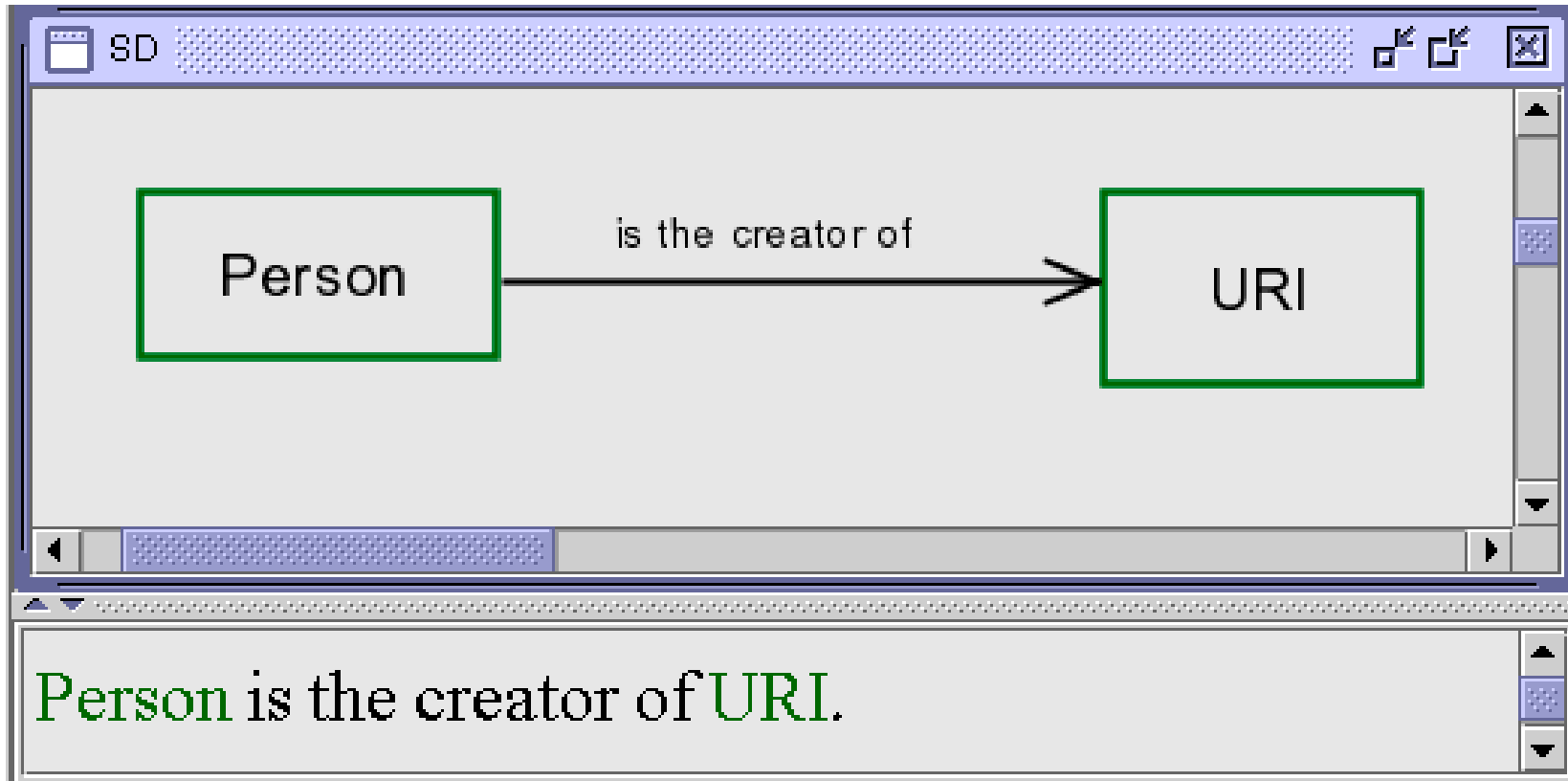
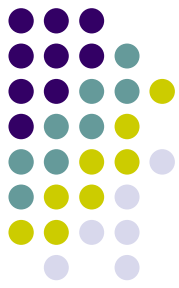
```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://description.org/schema/">
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>Ora Lassila</s:Creator>
  </rdf:Description>
</rdf:RDF>
```

# ViSWeb: An OPM-Based Visual Semantic Web Spec Alternative

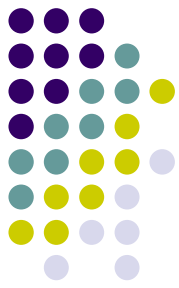


Ora Lasilla is the creator of [WWW.w3.org/Home/Lassila](http://WWW.w3.org/Home/Lassila).

