New Zealand

Meta Tools For Implementing Domain Specific Visual Languages

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

John Grundy [1,2] and John Hosking[1]

[1]Dept. of Computer Science and [2] Dept of Electrical and Computer Engineering
University of Auckland
New Zealand
{john-g, john}@cs.auckland.ac.nz

The University of Auckland

Introductions

- Who
- Where from
- What like to get from this morning’s tutorial… (in 2 sentences or less ☺)

Outline

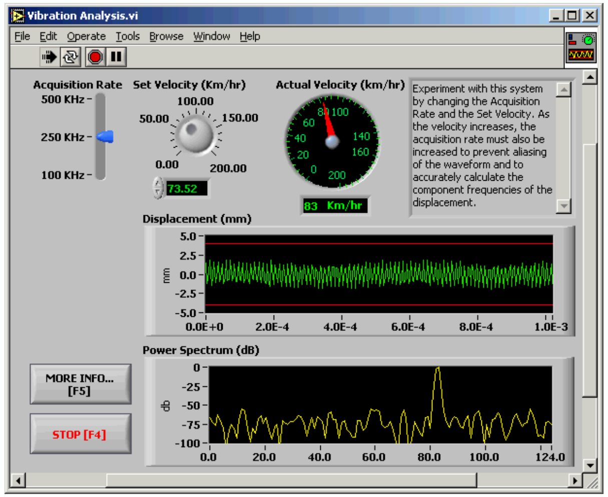
- **What' s a DSVL?**
- **Where do DSVLs arise?**
- **Relationship to Model Driven Design**
- **Elements of a DSVL & its environment**
- **Metamodels & their development**
  - Design exercise
- **DSVL Notation design**
  - Design exercise
- **Example meta tool**
  - DSVL implementation exercise
- **Other meta tools**
- **Wrap up**

What’s a Domain Specific Visual Language?

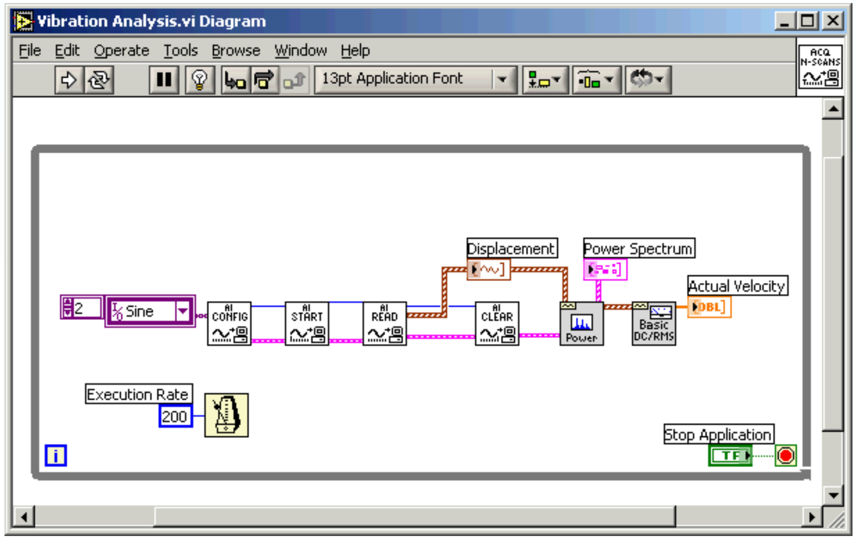
- **A DSVL is a visual language where the notation is *customised for a particular problem domain***
- **Have a trade off between generality of language (ie range of problems able to be solved) and terseness of notation and closeness of mapping**
- **Tim Menzies’ DSL preconditions (applicable to DSVLs)**
  - The 1 day rule:
    - Users can get productive with the DSL in 1 day.
    - Not *all users*, just *some users*.
      - Just the users who had the DSL created.
      - Implies that DSL is not just high-level programmin constructs;
      - But constructs for an audience.
  - The elbow test:
    - Users elbow the analyst out of the way in their haste to get to the screen to change something that is obviously wrong to them.
    - Implies rapid comprehension of sentences in the DSL.
      - *tim@menzies.com*

Example DSVL

- **LabView uses a visual dataflow metaphor but applied in a domain specific way**
  - Domain is **lab instrumentation**: access and analysis of sensor data attached to computer
  - Processing elements include math data transformations (eg FFTs, integrators, differentiators)
- **Very successful commercial Domain Specific VL http://www.ni.com/labview/**



Experiment with this system by changing the Acquisition Rate and the Set Velocity. As the velocity increases, the acquisition rate must also be increased to prevent aliasing of the waveform and to accurately calculate the component frequencies of the displacement.



Where do DSVLs come from?

- **XML configuration files are endemic**

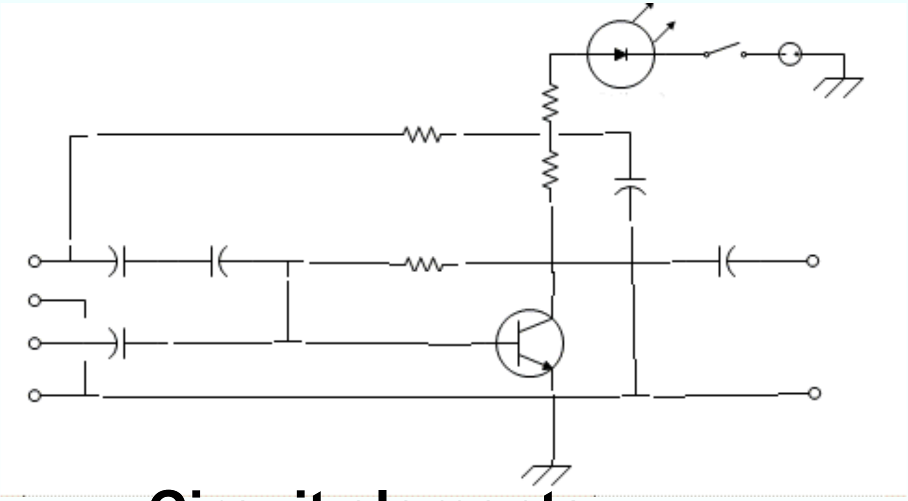
> “It's been almost a year since I wrote any Java, and I have finally figured out the real reason I gave up on it and switched to Python. It's nothing to do with the language itself--I often miss the sanity checks that come with strong typing. And it certainly wasn't because Java lacked tools, libraries, documentation, or an active developer community.
>
> No, the reason I switched can be summed in a single phrase, one that I've come to dread--XML configuration files. You have to write one for Ant to describe what you want to compile, a second for Hibernate to tell it how to map your classes to the tables in your database, a third for Tomcat to tell it how to map URLs and HTTP requests to your app, and on and on and on.”

Gregory Wilson, “It's the XML Configuration File's Fault” http://www.ddj.com/184407816

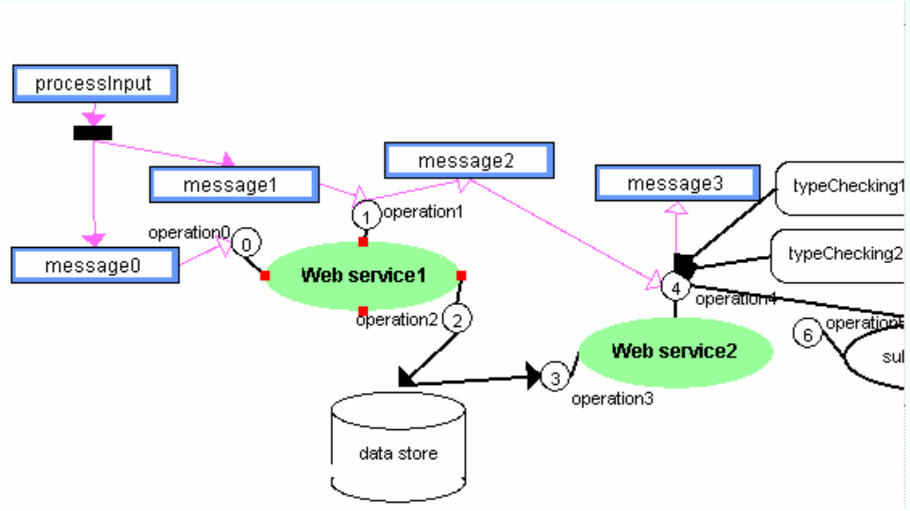
Where do DSVLs come from?

- **Companies don’t understand they need DSVLs but:**
- **They know they have problems configuring their products**
  - Configuration/customisation a common problem VL and SV can assist with
  - Natural consequence of the large framework/software product line evolution
  - Companies often spend large expensive programmer resource on it & want to de-skill to lower costs & make accessible to customers

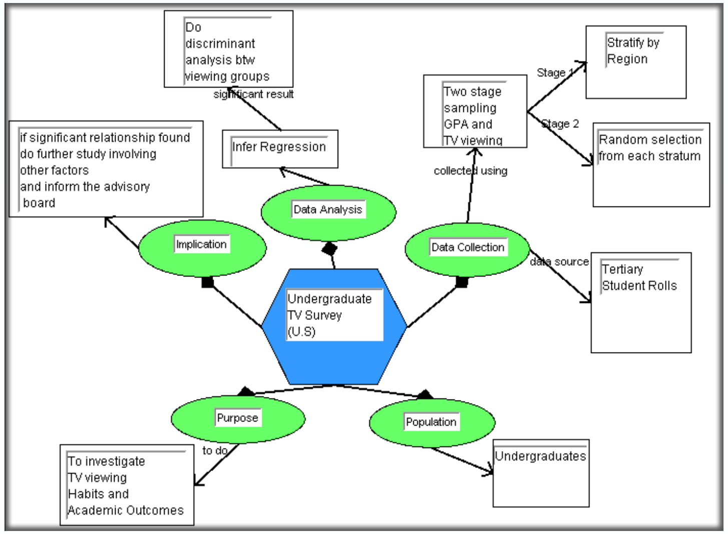
Domain data and metaphors



Circuit elements

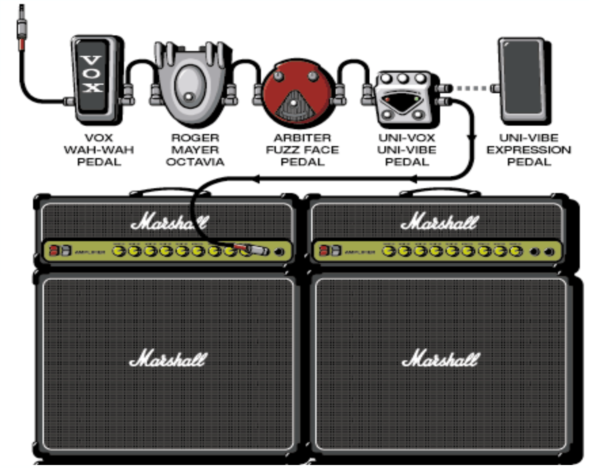


Services and Connectors


Do discriminant analysis btw viewing groups
significant result
Stratify by Region
Stage 1
Two stage sampling GPA and TV viewing
Stage 2
if significant relationship found do further study involving other factors and inform the advisory board
Infer Regression
Random selection from each stratum
collected using
Data Analysis
Implication
Data Collection
data source
Tertiary Student Rolls
Undergraduate TV Survey (U.S)
Purpose
Population
to do
To investigate TV viewing Habits and Academic Outcomes
Undergraduates

Statistical

Processes & Data

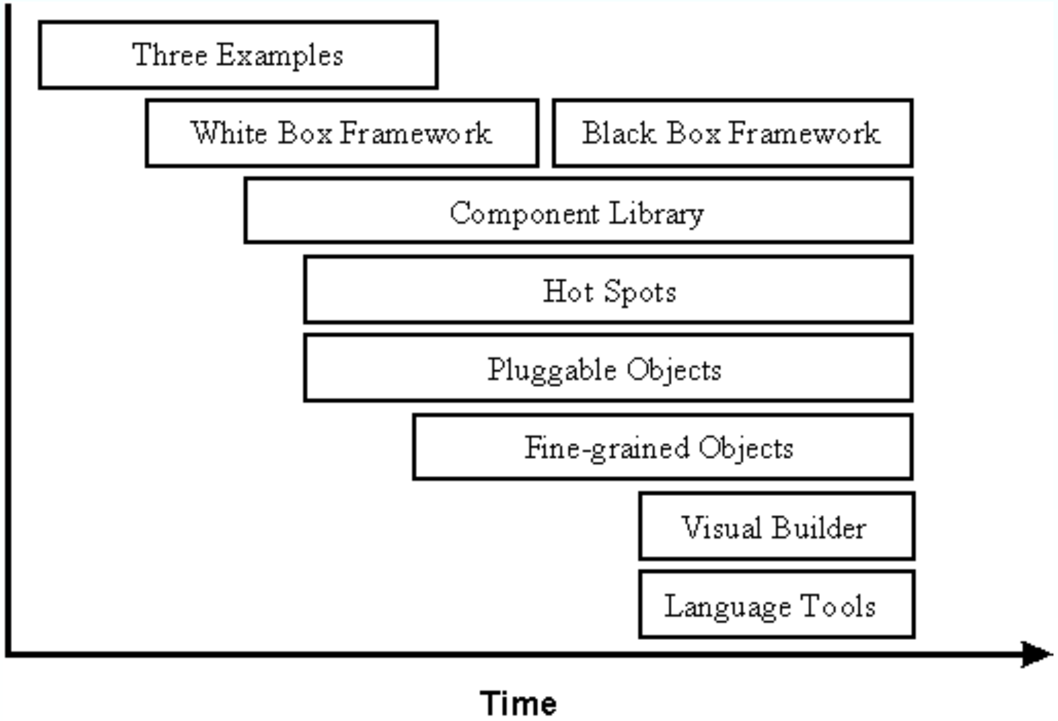


Music (from guitargeek.com)

Evolving Frameworks Pattern Language

- **EFPL tells us that Visual Builders (ie visual langs for configuration) & Language Tools (software visualizers) are a natural step in large framework evolution**

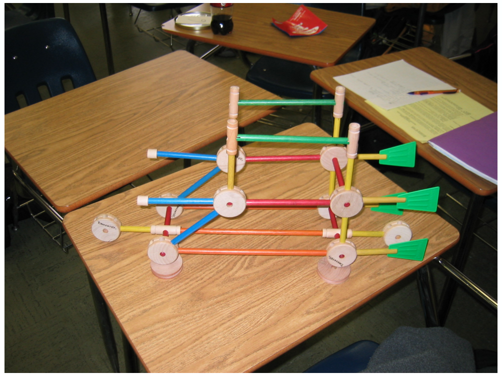
D. Roberts, R.Johnson "Evolving Frameworks"
http://st-www.cs.uiuc.edu
/users/droberts/evolve.html



- **Backed up by SEI software product line work & Microsoft Software Factory/DSL Tool approaches**

Model Driven Design and DSVLs

- **Model Driven Design is where applications are generated from high level models**
  - Typically the models are represented using DSVL(s)
- **This can be seen as a natural consequence of EFPL**
  - Moving from framework/text configuration language
    -> VL/visual tools
- **Model Driven Architecture (MDA) is MDD where the VL is UML (usually heavily stereotyped)**
  - Problems of lack of domain specificity in notation
- **Note: alternative is wizard approach**
  - problems with that: lack of overview,
    highly constrained

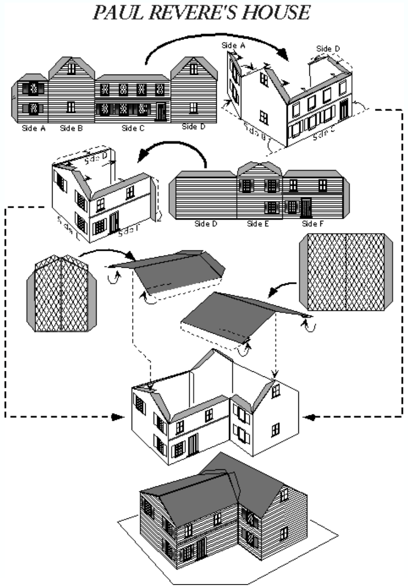


Elements in a DSVL specification

- **The notational elements**
  - Icons, connectors, metaphors
- **The notations**
  - Views/editors using those notational elements
- **The meta model**
  - Underlying model definition
- **Notation to model mappings**
  - One model element type may have multiple view repns
- **Behaviour and constraints**
  - Interaction/editing constraints and model constraints
- **Back end code generation (and import)**
  - The generation element for MDD

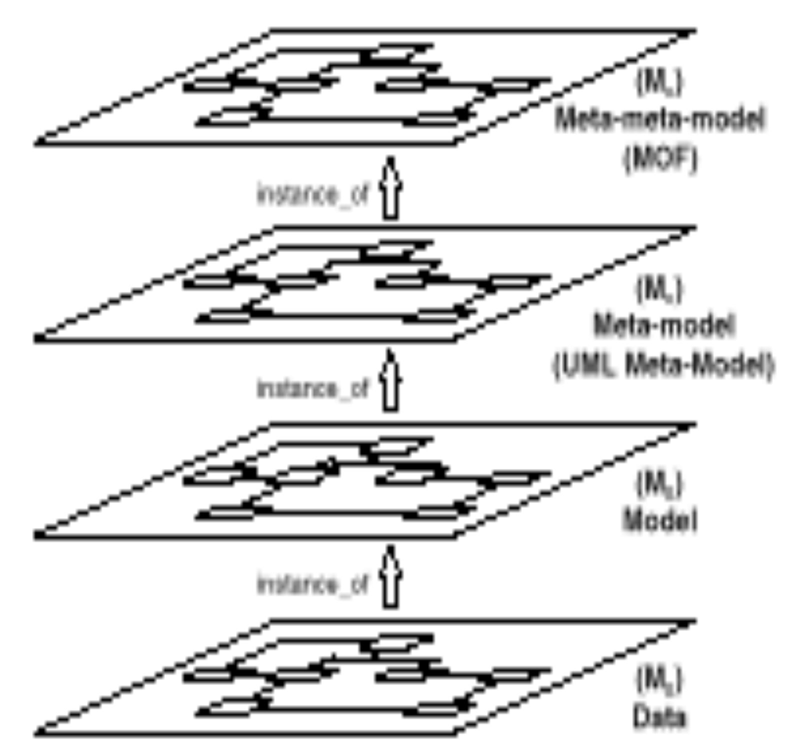
Meta models

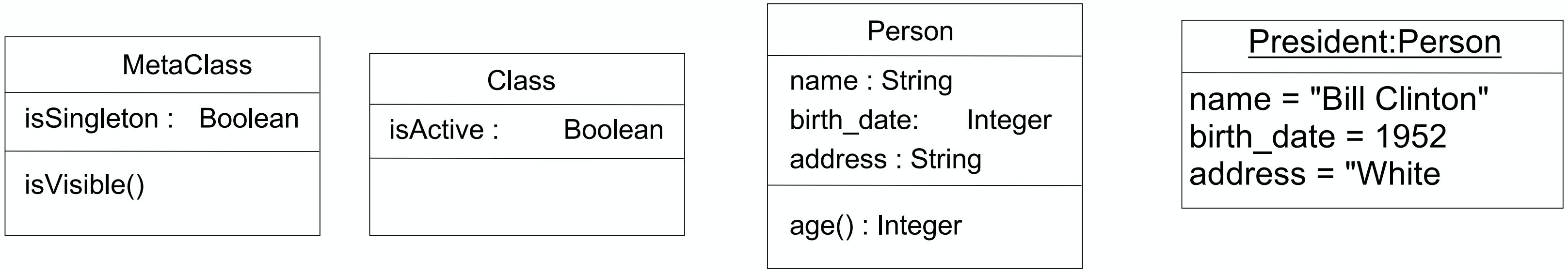
- **What’s a meta model?**
  - A model that defines/describes a model
  - Eg the UML meta model describes abstract concepts such as class type, association type, generalisation type, etc, that have instances in a particular model, (eg customer class, order class, customer-order association, customer-organisation generalisation)
- **How are they described?**
  - Using a meta modelling language
    - Eg MOF (UML class diagram like)
    - Eg Extended Entity Relationship



Meta-modelling approaches: MOF (UML)

- **MOF 4-level approach:**
  - M3: MOF MetaClass
  - M2: UML Class, instance of MOF Class; very similar to MOF concept of a Class
  - M1: Person, a typical instance of UML Class
  - M0: <u>President:Person</u>, a typical instance of Class Person
  - From C. Atkinson, Supporting and applying the UML conceptual framework.





MetaClass
isSingleton : Boolean
isVisible()

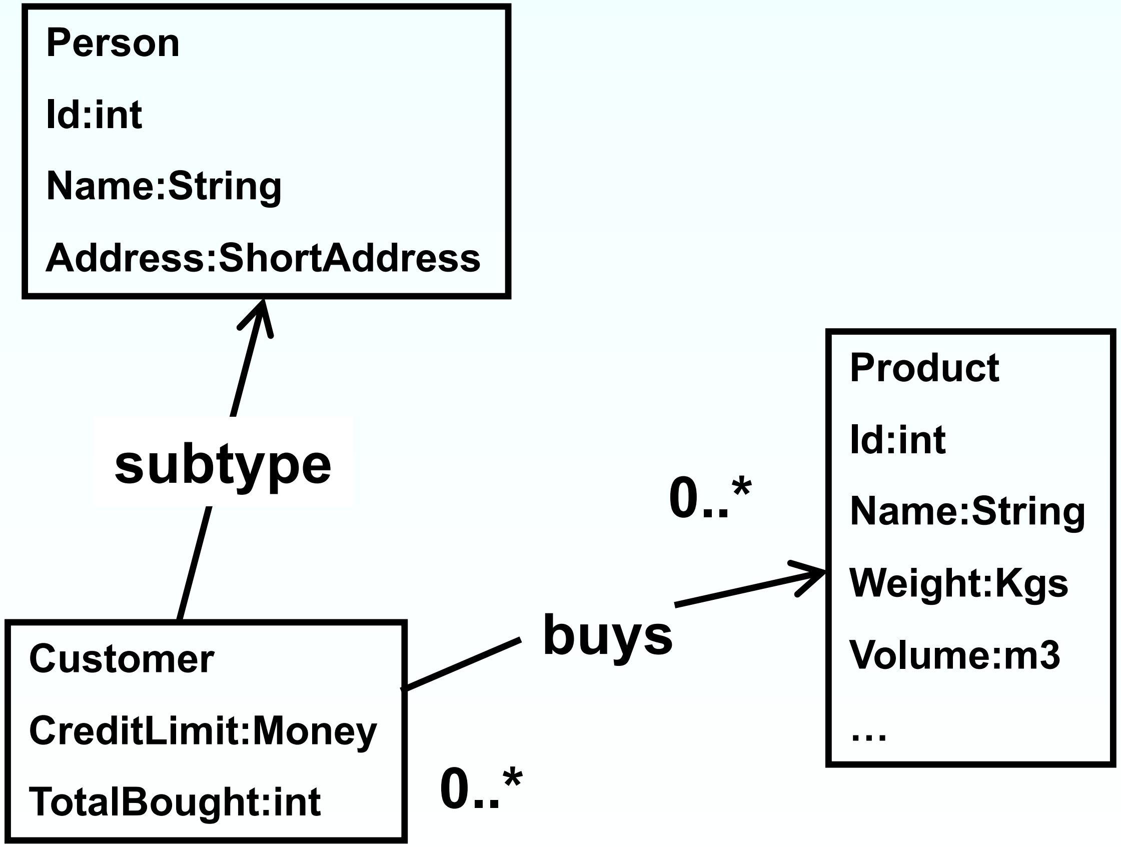
Class
isActive : Boolean

Person
name : String
birth_date: Integer
address : String
age() : Integer

<u>President:Person</u>
name = "Bill Clinton"
birth_date = 1952
address = "White

Meta-modelling approaches (ER)

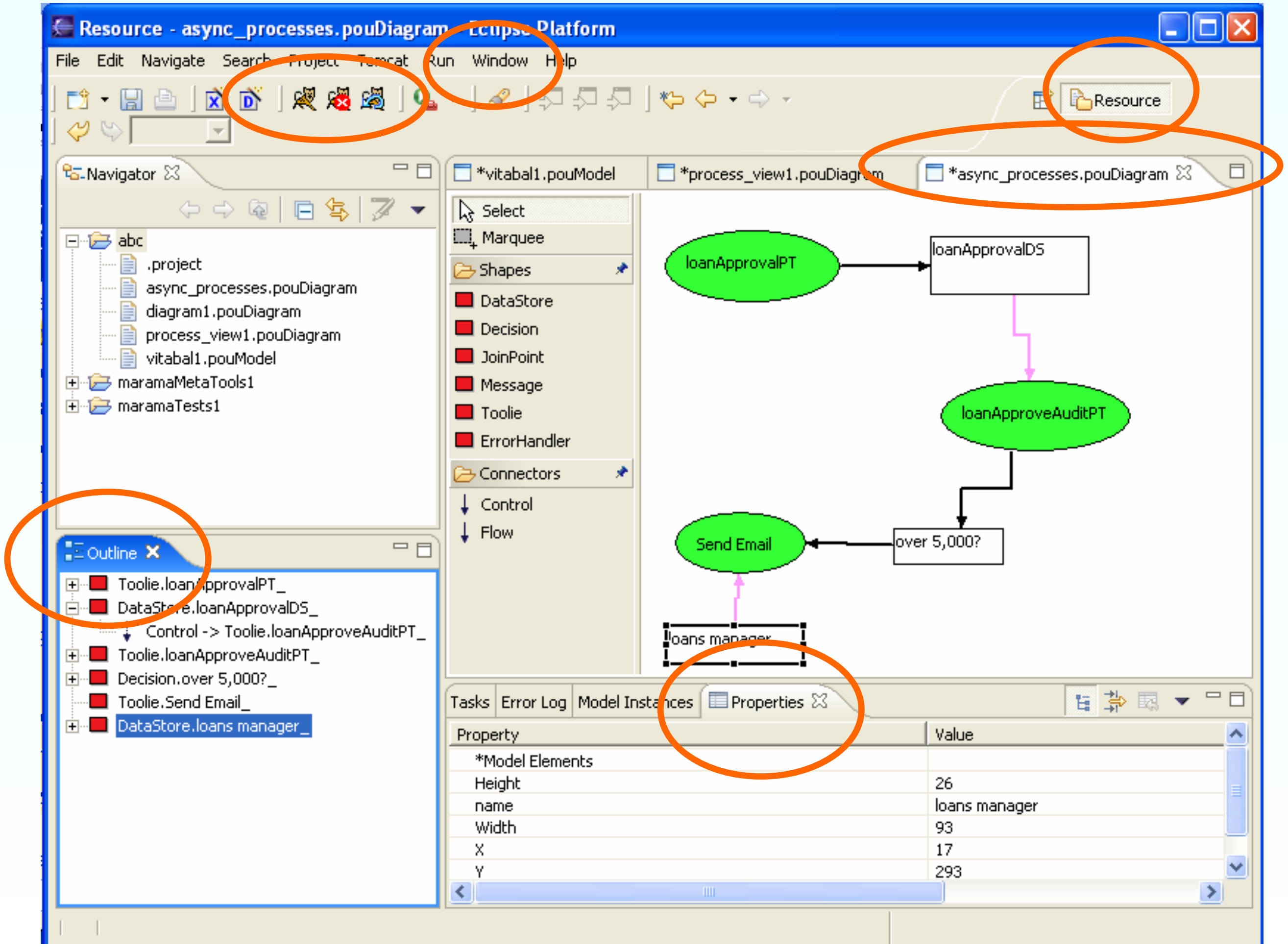
- **Entities, relationships**
- **Sub-typing (EER)**
- **Constraints e.g. arities (1:1, 0:n etc); exclusive-or; temporal (can-be connected to after…)**
- **Forms “schema” for data structures, database, …**



Exercise 1: Metamodel design

Problem: Configuring Eclipse plug-ins…

- **Eclipse IDE has a “plug-in” concept to add new elements to the environment**
- **Has an esoteric configuration file…**
- **For details:**



Property	Value
*Model Elements	
Height	26
name	loans manager
Width	93
X	17
Y	293

www.eclipse.org/documentation/pdf/org.eclipse.pde.doc.user_3.0.1.pdf

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.0"?>
<plugin
  id="nz.ac.auckland.cs.marama.MaramaEditor"
  name="Marama Editor Plug-in"
  version="1.0.0"
  provider-name="University of Auckland"
  class="nz.ac.auckland.cs.marama.MaramaEditorPlugin">

…
  <requires>
    <import plugin="nz.ac.auckland.cs.marama.MaramaModel" export="true"/>
    <import plugin="nz.ac.auckland.cs.marama.MaramaBasicHandlerLibrary" export="true"/>
    <import plugin="nz.ac.auckland.cs.marama.MaramaMTETool" export="true"/>
  </requires>

 <extension point="org.eclipse.ui.editors">
                    <editor
                                        name="Marama Eclipse Editor"
                                        extensions="pouDiagram, pouModel"
                                        icon="shapes.gif"
                                        default="true"
                                        class="nz.ac.auckland.cs.marama.MaramaEditor"
                                        contributorClass="nz.ac.auckland.cs.marama.MaramaEditorActionBarContributor"
                                        id="nz.ac.auckland.cs.marama.MaramaEditor" />
 </extension>