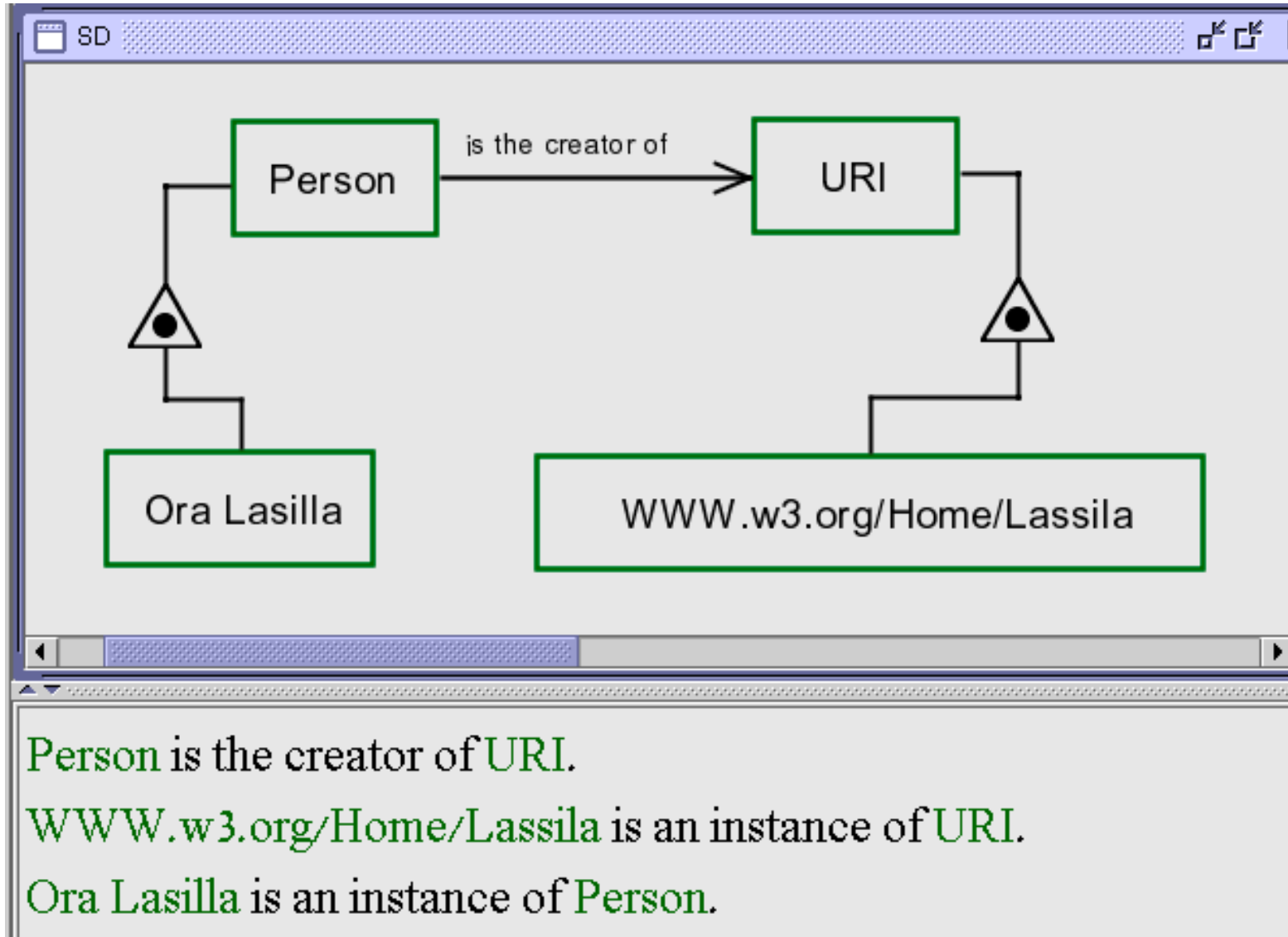
# The ViSWeb Schema: Adding Class Information