 <extension point="org.eclipse.ui.newWizards">
                    …
```

Define a meta-model for this…

- **Look at the example plug-in definition file provided**

- **What are the key elements of the model?**
- **What are their key properties?**
- **How are the elements related?**

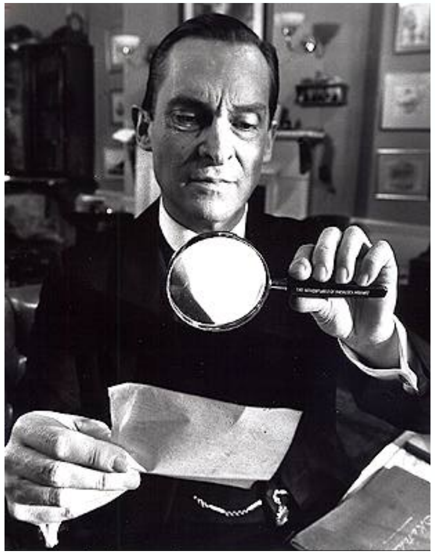
- **What is missing from the example you would have to find more about?**

- **How could this thing be made easier to understand, build…?**

Solution

Designing and Evaluating DSVLs

- **How "good" is a visual language?**
- **How can we design DSVLs so they meet users needs?**
  - Difficult:
  - Combination of psychology, user interface design, abstraction skills, expressability, narrowness of task, etc, etc
  - Typical usability studies are VERY expensive
  - Need some lightweight "tools" to help us understand the impact of design decisions
- **Look at:**
  - Basic approaches
  - End users and metaphors
  - Cognitive Dimensions
  - Attention Investment
  - Champagne Prototyping



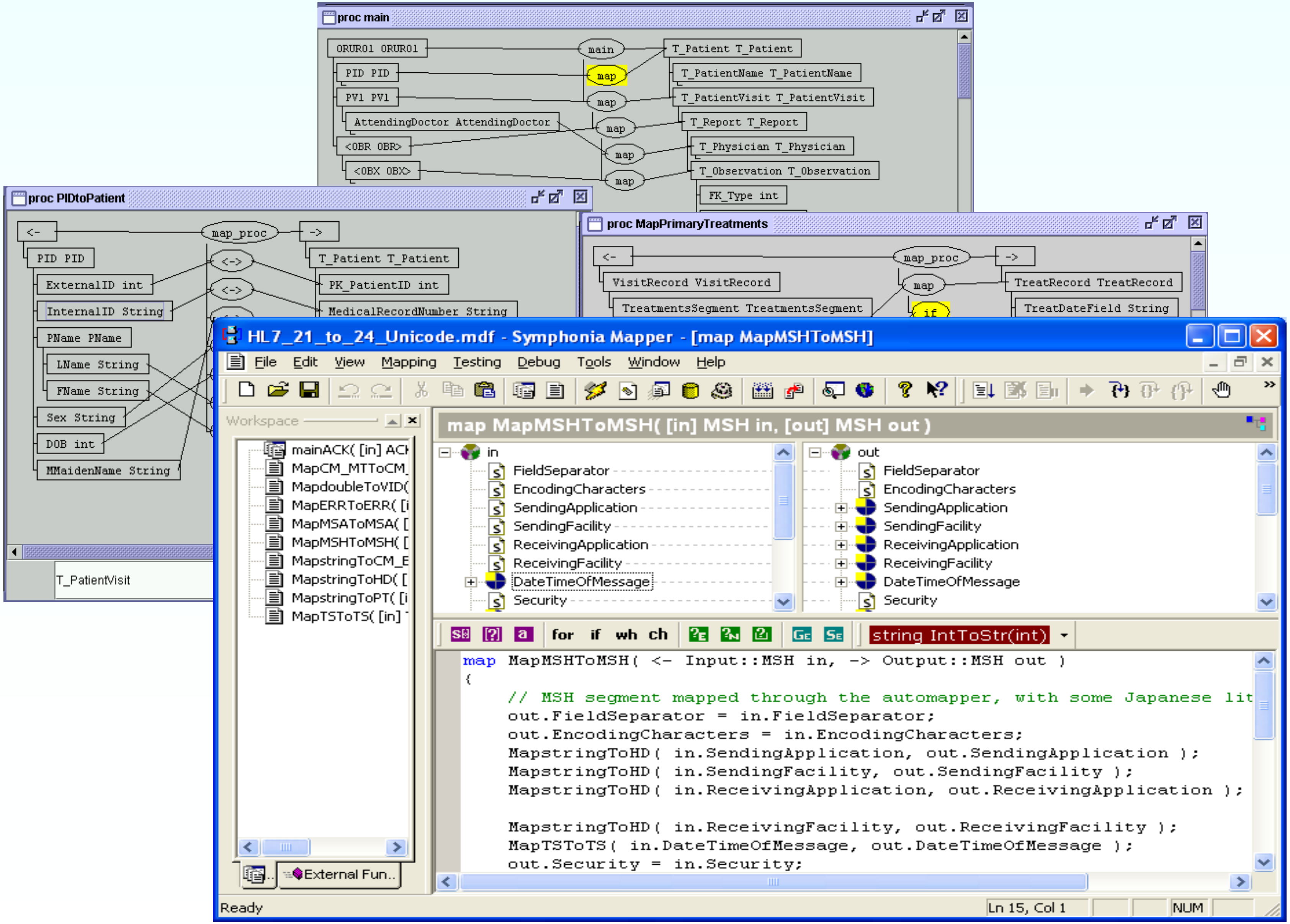
Basic notational approaches

- **Several main approaches, often combined together, differences mainly around how relationships represented:**
  - Icons plus connectors
    - Connectors may represent structure or flow or relationship
    - Eg Labview computational elements plus flow connectors
    - Eg Explorer hierarchical file view
    - Eg ER diagrams
  - Containment
    - Common for hierarchical systems as alternative to explorer style views. Common for structured components
    - Eg Labview blocks
    - Eg UML class icons
  - Proximity
    - Adjacency used as basis for a relationship
    - Eg Speadsheets, Grid languages such as KidSim/Cocoa

End users and metaphors

- **Understanding both the target domain and the end user are critical in DSVL design**
- **Target domain suggests metaphors/abstractions that may be useful**
  - Eg Labview: circuit diagrams
  - Eg Spreadsheet: financial table plus calculator
- **End user acceptance of metaphors is critical**
  - Can’t use abstractions that target end users don’t understand, don’t have affinity with
- **Example: tool for specifying mapping between health message formats**
  - Told the target end user was a DBA – someone familiar with data management, spreadsheet programming

End users and metaphors



```
map MapMSHToMSH( <- Input::MSH in, -> Output::MSH out )
{
    // MSH segment mapped through the automapper, with some Japanese lit
    out.FieldSeparator = in.FieldSeparator;
    out.EncodingCharacters = in.EncodingCharacters;
    MapstringToHD( in.SendingApplication, out.SendingApplication );
    MapstringToHD( in.SendingFacility, out.SendingFacility );
    MapstringToHD( in.ReceivingApplication, out.ReceivingApplication );

    MapstringToHD( in.ReceivingFacility, out.ReceivingFacility );
    MapTSToTS( in.DateTimeOfMessage, out.DateTimeOfMessage );
    out.Security = in.Security;
```

Cognitive Dimensions Framework

- **Green and Petre 1996 (since developed by Blackwell)**
  - See this afternoon's tutorial
- **Establishes a set of "dimensions" to think about the tradeoffs made in implementing visual programming *environments***
  - Means of explaining effects of design decisions
  - Has had very strong influence on the VL community
- **Comes out of cognitive psychology community**
- **Lightweight – doesn't need large usability studies to get useful insight**
- **Can be used for evaluation and also as a design aid**

Cognitive Dimensions

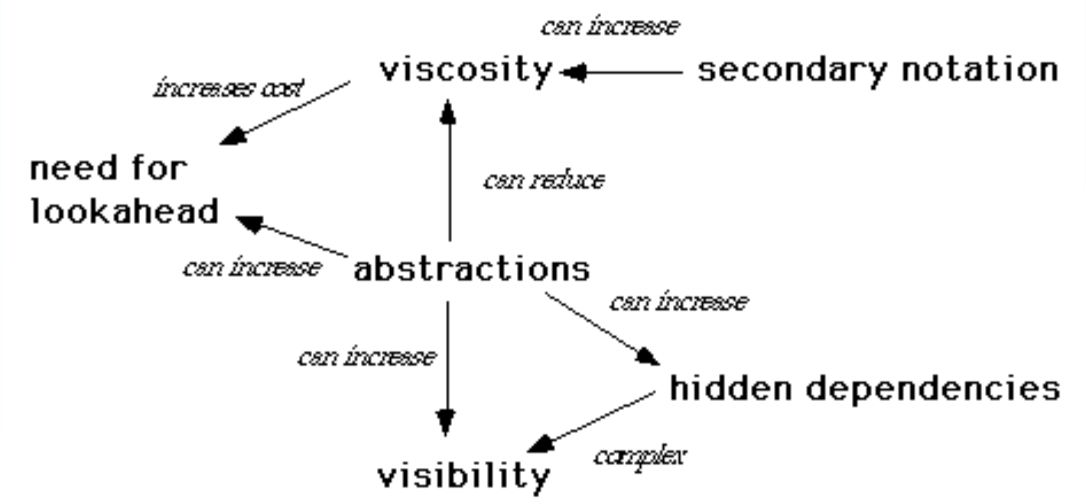
- Abstraction gradient What are the minimum and maximum levels of abstraction? Can fragments be encapsulated?
- Closeness of mapping What ‘programming games’ need to be learned?
- Consistency When some of the language has been learnt, how much of the rest can be inferred?
- Diffuseness How many symbols or graphic entities are required to express a meaning?
- Error-proneness Does the design of the notation induce ‘careless mistakes’?
- Hard mental operations Are there places where the user needs to resort to fingers or penciled annotation to keep track of what’s happening?
- Hidden dependencies Is every dependency overtly indicated in both directions? Is the indication perceptual or only symbolic?

Cognitive Dimensions

- Premature commitment Do programmers have to make decisions before they have the information they need?
- Progressive evaluation Can a partially-complete program be executed to obtain feedback on “How am I doing”?
- Role-expressiveness Can the reader see how each component of a program relates to the whole?
- Secondary notation Can programmers use layout, color, or other cues to convey extra meaning, above and beyond the ‘official’ semantics of the language?
- Viscosity How much effort is required to perform a single change?
- Visibility Is every part of the code simultaneously visible (assuming a large enough display), or is it at least possible to compare any two parts side-by-side at will? If the code is dispersed, is it at least possible to know in what order to read it?

Use of Cognitive Dimensions

- **Note the tradeoffs that occur**
  - May add an abstraction that makes it easier to change things (reduced viscosity) but increases the difficulty of understanding (increased abstraction gradient and increased hidden dependencies).
  - See Green and Petre paper for several examples illustrating tradeoffs made



- **Burnett provides a set of representation benchmarks that assist in operationalising the use of the CD framework.**
  - See http://web.engr.oregonstate.edu/~burnett/reprints.html

Cognitive Dimensions provides vocabulary

Verbatim transcript from a newsgroup discussion (real words from real users).

NB: this discussion referred to a version of Framemaker that is now obsolete.

- *A:* ALL files in the book should be identical in everything except body pages. Master pages, paragraph formats, reference pages, should be the same.
- *B:* Framemaker does provide this ... File -> Use Formats allows you to copy all or some formatting categories to all or some files in the book.
- *A:* Grrrrrrrrr .... Oh People Of Little Imagination !!!!!!
- Sure I can do this ... manually, every time I change a reference page, master page, or paragraph format .....
- What I was talking about was some mechanism that automatically detected when I had made such a change. (.....) Or better yet, putting all of these pages in a central database for the entire book ......
- *C:* There is an argument against basing one paragraph style on another, a method several systems use. A change in a parent style may cause unexpected problems among the children. I have had some unpleasant surprises of this sort in Microsoft Word.

Improved Discussion