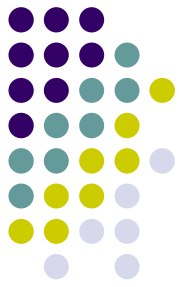


**Person is the creator of URI.**

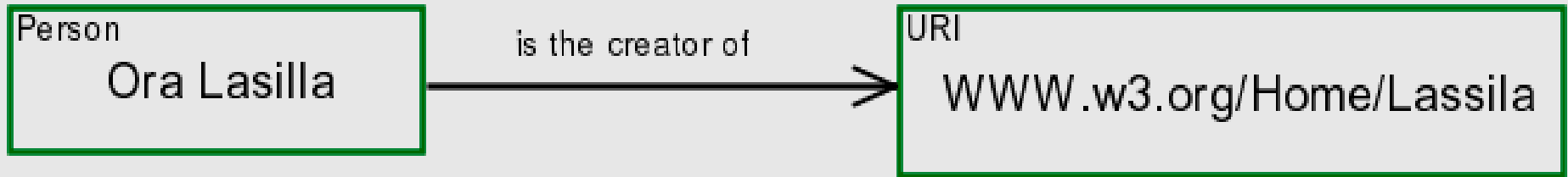


# Instantiating the ViSWeb Schema



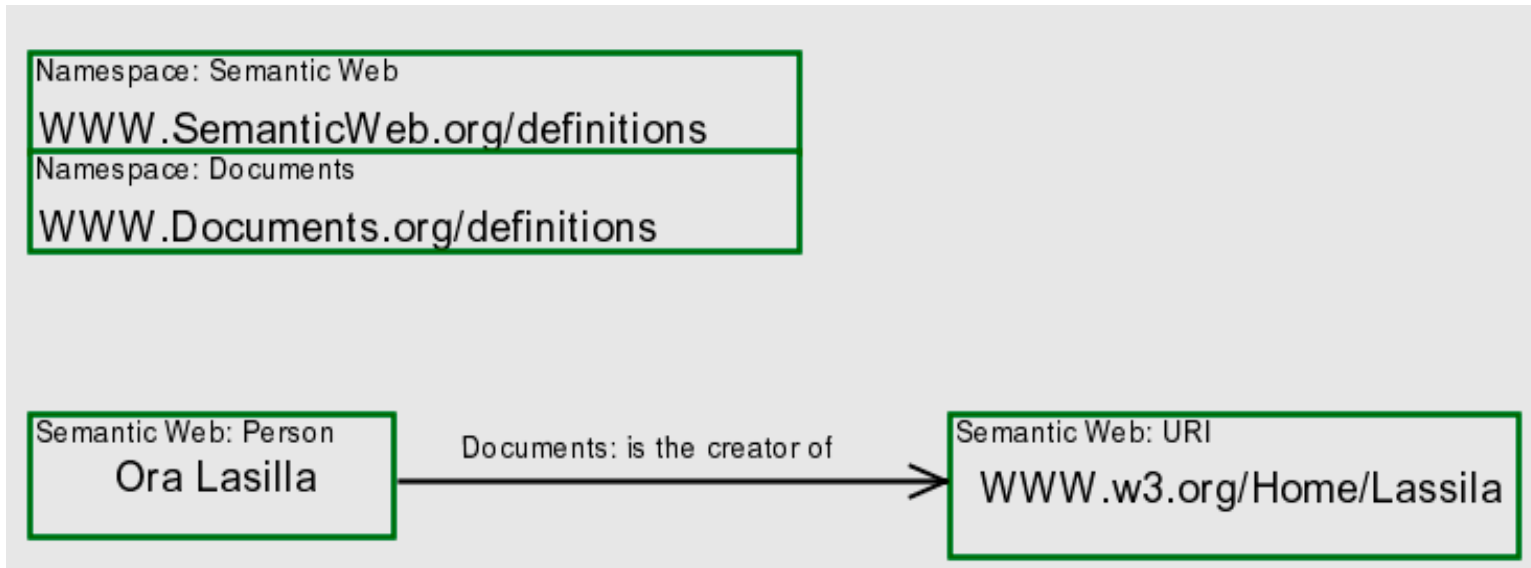


# A more compact version:



The **Person Ora Lasilla** is the creator of the **URI** [WWW.w3.org/Home/Lassila](http://WWW.w3.org/Home/Lassila).

# OPM Namespace Specification



The namespace **Semantic Web** is at URL [WWW.SemanticWeb.org/definitions](http://WWW.SemanticWeb.org/definitions).

The namespace **Documents** is at URL [WWW.Documents.org/definitions](http://WWW.Documents.org/definitions).

*(Namespace declaration sentences)*

The namespace **Semantic Web** defines the class **Person**.

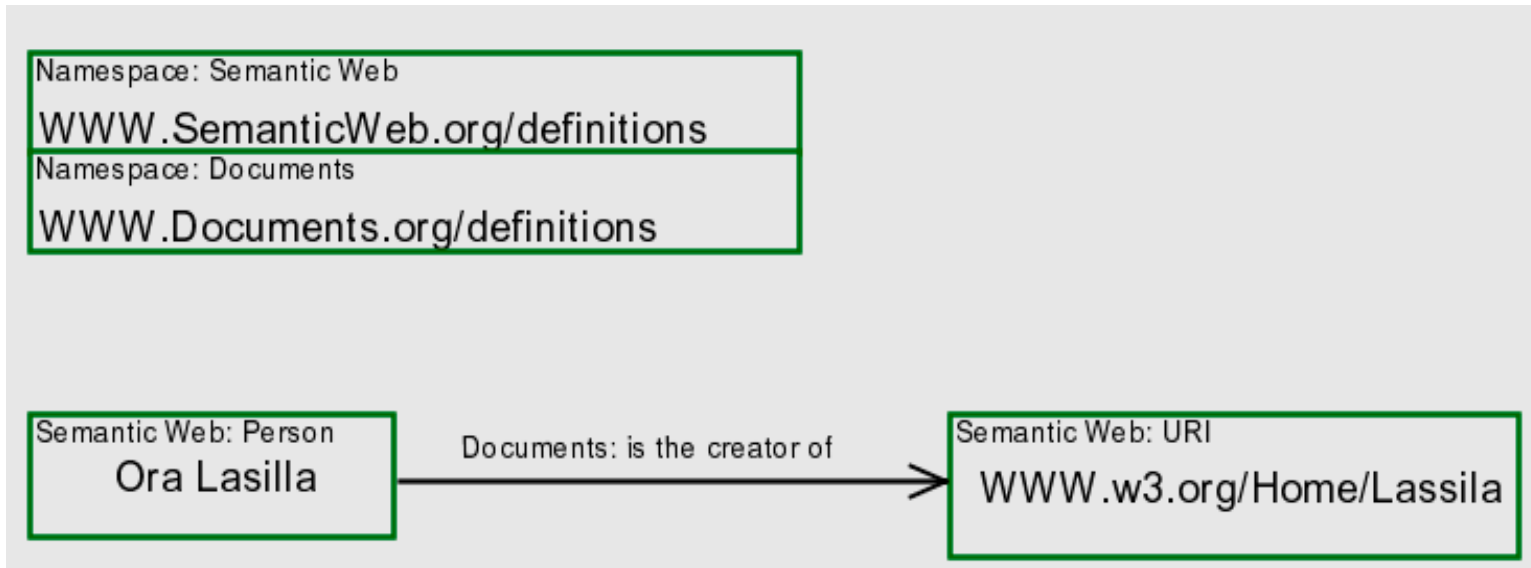
The namespace **Semantic Web** defines the class **URL**.

*(Class definition sentences)*

The namespace **Documents** defines the relation '**is the creator of**'.

*(Relation definition sentence)*

# Default Namespace Specification



The default namespace **Semantic Web** is at [WWW.SemanticWeb.org/definitions](http://WWW.SemanticWeb.org/definitions).

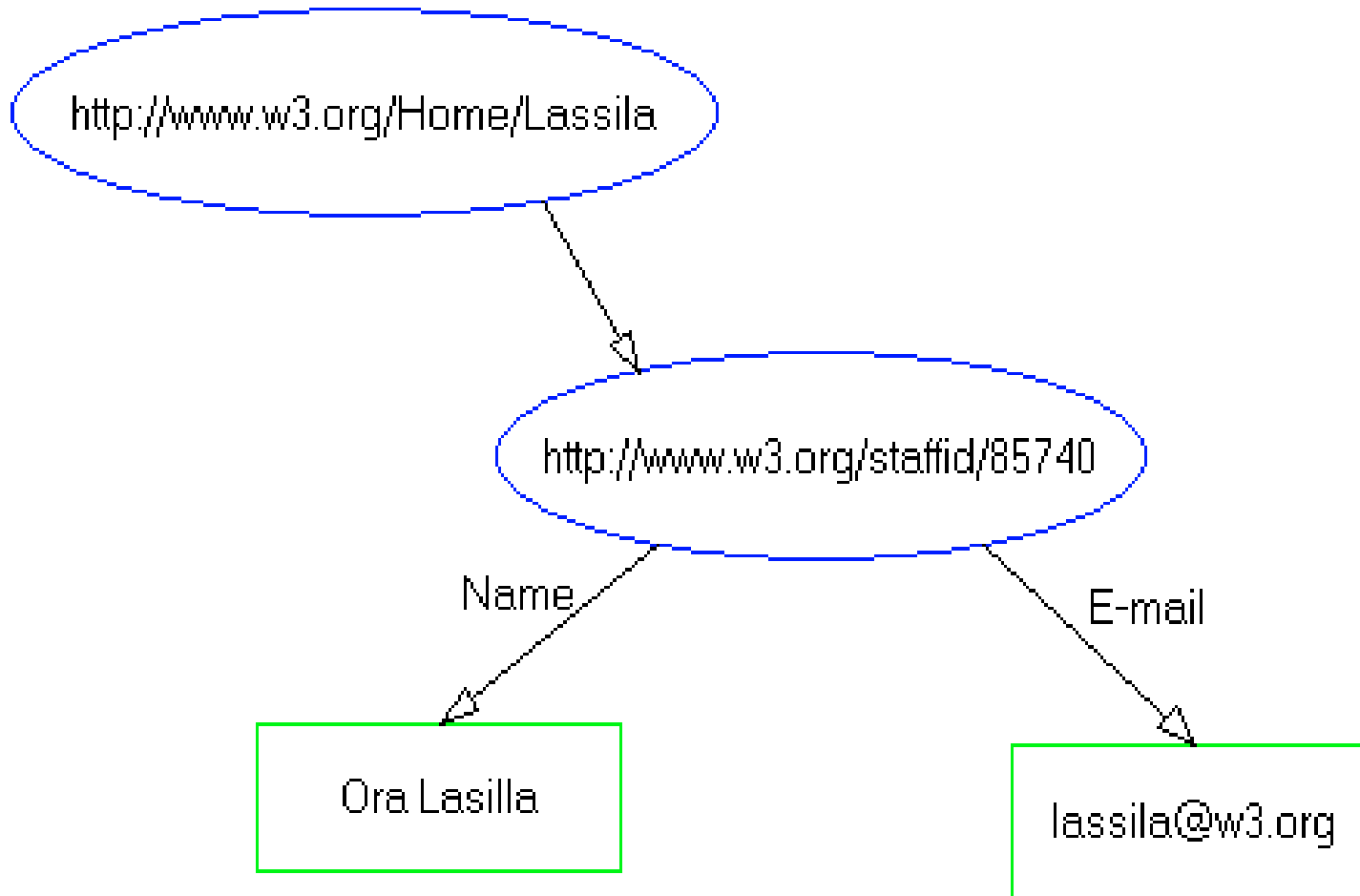
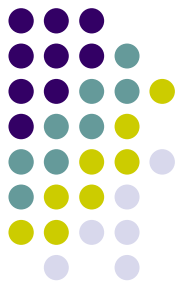
The namespace **Documents** is at [WWW.Documents.org/definitions](http://WWW.Documents.org/definitions).

The namespace **Documents** defines the relation '**is the creator of**'.

The **Person Ora Lasilla** is the creator of the **URI** [WWW.w3.org/Home/Lassila](http://WWW.w3.org/Home/Lassila).