- *A:* Framemaker is too viscous.
- *B:* With respect to what task?
- *A:* With respect to updating components of a book. It needs to have a higher abstraction level, such as a style tree.
- *C:* Watch out for the hidden dependencies of a style tree.
- *(further possible comments)*
- The abstraction level will be difficult to master; getting the styles right may impose lookahead.

From: An Introduction to the Cognitive Dimensions Framework, T R G Green

http://homepage.ntlworld.com/greenery/workStuff/Papers/introCogDims/index.html

Labview in CD terms

- **Metaphor used – dataflow wiring plus computation blocks – has *high closeness of mapping***
  - End users are electronic engineers – very familiar with circuit wiring
- **Modularity via blocks – again very similar to electrical circuit concepts hence *low abstraction gradient* for end users and *hidden dependencies* are of a sort that end users are familiar with**
- **Problems of *high viscosity* due to layout reorganisation not an major issue with user audience – familiar with these problems from circuit design tools**
- **Language relatively *terse* at one level (general concepts) but quite *diffuse* at another (many predefined operations with their own iconic representation)**
- **Attention to front end – ability to create realistic looking virtual instrument front panel, providing *high closeness of mapping* for end users**

Spreadsheet in CD terms

- **Strong and consistent metaphor providing high *closeness of mapping* to typical balance sheet etc problems**
- **At one level notation is quite *terse* (sheet and cell metaphor), at another it is quite *verbose* (extensive range of functions that stretch the bounds of the metaphor)**
- ***Progressive evaluation* well supported: values calculated immediately a formula entered**
- ***Hidden dependencies* a real issue – a strong cause of errors, ie leading to *error proneness***

Champagne Prototyping

- **A “cheap” method for early design evaluation**
- **Combines:**
  - simple prototyping
    - used overlays and “look don’t touch” approach
  - cognitive walkthroughs with credible participants
  - cognitive dimensions & attention investment for analysis

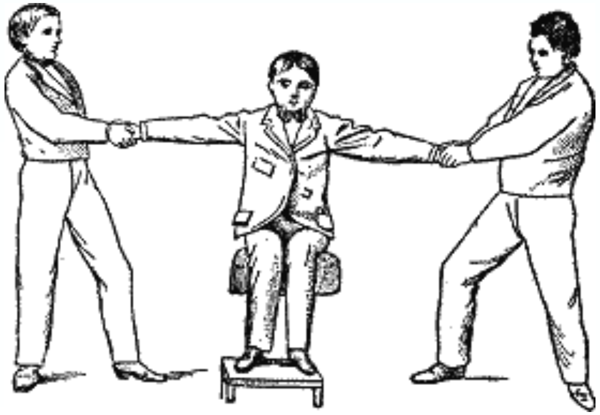
to assist in answering questions at early design phase of visual environments

- **Blackwell, Burnett and Peyton Jones, Champagne Prototyping: a research technique for early evaluation of complex end user programming systems, IEEE VL/HCC, 2004, 47-54**

Final notation design comments

- **Design is a creative process: examine multiple candidate designs, don't just develop one**
- **Try to find notations that are in some senses natural for the end user (ie rate closeness of mapping highly as a cognitive dimension)**
  - Function blocks + wiring for Labview
  - Table + calculator for spreadsheet
  - Tree or Form + drag and connect + formula for Mapper
- **Look to reuse diagrams at execution time to visualise behaviour at the same level of abstraction used to construct the program (moving towards liveness/progressive evaln and concreteness but recognising that compile cycle inevitable in many applications)**
- **Common to use terse high level abstractions and more verbose lower level detail (often textual) which gives some hidden dependencies and can lead to error proneness (cf spreadsheets)**

Exercise 2: Notation design



Problem

- **Have meta-model for Eclipse plug-in description file**
- **Want one (or more!) visual notation(s) that allow users to build up a specification from visual building-blocks vs edit the XML or use form-based editor (like the Eclipse PDE plug-in…)**

- **What are the visual metaphor(s)?**
- **What are the shapes & connectors?**
- **How do we relate shape/connector to meta-model elements (entities, relationships)?**
- **What consistency constraints are their e.g. layout; change property=> change …; delete shape => ??**

Solution

Meta tools

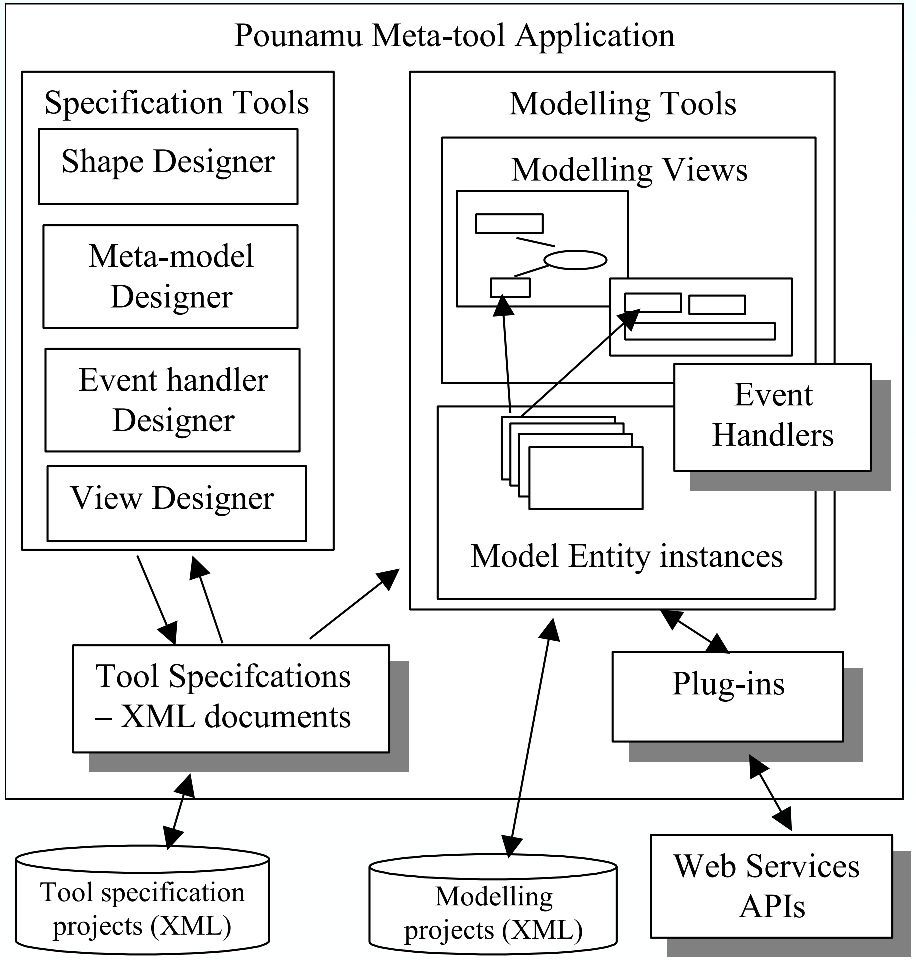
- **What's a meta tool?**
  - A tool that allows you to define meta models and notations which can be used to *generate* environments for modelling using the notations
- **Needs means of specifying the DSVL (cf earlier)**
  - Notation or notations – metaphors, elements, views
  - Meta model definition
  - Notation-model mapping
  - Behaviour and constraints
  - Back end code generation and import

Pounamu – exemplar meta tool

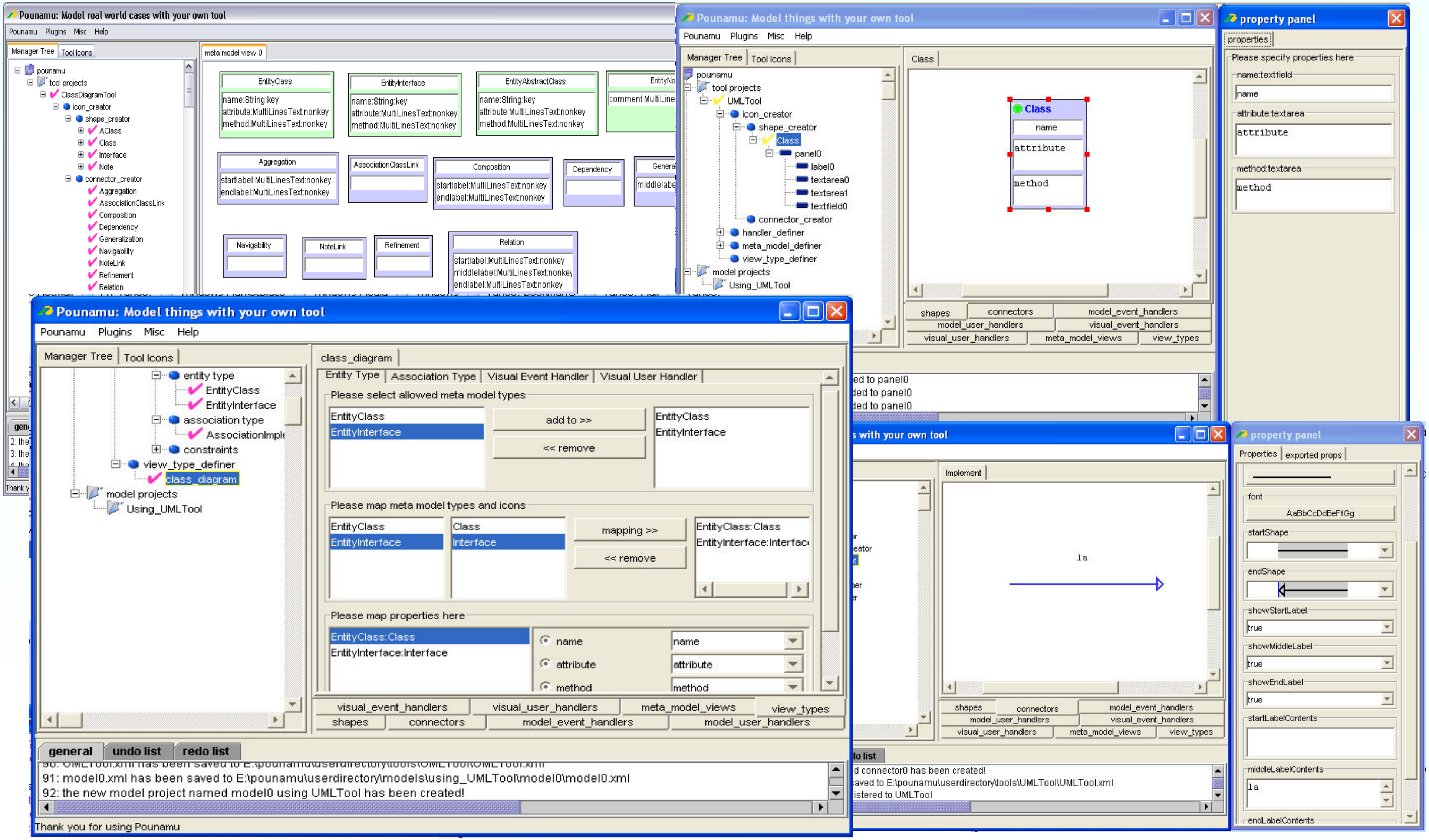
- **Pounamu overarching design requirements**
  - Simplicity of use.
    - It should be very easy to express the design of a visual notation, and generate an environment to support modelling using the notation.
  - Simplicity of extension and modification.
    - It should be possible to rapidly evolve proof of concept tools by modification of the notation, addition of back end processing, integration with other tools, and behavioural extensions (eg complex constraints).
- **Led to a lightweight structure, with extensibility, customisation strongly built in, plus web services interface**

Pounamu components

- **Shape creator and connector creator tools**
  - Used to define icons, connectors and associated properties
- **Event Handler Designer tool**
  - Specifies dynamic behaviour in response to events (eg shape creation). Currently Java code using API. Have two experimental DSVLs for this as well.
- **Meta model designer tool**
  - Specifies tool meta models
- **View type designer tool**
  - Specifies an editor for a set of shapes, connectors and handlers, and their relationship to a meta model
- **Model projects**
  - Instances of a specified tool in use



Pounamu designer examples



EntityClass: name:String:key, attribute:MultiLinesText:nonkey, method:MultiLinesText:nonkey

EntityInterface: name:String:key, attribute:MultiLinesText:nonkey, method:MultiLinesText:nonkey

EntityAbstractClass: name:String:key, attribute:MultiLinesText:nonkey, method:MultiLinesText:nonkey

EntityNo…: comment:MultiLine…

Aggregation: startlabel:MultiLinesText:nonkey, endlabel:MultiLinesText:nonkey

Composition: startlabel:MultiLinesText:nonkey, endlabel:MultiLinesText:nonkey

Genera…: middlelabe…

Relation: startlabel:MultiLinesText:nonkey, middlelabel:MultiLinesText:nonkey, endlabel:MultiLinesText:nonkey

Please specify properties here

name:textfield: name

attribute:textarea: attribute

method:textarea: method

Please select allowed meta model types

EntityClass
EntityInterface

EntityClass
EntityInterface

Please map meta model types and icons

EntityClass → Class
EntityInterface → Interface

EntityClass:Class
EntityInterface:Interface

Please map properties here

EntityClass:Class
EntityInterface:Interface

○ name: name
○ attribute: attribute
○ method: method

```
…ed to panel0
…ed to panel0
…ed to panel0
```

```
90: UMLTool.xml has been saved to E:\pounamu\userdirectory\tools\UMLTool\UMLTool.xml
91: model0.xml has been saved to E:\pounamu\userdirectory\models\using_UMLTool\model0\model0.xml
92: the new model project named model0 using UMLTool has been created!
```

Thank you for using Pounamu

```
…d connector0 has been created!
…aved to E:\pounamu\userdirectory\tools\UMLTool\UMLTool.xml
…istered to UMLTool
```

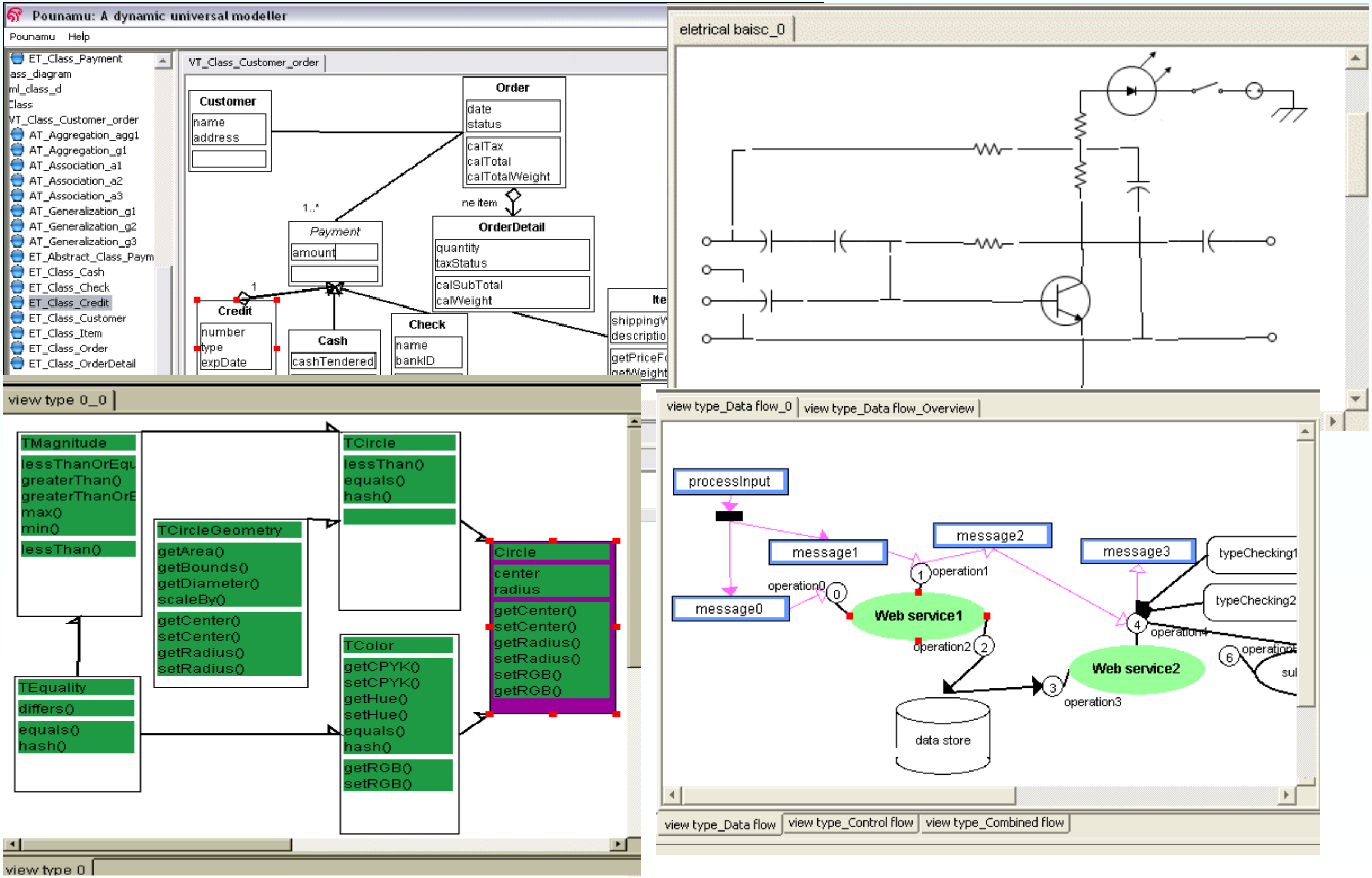
showStartLabel: true

showMiddleLabel: true

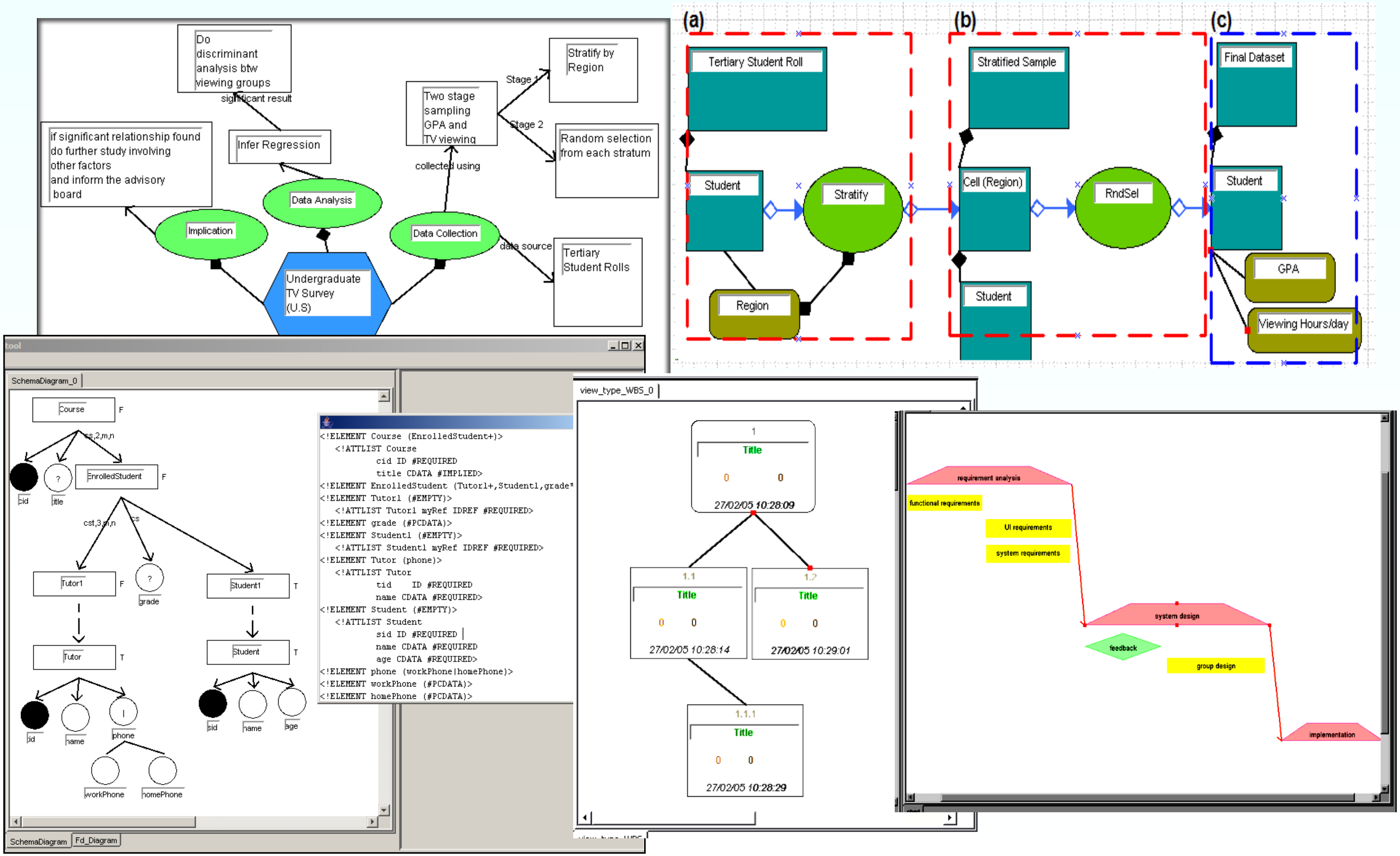
showEndLabel: true

middleLabelContents: la

Examples


Pounamu: A dynamic universal modeller
Pounamu Help
ET_Class_Payment
ass_diagram
ml_class_d
Class
VT_Class_Customer_order
AT_Aggregation_agg1
AT_Aggregation_g1
AT_Association_a1
AT_Association_a2
AT_Association_a3
AT_Generalization_g1
AT_Generalization_g2
AT_Generalization_g3
ET_Abstract_Class_Paym
ET_Class_Cash
ET_Class_Check
ET_Class_Credit
ET_Class_Customer
ET_Class_Item
ET_Class_Order
ET_Class_OrderDetail
VT_Class_Customer_order
Customer
name
address
Order
date
status
calTax
calTotal
calTotalWeight
ne item
1..*
Payment
amount
OrderDetail
quantity
taxStatus
calSubTotal
calWeight
1
Credit
number
type
expDate
Cash
cashTendered
Check
name
bankID
Ite
shippingW
descriptio
getPriceF
getWeight
eletrical baisc_0
view type 0_0
TMagnitude
lessThanOrEqu
greaterThan()
greaterThanOrE
max()
min()
lessThan()
TCircle
lessThan()
equals()
hash()
TCircleGeometry
getArea()
getBounds()
getDiameter()
scaleBy()
getCenter()
setCenter()
getRadius()
setRadius()
Circle
center
radius
getCenter()
setCenter()
getRadius()
setRadius()
setRGB()
getRGB()
TEquality
differs()
equals()
hash()
TColor
getCPYK()
setCPYK()
getHue()
setHue()
equals()
hash()
getRGB()
setRGB()
view type 0
view type_Data flow_0
view type_Data flow_Overview
processInput
message1
message2
message3
typeChecking1
typeChecking2
operation0
operation1
message0
0
1
Web service1
operation2
2
4
operation4
operation5
6
sul
Web service2
3
operation3
data store
view type_Data flow
view type_Control flow
view type_Combined flow

Examples



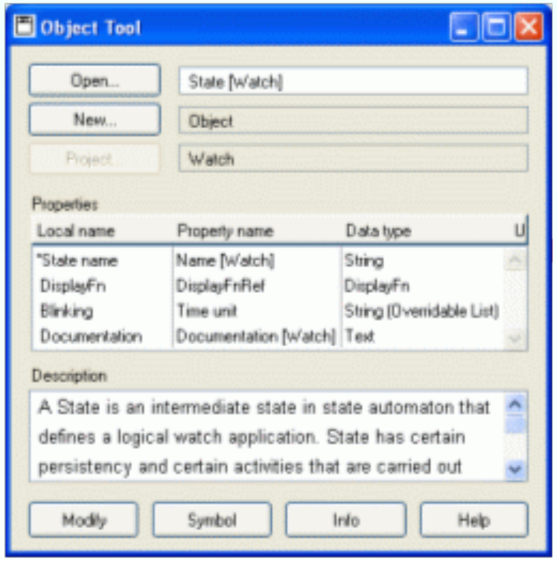
```
<!ELEMENT Course (EnrolledStudent+)>
   <!ATTLIST Course
          cid ID #REQUIRED
          title CDATA #IMPLIED>
<!ELEMENT EnrolledStudent (Tutor1+,Student1,grade*
<!ELEMENT Tutor1 (#EMPTY)>
   <!ATTLIST Tutor1 myRef IDREF #REQUIRED>
<!ELEMENT grade (#PCDATA)>
<!ELEMENT Student1 (#EMPTY)>
   <!ATTLIST Student1 myRef IDREF #REQUIRED>
<!ELEMENT Tutor (phone)>
   <!ATTLIST Tutor
          tid    ID #REQUIRED
          name CDATA #REQUIRED>
<!ELEMENT Student (#EMPTY)>
   <!ATTLIST Student
          sid ID #REQUIRED
          name CDATA #REQUIRED
          age CDATA #REQUIRED>
<!ELEMENT phone (workPhone|homePhone)>
<!ELEMENT workPhone (#PCDATA)>
<!ELEMENT homePhone (#PCDATA)>
```

Exercise 3: Pounamu demonstration

MetaEdit+

- **Commercial system from MetaCASE (cost E11,500) www.metacase.com**
  - (ex MetaEdit from U Jyvaskyla Finland)
- **Variety of text/form based tools to specify meta model**
  - Objects
  - Properties (attributes)
  - Relationships and Roles (endpoints)
  - Ports (constraints on connection points)
  - Graph (like Pounamu view tool)
- **Symbol and Dialog Box Editors**
- **Reports and generators (walk data structures to generate reports, code)**
- **External interfaces**
- **Model editors include diagrams, matrices, tables, browsers**

MetaEdit+



State [Watch]

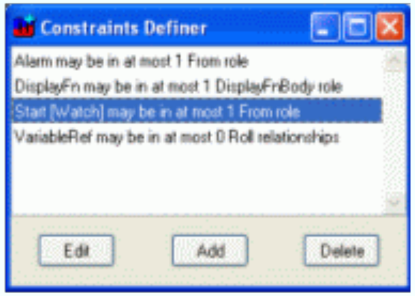
Object

Watch

Local name	Property name	Data type
*State name	Name [Watch]	String
DisplayFn	DisplayFnRef	DisplayFn
Blinking	Time unit	String (Overridable List)
Documentation	Documentation [Watch]	Text

A State is an intermediate state in state automaton that defines a logical watch application. State has certain persistency and certain activities that are carried out

Object Tool



Alarm may be in at most 1 From role

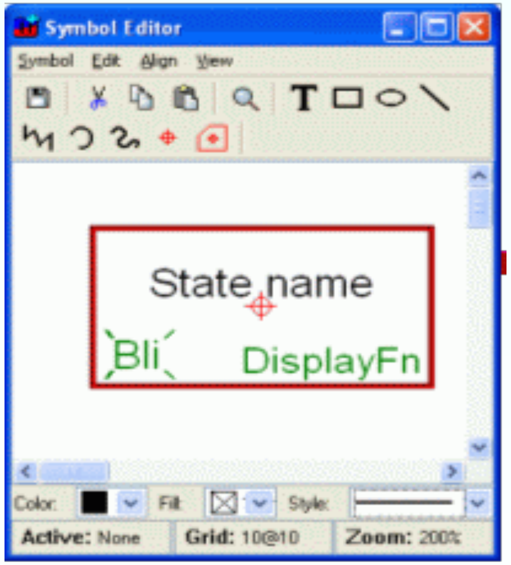
DisplayFn may be in at most 1 DisplayFnBody role

Start [Watch] may be in at most 1 From role

VariableRef may be in at most 0 Roll relationships

Constraints

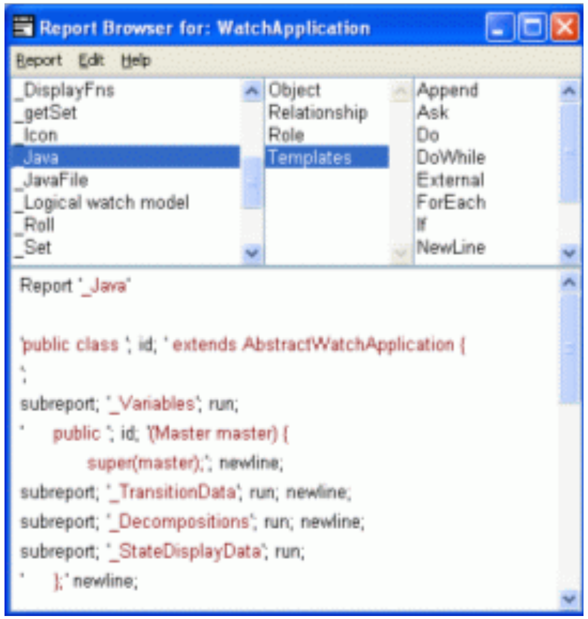
Symbol editor



Active: None

Grid: 10@10

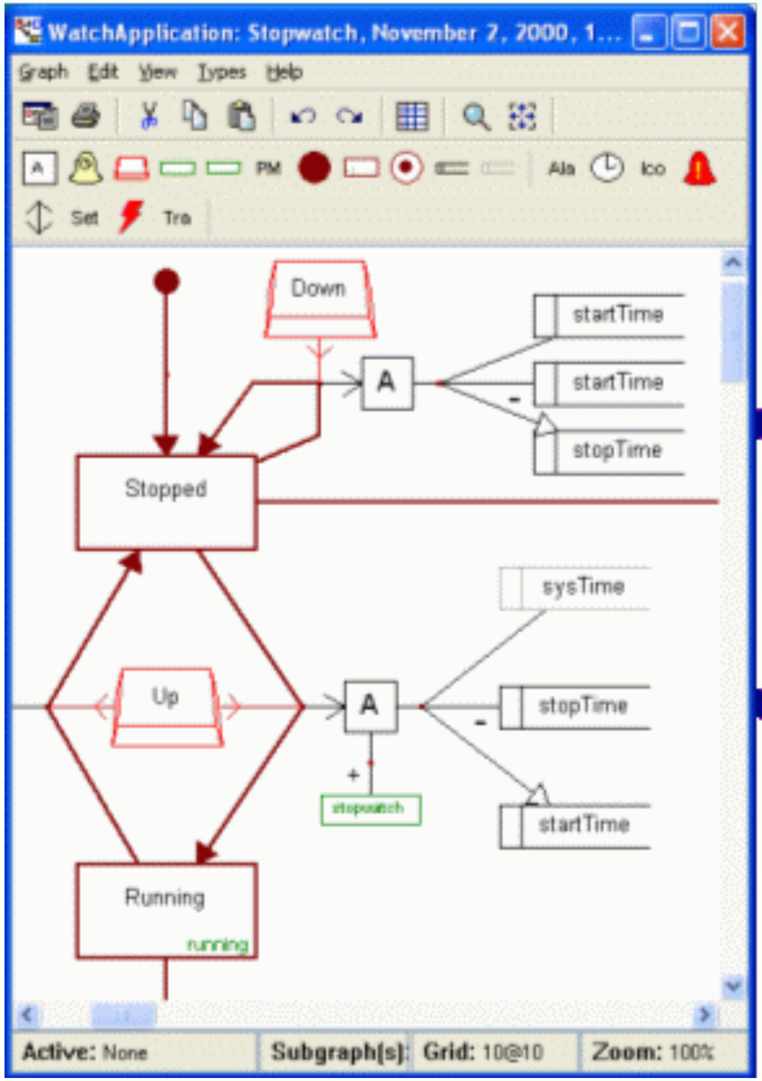
Zoom: 200%

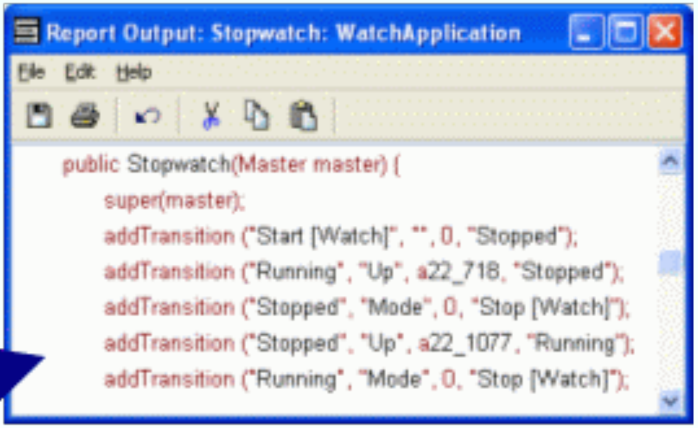


```
Report '_Java'

'public class '; id; ' extends AbstractWatchApplication {
';
subreport; '_Variables'; run;
'     public '; id; '(Master master) {
          super(master);'; newline;
subreport; '_TransitionData'; run; newline;
subreport; '_Decompositions'; run; newline;
subreport; '_StateDisplayData'; run;
'     };' newline;
```

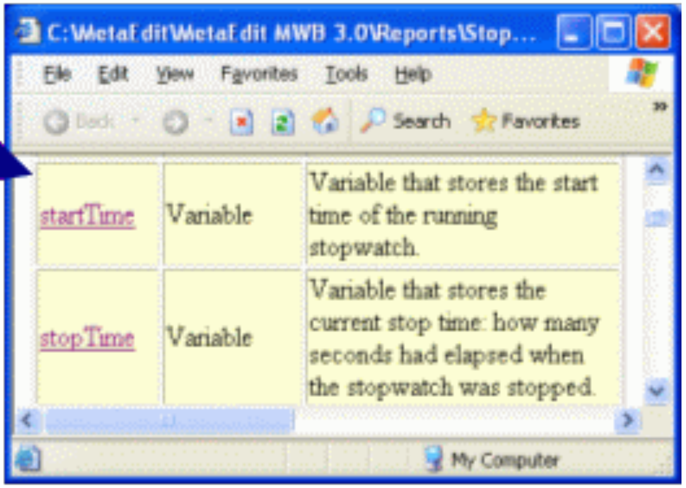
Generator

MetaEdit+ Generated System