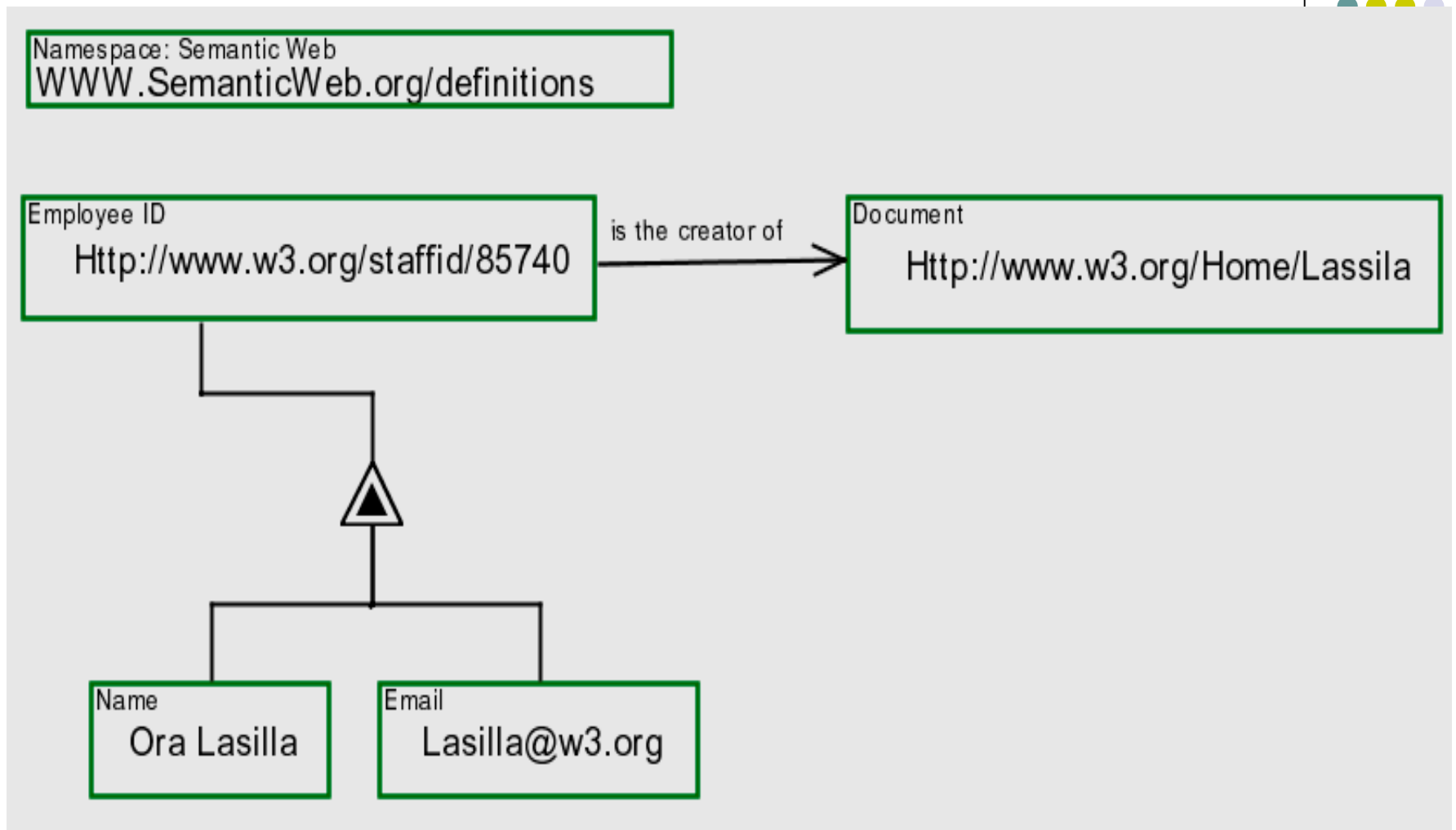


# Adding Attributes



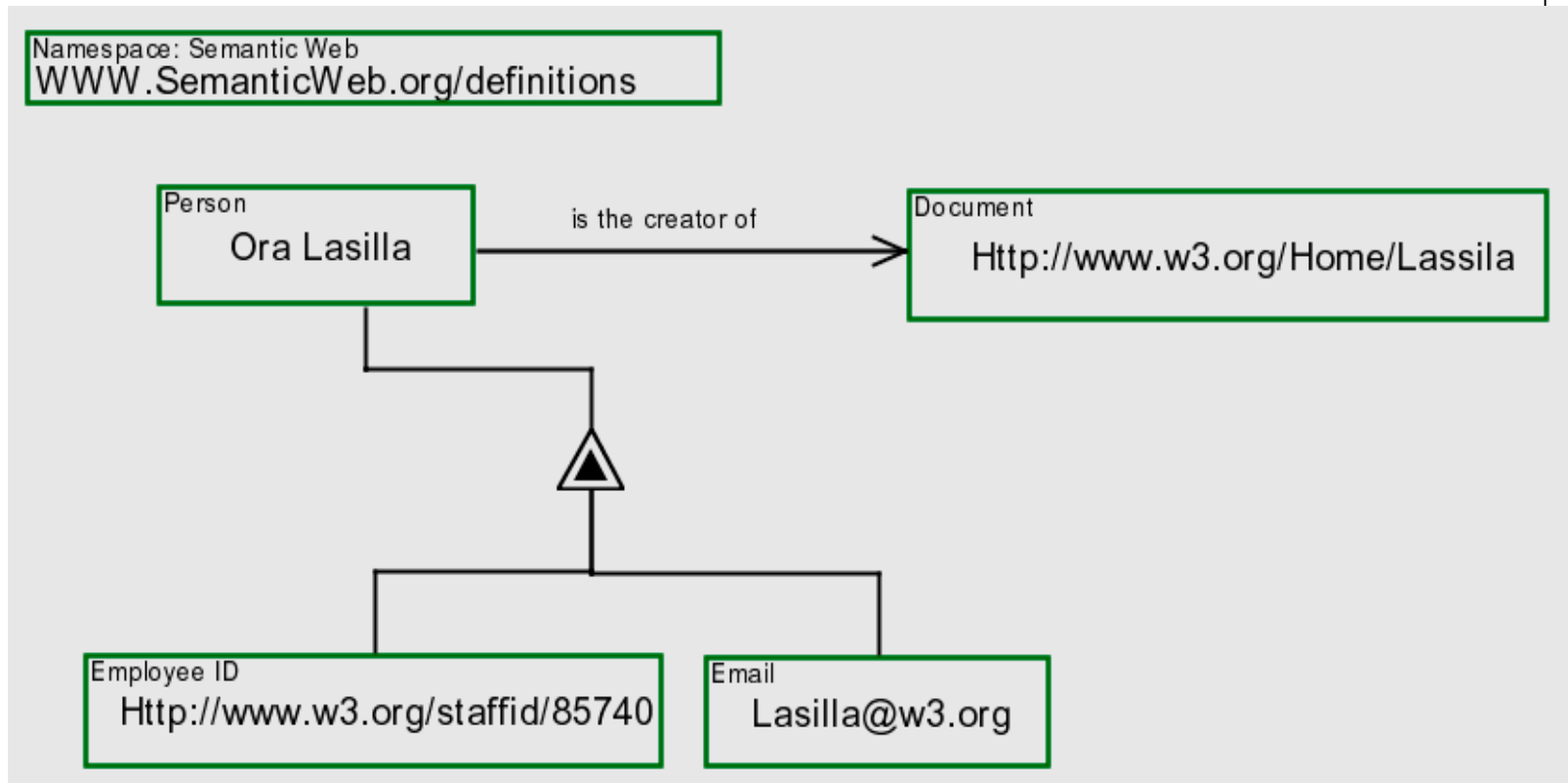
An identified property with structured value