```
public Stopwatch(Master master) {
    super(master);
    addTransition ("Start [Watch]", "", 0, "Stopped");
    addTransition ("Running", "Up", a22_718, "Stopped");
    addTransition ("Stopped", "Mode", 0, "Stop [Watch]");
    addTransition ("Stopped", "Up", a22_1077, "Running");
    addTransition ("Running", "Mode", 0, "Stop [Watch]");
```



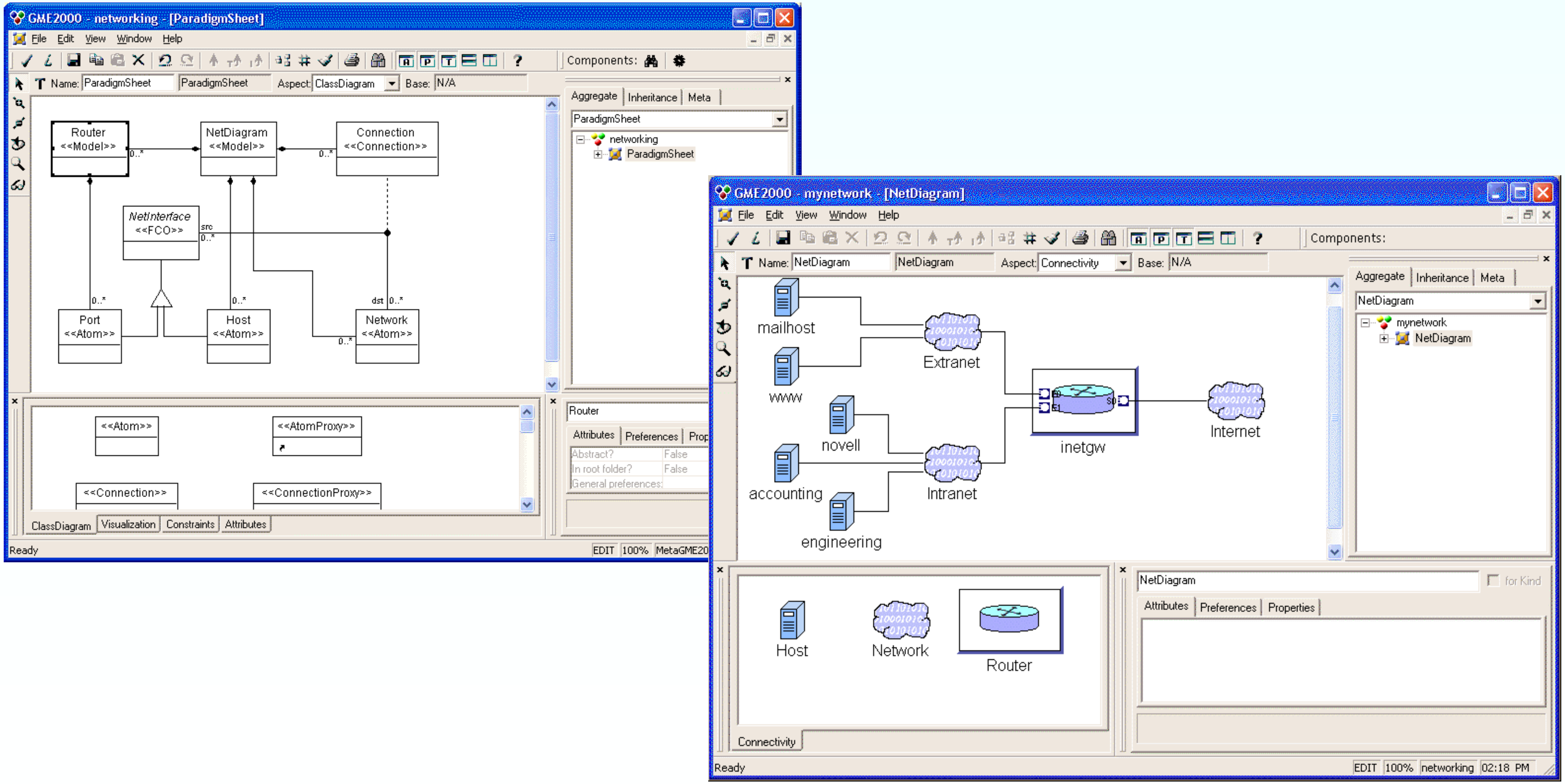
C:\MetaEdit\MetaEdit MWB 3.0\Reports\Stop...
File Edit View Favorites Tools Help
Back
Search
Favorites
My Computer

startTime	Variable	Variable that stores the start time of the running stopwatch.
stopTime	Variable	Variable that stores the current stop time: how many seconds had elapsed when the stopwatch was stopped.

GME

- **Generic Modelling Environment, Ledeczi et al, Vanderbuilt**
- **http://www.isis.vanderbilt.edu/Projects/gme/default.html**
- **Visual MetaModel composed of several parts**
  - Class diagram with stereotypes representing metatype
    - Metatypes defined by MetaGME meta model
    - Atoms, connections, models
  - Attributes, constraints
    - Constraints represented using OCL (see UML later)
  - Visualization
    - Like Pounamu view definer – defines *aspects*
    - Symbols from simple built-in symbols or bitmaps + code for more complex symbols
- **Extensibility via COM interfaces and XML import/export**

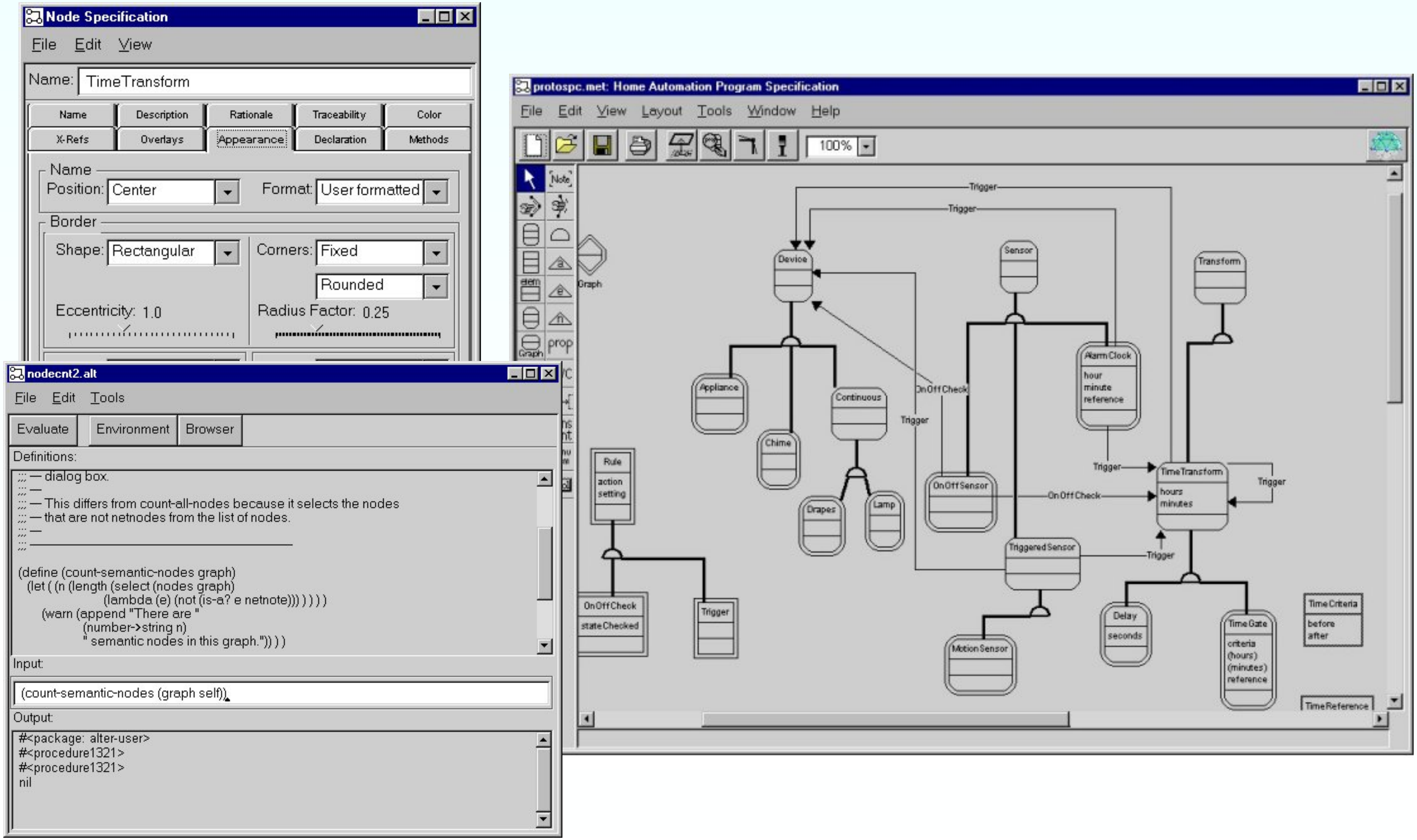
GME Example


GME2000 - networking - [ParadigmSheet]
File Edit View Window Help
Components:
Name: ParadigmSheet
ParadigmSheet
Aspect: ClassDiagram
Base: N/A
Router <<Model>>
NetDiagram <<Model>>
Connection <<Connection>>
NetInterface <<FCO>>
src
dst
0..*
Port <<Atom>>
Host <<Atom>>
Network <<Atom>>
<<Atom>>
<<AtomProxy>>
<<Connection>>
<<ConnectionProxy>>
ClassDiagram
Visualization
Constraints
Attributes
Aggregate
Inheritance
Meta
networking
Router
Preferences
Abstract? False
In root folder? False
General preferences:
Ready
EDIT 100% MetaGME20
GME2000 - mynetwork - [NetDiagram]
Name: NetDiagram
NetDiagram
Aspect: Connectivity
mailhost
www
novell
accounting
engineering
Extranet
Intranet
inetgw
Internet
Host
Network
Router
Connectivity
mynetwork
NetDiagram
for Kind
Properties
Ready
EDIT 100% networking 02:18 PM

DOME

- **Notations defined by filling in properties on an object model using the DOME Tool Specification Language.**
  - Includes object class, property and relationship definitions, connector types, dynamic object appearances, tool buttons, menus, annotations, and semantic relationships.
  - Graphical languages can also include textual, numeric, and symbolic annotations.
- **Graphical meta-modeling capability ProtoDOME**
  - allows specn of new notations and running them in an interpreted mode.
- **Projector and Alter are DOME's code and document generation tools:**
  - Projector, is a visual dataflow language;
  - Alter, a functional textual language
  - Both provide functionality to write complex model transformations.
- **http://www.htc.honeywell.com/dome**

DOME – Tool Specn



Node Specification

Name: TimeTransform

Name

Position: Center

Format: User formatted

Border

Shape: Rectangular

Corners: Fixed

Rounded

Eccentricity: 1.0

Radius Factor: 0.25

nodecnt2.alt

Definitions:

```
;;; — dialog box.
;;; —
;;; — This differs from count-all-nodes because it selects the nodes
;;; — that are not netnodes from the list of nodes.
;;; —
;;; ——————————————————————————
;;;

(define (count-semantic-nodes graph)
  (let ( (n (length (select (nodes graph)
                     (lambda (e) (not (is-a? e netnote))) ) ) ) )
     (warn (append "There are "
                 (number->string n)
                 " semantic nodes in this graph.")) ) )
```

Input:

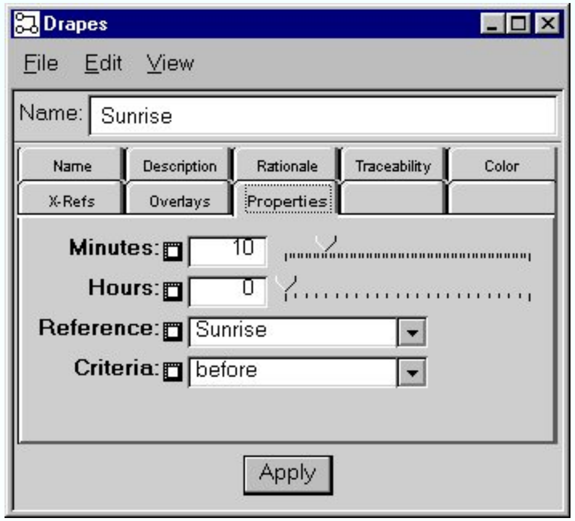
```
(count-semantic-nodes (graph self))
```

Output:

```
#<package: alter-user>
#<procedure1321>
#<procedure1321>
nil
```

protospc.met: Home Automation Program Specification

DOME Model Instantiation



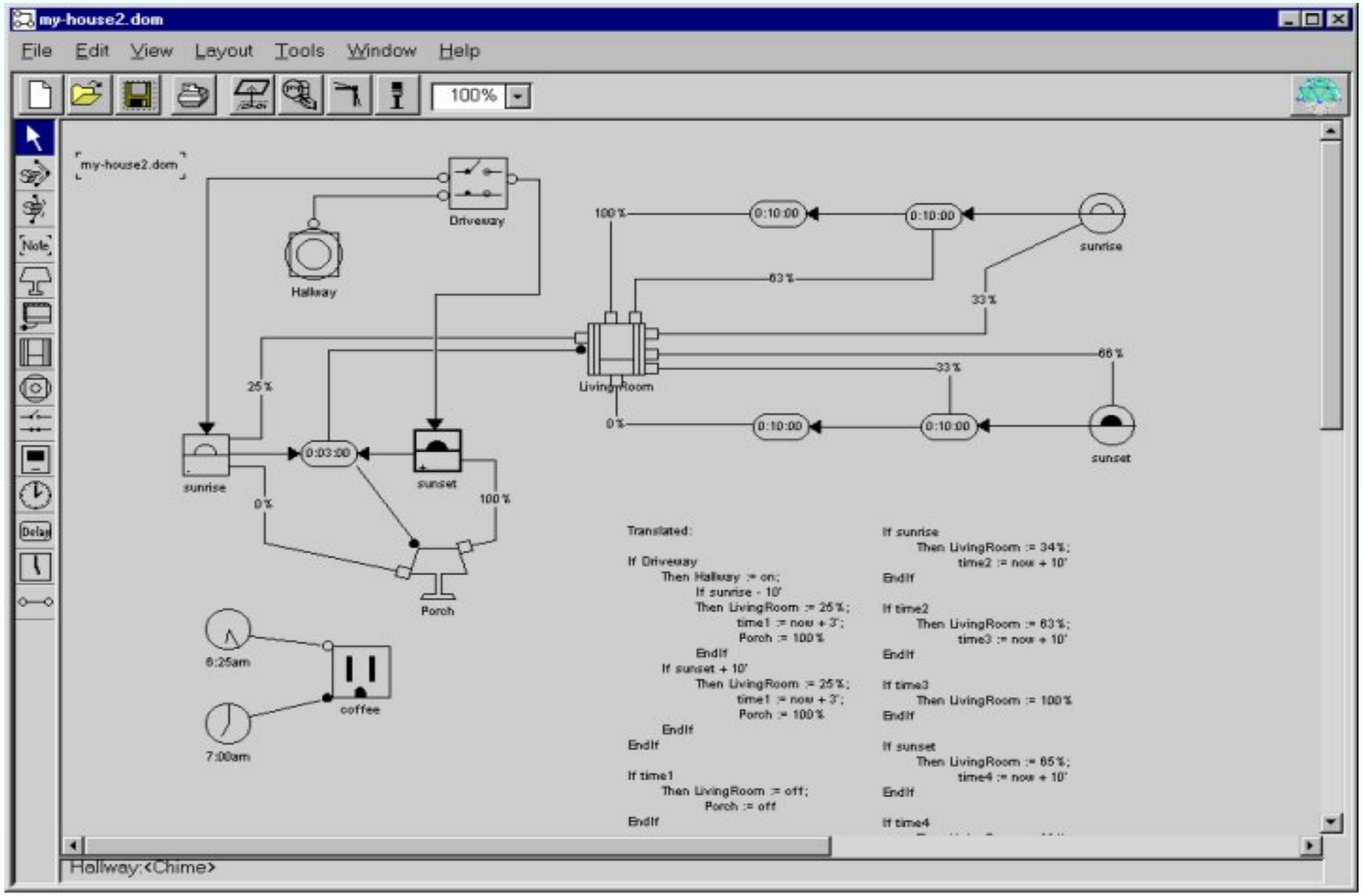
Name: Sunrise

Minutes: 10

Hours: 0

Reference: Sunrise

Criteria: before



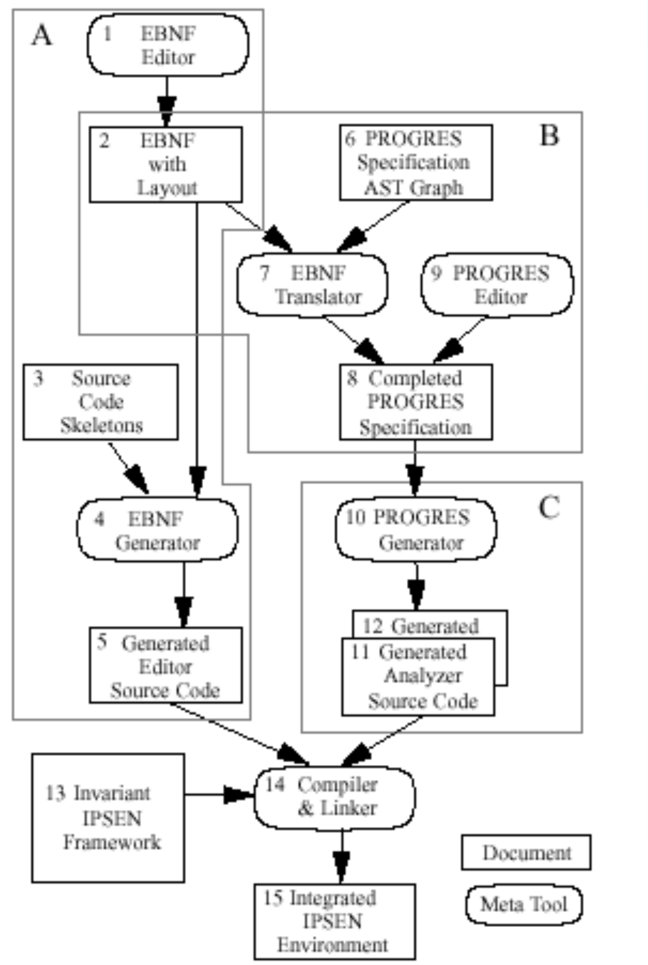
```
Translated:

If Driveway
    Then Hallway := on;
        If sunrise - 10'
        Then LivingRoom := 25%;
            time1 := now + 3';
            Porch := 100%
        EndIf
    If sunset + 10'
        Then LivingRoom := 25%;
            time1 := now + 3';
            Porch := 100%
    EndIf
EndIf

If time1
    Then LivingRoom := off;
        Porch := off
EndIf
```

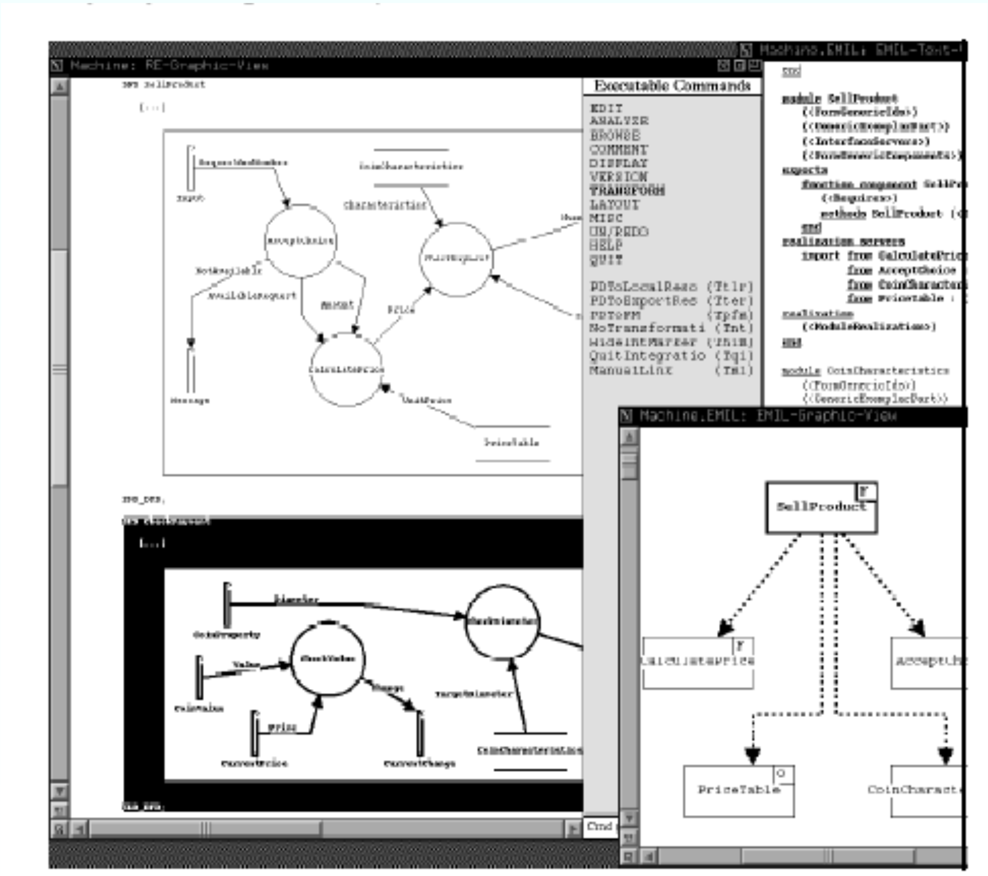
```
If sunrise
    Then LivingRoom := 34%;
        time2 := now + 10'
EndIf

If time2
    Then LivingRoom := 63%;
        time3 := now + 10'
EndIf

If time3
    Then LivingRoom := 100%
EndIf

If sunset
    Then LivingRoom := 65%;
        time4 := now + 10'
EndIf

If time4
```

IPSEN

- **Klein and Schurr, AAchen (Schurr now @ Darmstadt)**
  - See SEE' 97 paper
- **Quite different approach to the other tools**
  - Context free grammars used to specify syntax and layout of languages
  - Graph rewriting rules (PROGRESS) used for specifying semantics
  - Both mechanisms use textual specification to generate syntax directed visual editor

IPSEN


A
1 EBNF Editor
2 EBNF with Layout
B
6 PROGRES Specification AST Graph
7 EBNF Translator
9 PROGRES Editor
3 Source Code Skeletons
8 Completed PROGRES Specification
4 EBNF Generator
C
10 PROGRES Generator
12 Generated
11 Generated Analyzer Source Code
5 Generated Editor Source Code
13 Invariant IPSEN Framework
14 Compiler & Linker
15 Integrated IPSEN Environment
Document
Meta Tool



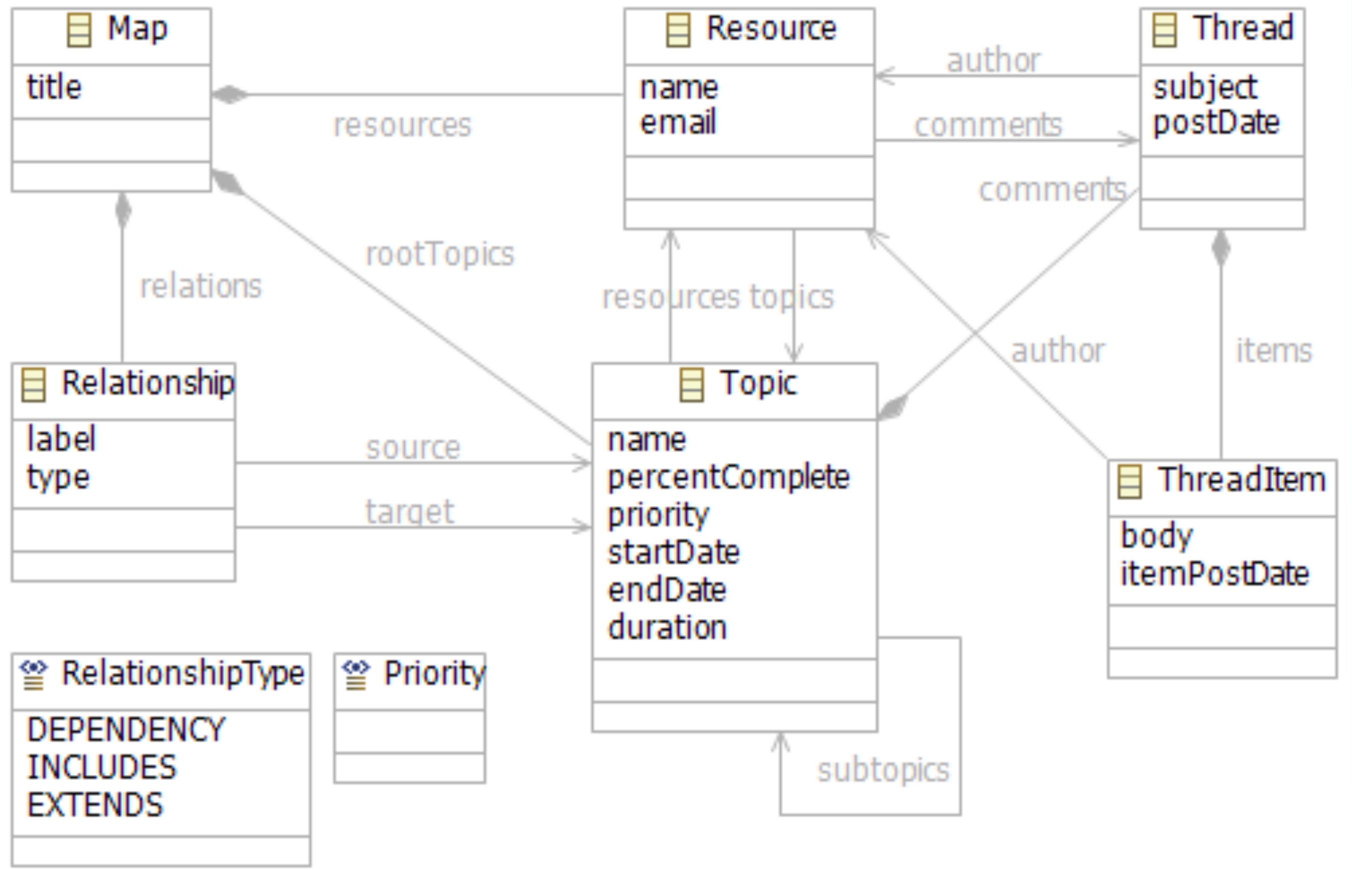
```
end

module SellProduct
   (<FormGenericIds>)
   (<GenericExemplarPart>)
   (<InterfaceServers>)
   (<FormGenericComponents>)
exports
   function component SellPr
      (<Requires>)
      methods SellProduct (<
   end
realization servers
   import from CalculatePric
          from AcceptChoice
          from CoinCharacteri
          from Pricetable :
realization
   (<ModuleRealization>)
end