# Adding Attributes



The corresponding OPD

# A better representation

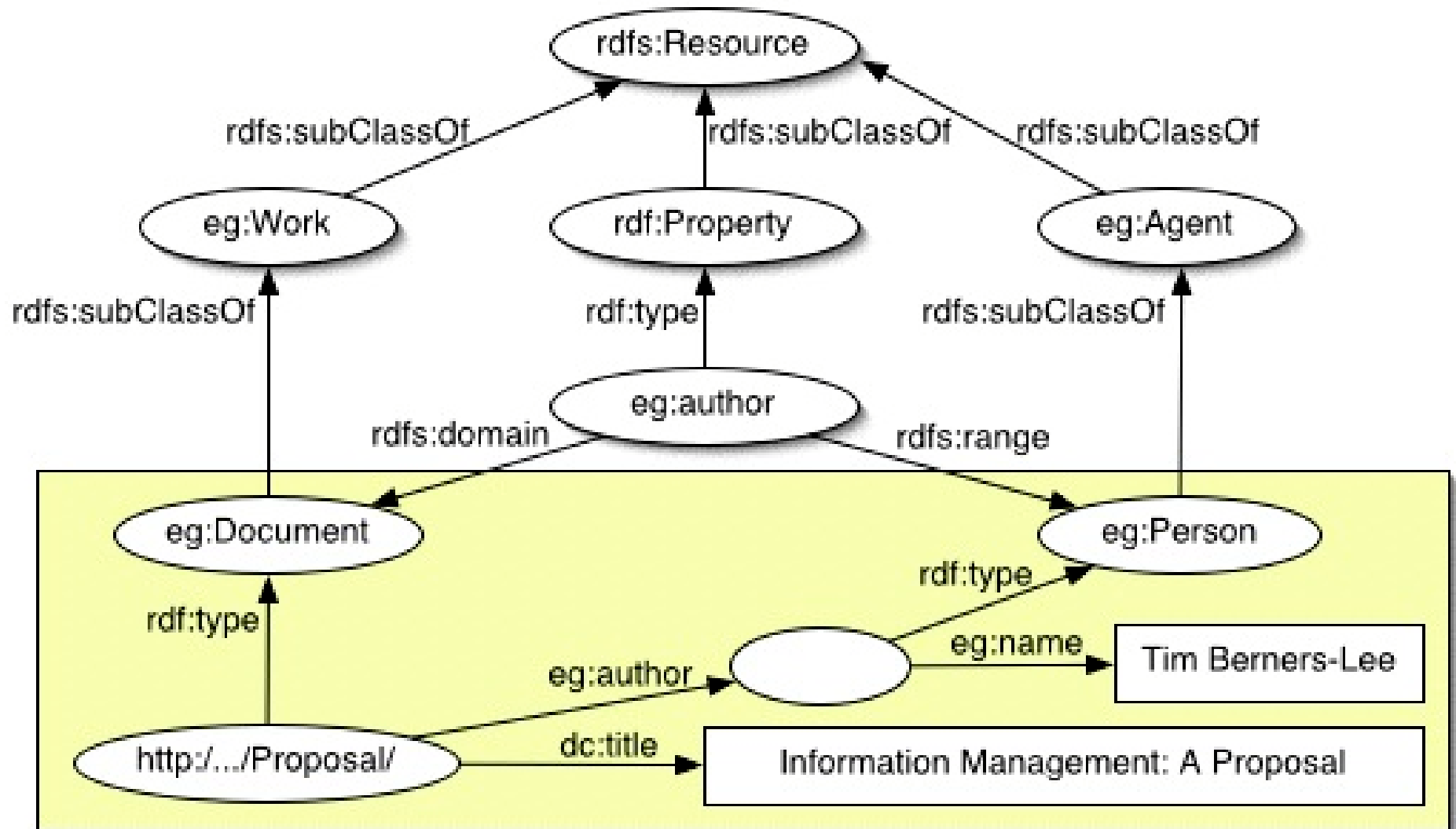


The default namespace **Semantic Web** is at [WWW.SemanticWeb.org/definitions](http://WWW.SemanticWeb.org/definitions).

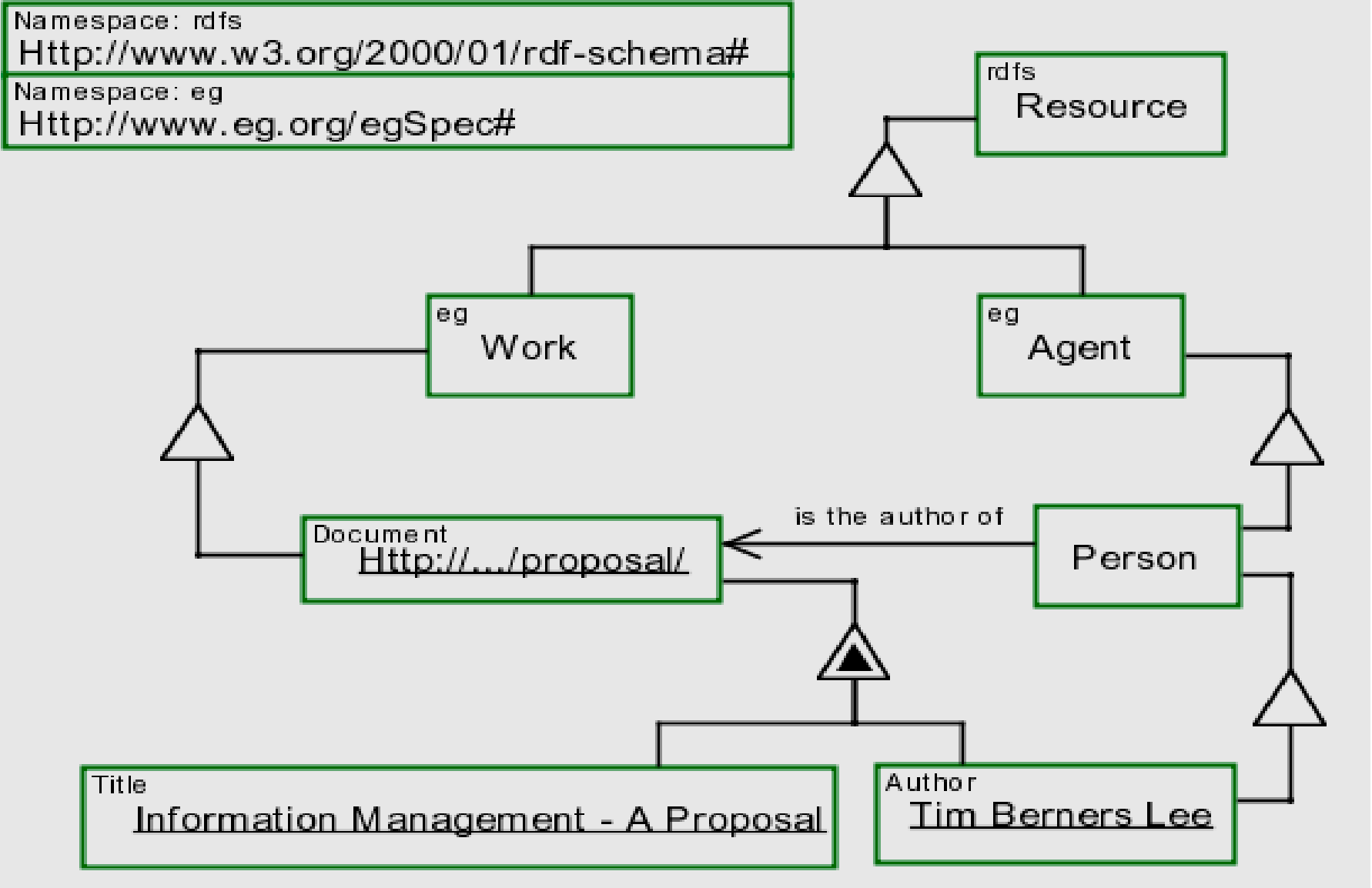
The **Person Ora Lasilla** is the creator of the **Document** [WWW.w3.org/Home/Lassila](http://WWW.w3.org/Home/Lassila).

The **Person Ora Lasilla** exhibits the **Employee ID** [WWW.w3.org/staffid/85740](http://WWW.w3.org/staffid/85740) and the **Email** [Lasilla@w3.org](mailto:Lasilla@w3.org).

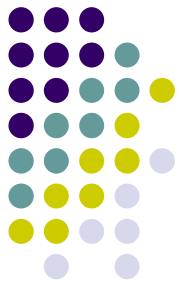
# A final RDF and OPM example



# A final RDF and OPM example



# A final RDF and OPM example



The default namespace **rdfs** is at [WWW.w3.org/2000/01/rdf-schema#](http://WWW.w3.org/2000/01/rdf-schema#).

The namespace **eg** is at [WWW.eg.org/egSpecs#](http://WWW.eg.org/egSpecs#).

The namespace **eg** defines **Work** and **Agent**.  
**Work** and **Agent** are **Resources**.

**Document** is a **Work**.

**Person** is an **Agent**.

**Author** is a **Person**.

**Document** exhibits **Author** and **Title**.

The **Document** [Http://.../Proposal](http://.../Proposal) exhibits the **Author Tim Berners Lee** and the **Title Information Management – A Proposal**.



# Advantages of the VisWeb Paradigm

- **Graphic-text knowledge representation:**
  - The powerful graphic-text bimodal representation of OPM is extended to the Visual Semantic Web paradigm.
  - Rather than mentally parsing cryptic XML scripts, knowledge is presented to the user in a subset of natural language as well as diagrammatically.
  - Puts to work the "two sides of the human brain," the visual and the lingual.

# Advantages of the VisWeb Paradigm



- **Visual navigability:**

- diagrammatic display enables users to surf and navigate the Web in a visual way in search for knowledge.
- Objects, processes, classes and links can be hyperlinked to pertinent Web sites, which themselves may contain VSW or any other multimedia knowledge representations.



# Advantages of the VisWeb Paradigm



- **Semantic sentence interpretation:**

- the basis of the RDF framework is syntactic rather than semantic:
- It draws on the concepts of *subject*, *predicate* and *object*, which are parts of speech used to analyze natural language sentences from a syntactic viewpoint.
- The same semantics can be expressed by inverse syntactic expressions.
- VisWeb is based on a sound ontology of *objects* with *states* and *processes*:

# Advantages of the VisWeb Paradigm



- **Semantic sentence interpretation (cont.):**
  - VisWeb is based on a sound ontology of *objects* with *states* and *processes*:
  - Objects are things that exist, (possibly at some state)
  - Processes are things that happen to objects and transform them:
    - create or destroy them, or
    - change their state
  - Based on this ontology, sentences can be interpreted semantically rather than syntactically.

# Advantages of the VisWeb Paradigm



- **Specification of system dynamics:**
  - Current work on the Semantic Web places emphasis on declaratively specifying structural knowledge, which relates to the static aspect of systems.
  - A major part of the knowledge about a system is functional (what is its purpose) and dynamic (how it operates).
  - Since OPM combines function, structure, and behavior in the same bimodal model, it provides a sound infrastructure for representing system dynamics and function in the ViSWeb model.

# Advantages of the VisWeb Paradigm



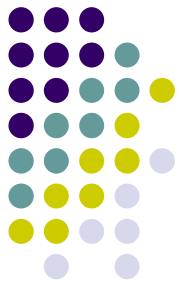
- **Complexity management:**
  - A major problem in real-life systems is their complexity due to the sheer amount of knowledge details.
  - OPM has built in abstraction-refinement mechanisms, including in-zooming and out-zooming, unfolding and folding, and state expression and suppression.
  - These provide for building hierarchies of knowledge representation in general and over the Web in particular, enabling navigation up and down abstraction-refinement hierarchies.

# Summary



- The Visual Semantic Web (VisWeb) paradigm proposes to unify human and machine representations of knowledge.
- The foundation for this unification is OPM.
- OPM advocates the integration of a system's structure and behavior is a single, graphic and textual model.
- Like OPM, the VisWeb model enables the representation of static and dynamic knowledge.

# Summary 2



- VisWeb uses a combination of Object-Process Language (OPL), a subset of English, and Object-Process Diagrams (OPDs), an equivalent visual formalism.
- The advantages of this approach:
  - graphic-text knowledge representation,
  - visual navigability,
  - semantic sentence interpretation,
  - specification of system dynamics, and
  - complexity management.
  - As noted in [7], "It is also important to understand that this XMI syntax is only one possible syntax for

# Summary 3



- As noted in W3C RDF Documentation
- *"It is also important to understand that this XML syntax is only one possible syntax for RDF and that alternate ways to represent the same RDF data model may emerge."*
- Indeed, the OPM-based approach to representing the Semantic Web on top of the RDF data model, is expressed
  - graphically, using OPDs, and
  - textually in Object-Process Language

# Future Work – Theoretical



- Proceed in both the theoretical and practical paths.
- The theory will focus on extending the idea behind the VisWeb paradigm to other knowledge and system representation aspects.
- VisWeb should be able to also handle procedural, dynamic behavioral aspects, as well as functional ones.



# Future Work – Practical



- The practical work will augment the current capabilities of OPCAT so it can
  - model the various VisWeb requirements
  - provide the services of bi-directional RDF-VSW compilation.
- Design and build a Web crawler which will automatically generate VSW representations of knowledge stored in Web pages.
- Accomplishing even some of these goals will greatly benefit the huge World Wide Web user community.