module CoinCharacteristics
   (<FormGenericIds>)
   (<GenericExemplarPart>)
```

Eclipse GMF (Graphical Modelling Framework)

- **Framework for using the EMF (Eclipse Modelling Framework) and GEF (Graphical Editor Framework) to build graphical editors**
- **Proivides set of (currently basic) meta-tools to specify meta-model, graphical elements and mapping (“view type” specification)**
- **Generates EMF and GEF code to implement editors as Eclipse plug-ins**
- **Open source and free**

GMF – Domain model definition

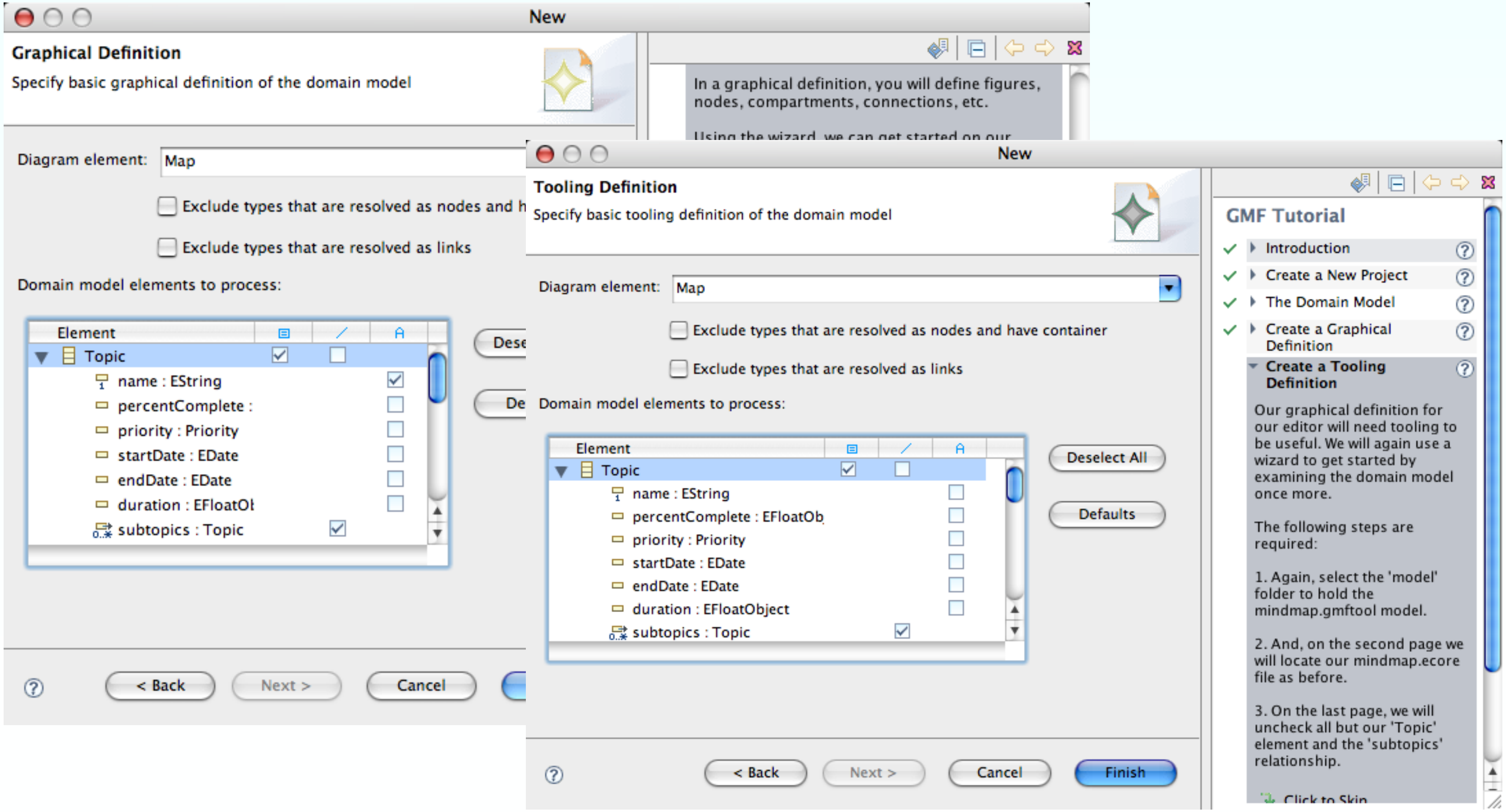
Import EMF model from class diagram, XSD file etc:



From: wiki.eclipse.org/index.php/GMF_Tutorial

GEF – graphical element definition

Use Wizards to specify appearance, tools for editor:



Graphical Definition

Specify basic graphical definition of the domain model

In a graphical definition, you will define figures, nodes, compartments, connections, etc.

Using the wizard, we can get started on our

Diagram element: Map

☐ Exclude types that are resolved as nodes and h

☐ Exclude types that are resolved as links

Domain model elements to process:

Element			A
Topic	☑	☐	
name : EString			☑
percentComplete :			☐
priority : Priority			☐
startDate : EDate			☐
endDate : EDate			☐
duration : EFloatOb			☐
subtopics : Topic		☑	

Tooling Definition

Specify basic tooling definition of the domain model

Diagram element: Map

☐ Exclude types that are resolved as nodes and have container

☐ Exclude types that are resolved as links

Domain model elements to process:

Element			A
Topic	☑	☐	
name : EString			☐
percentComplete : EFloatOb			☐
priority : Priority			☐
startDate : EDate			☐
endDate : EDate			☐
duration : EFloatObject			☐
subtopics : Topic		☑	

GMF Tutorial

- ✓ Introduction
- ✓ Create a New Project
- ✓ The Domain Model
- ✓ Create a Graphical Definition
- **Create a Tooling Definition**

Our graphical definition for our editor will need tooling to be useful. We will again use a wizard to get started by examining the domain model once more.

The following steps are required:

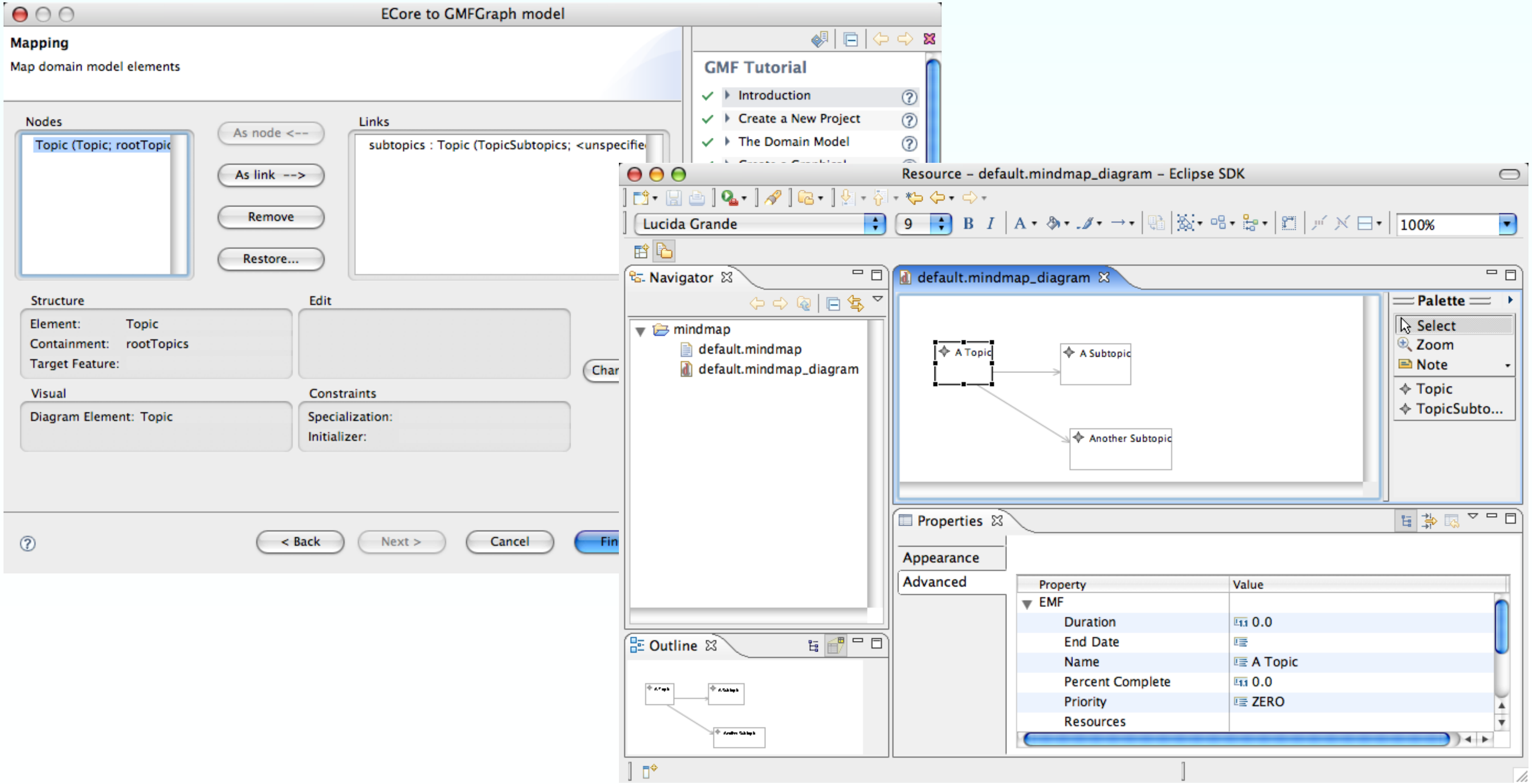
1. Again, select the 'model' folder to hold the mindmap.gmftool model.

2. And, on the second page we will locate our mindmap.ecore file as before.

3. On the last page, we will uncheck all but our 'Topic' element and the 'subtopics' relationship.

GEF – mapping & example tool

Specify model<->graphical mapping via Wizard:



ECore to GMFGraph model

Mapping

Map domain model elements

Nodes

Topic (Topic; rootTopic

Links

subtopics : Topic (TopicSubtopics; <unspecifie

Structure

Element: Topic
Containment: rootTopics
Target Feature:

Visual

Diagram Element: Topic

Constraints

Specialization:
Initializer:

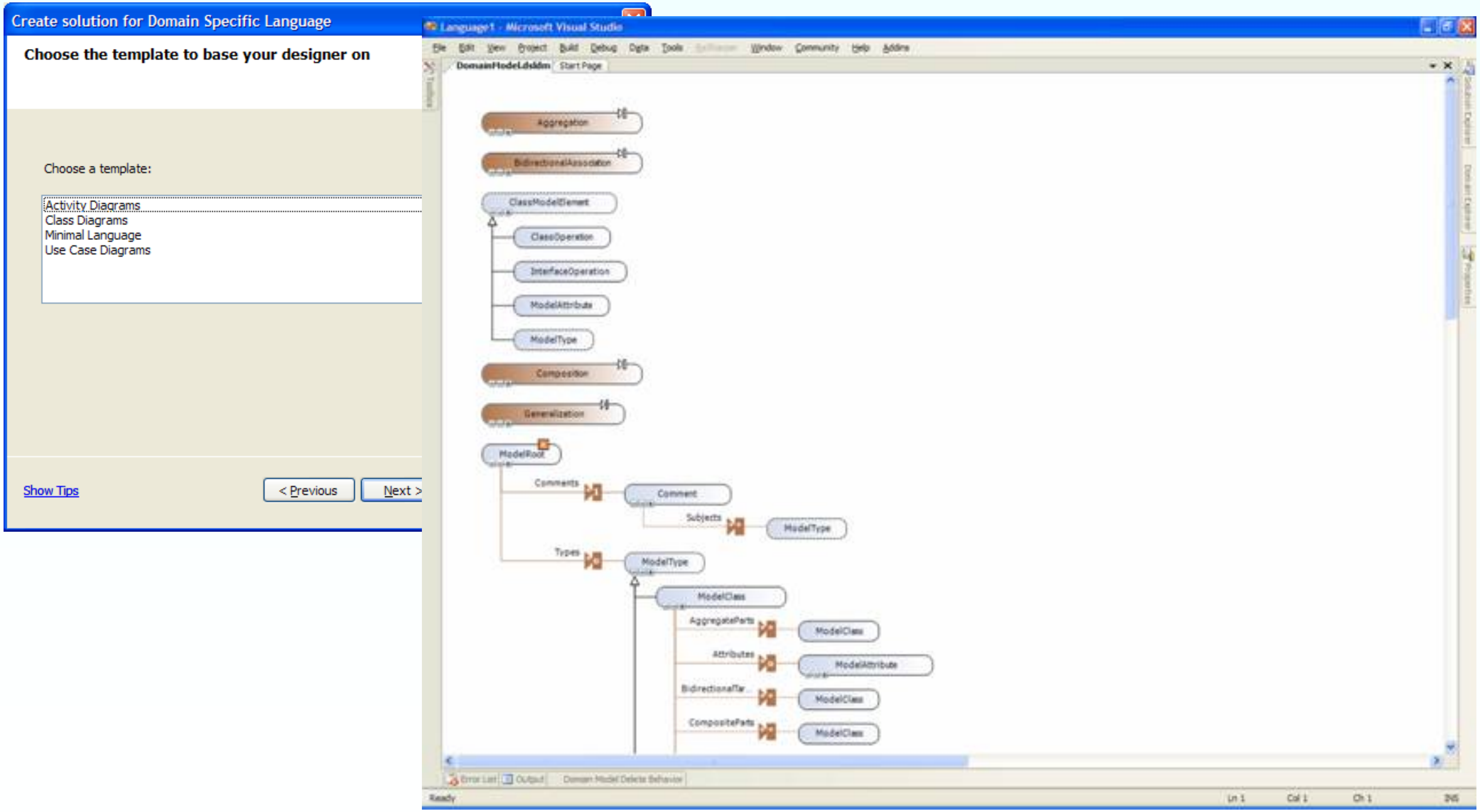
Resource – default.mindmap_diagram – Eclipse SDK

Property	Value
EMF	
Duration	0.0
End Date	
Name	A Topic
Percent Complete	0.0
Priority	ZERO
Resources	

Microsoft VisualStudio 2005 DSL Tools

- **Set of frameworks to build model-driven engineering tools with DSVLs for VS 2005**
- **Provides meta-modelling based on UML meta-model extensions**
- **Provides diagram editors via XML configuration files**
- **Generates code for VS 2005 SDK – plug-ins to VS 2005 to produce UML models for model-driven engineering**
- **Models transformed to code in VS 2005 and code can be further enhanced**

VS2005 DSL – model definition



Create solution for Domain Specific Language

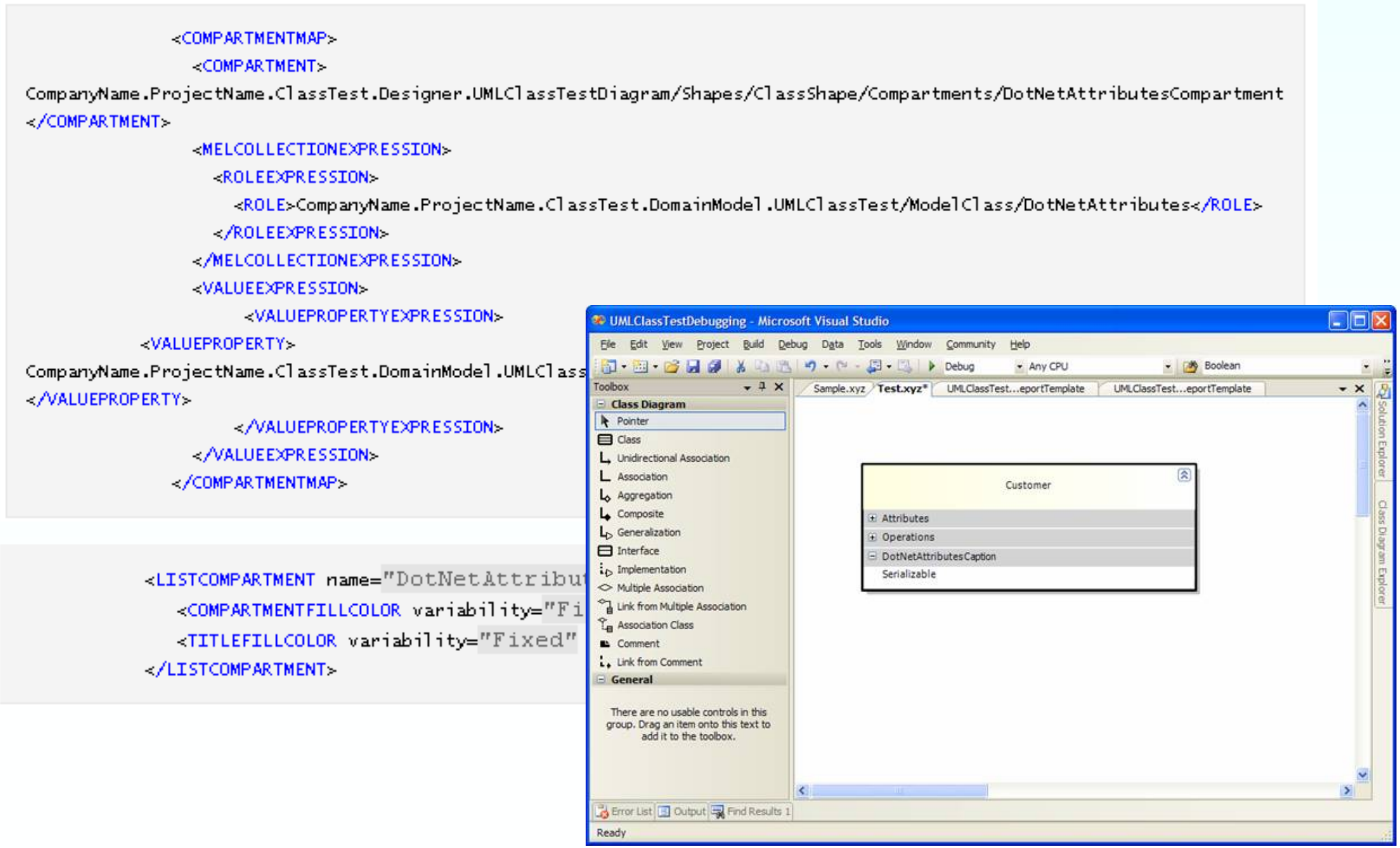
Choose the template to base your designer on

Choose a template:

- Activity Diagrams
- Class Diagrams
- Minimal Language
- Use Case Diagrams

From: http://www.developerland.com/DotNet/Design/444.aspx

VS 2005 DSL – Diagram definition



```
                <COMPARTMENTMAP>
                  <COMPARTMENT>
CompanyName.ProjectName.ClassTest.Designer.UMLClassTestDiagram/Shapes/ClassShape/Compartments/DotNetAttributesCompartment
</COMPARTMENT>
                  <MELCOLLECTIONEXPRESSION>
                    <ROLEEXPRESSION>
                      <ROLE>CompanyName.ProjectName.ClassTest.DomainModel.UMLClassTest/ModelClass/DotNetAttributes</ROLE>
                    </ROLEEXPRESSION>
                  </MELCOLLECTIONEXPRESSION>
                  <VALUEEXPRESSION>
                       <VALUEPROPERTYEXPRESSION>
              <VALUEPROPERTY>
CompanyName.ProjectName.ClassTest.DomainModel.UMLClass
</VALUEPROPERTY>
                      </VALUEPROPERTYEXPRESSION>
                  </VALUEEXPRESSION>
                </COMPARTMENTMAP>
```

```
        <LISTCOMPARTMENT name="DotNetAttribu
           <COMPARTMENTFILLCOLOR variability="Fi
           <TITLEFILLCOLOR variability="Fixed"
        </LISTCOMPARTMENT>
```

Comparison

Tool	MetaModel Paradigm	Meta Model Specn	Visual Elmt Specfn	Behaviour Specfn
MetaEdit+	Unkown (MetaEdit was MOF)	Tabular/ Form based	Symbol Editor	Constraints
GME	OO based on MetaGME	Visual – several editors	Bitmaps, simple shapes	OCL constraints
IPSEN	EBNF and graph grammars	Text	EBNF	Graph Grammars
DOME	Object Model	ProtoDome	ProtoDome	Visual & textual scripting
Pounamu	Entity Relationship	Visual (currently limited)	Shape & Connector tools	Event handlers
Eclipse GMF	EMF (EMOF)	UML, XSD, code import	Shape, relationship Container	Code using EMF, GEF APIs
MS VS 2005 DSL	UML meta-model	UML	Shape, relationship Container	Simple constraints; code using VS SDK APIs

Comparison

Tool	Storage	Code gen support	Integration API	Multi paradigm
MetaEdit+	Custom DB	Custom scripting language	SOAP	Partially
GME	Variety - customisable	Model interpreters	COM interfaces	Yes, aspects
IPSEN	Graph based database	Graph grammars	Unknown	No
DOME	Custom	Extensive	Custom – has plug ins	Yes
Pounamu	XML files	XML tools	SOAP, RMI	Yes, view definer
Eclipse GMF	XMI, XML	EMF JET; GME ALT	Eclipse APIs	Yes, view definer
MS VS 2005 DSL	XML	VS 2005 MDE tools	VS 2005 SDK APIs	Separate DSL tools

Comparison

Tool	Multiuser tools	Liveness	Portability	Thin client support	Cost
MetaEdit+	Yes	Yes	Multi-platform	No	High
GME	Unclear	Versioning support	Java based	No	Free
IPSEN	No	No- compile cycle	No	No	Free
DOME	No	Yes, good support	Multi-platform	No	Free GNU
Pounamu	Yes for generated tools	Yes, some bugs!	Java based	Yes	Free for ac use
Eclipse GMF	No	No – code generation to GEF, EMF code	Java based	No	Free
MS VS 2005 DSL	Via VS 2005 SDK Support	No – code generation to SDK APIs	Theoretically any .NET platform	No	Moderate

Wrap up

References

- A. Blackwell, M. Burnett & S. Peyton Jones, Champagne Prototyping: a research technique for early evaluation of complex end user programming systems, IEEE VL/HCC, 2004, 47-54
- Cognitive Dimensions website http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions
- T.R. Green and M. Petre, Usability analysis of visual programming environments: a 'cognitive dimensions' framework, Journal of Visual Languages and Computing, (7), 1996, 131-174.
- Labview, http://www.ni.com/labview/
- D. Roberts, R.Johnson "Evolving Frameworks" http://st-www.cs.uiuc.edu /users/droberts/evolve.html
- Jimi Hendrix kit: http://guitargeek.com/rigview/354/
- MS Visual Studio DSL tools example: http://www.developerland.com/DotNet/Design/444.aspx
- Eclipse GMF tutorial: wiki.eclipse.org/index.php/GMF_Tutorial
- Zhu, N., Grundy, J.C., Hosking, J.G., Liu, N., Cao, S. and Mehra, A. Pounamu: a meta-tool for exploratory domain-specific visual language tool development, Journal of Systems and Software, Elsevier, vol. 80, no. 8, pp 1390-1407.
- Gundy, J.C., Hosking, J.G., Zhu, N. and Liu, N. Generating Domain-Specific Visual Language Editors from High-level Tool Specifications, In Proceedings of the 2006 IEEE/ACM International Conference on Automated Software Engineering, Tokyo, 24-28 Sept 2006, IEEE.
- Stoeckle, H., Grundy, J.C. and Hosking, J.G. A Framework for Visual Notation Exchange, Journal of Visual Languages and Computing, Volume 16, Issue 3 , June 2005, Elsevier, pp.187-212.
- Zhu, N., Grundy, J.C. and Hosking, J.G. Constructing domain-specific design tools with a visual language meta-tool, CAiSE 2005 Forum, Portugul, June 2005, Springer.