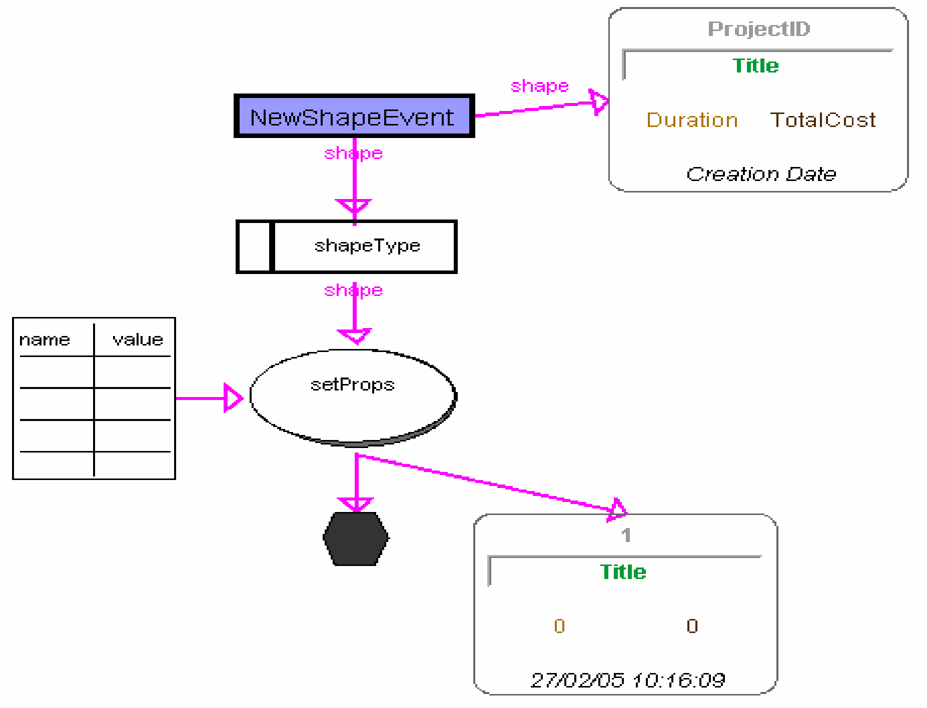
Marama Extensions

- *Aim of section:*
  - Look at work undertaken to extend Marama core features (recent and current)
  - Problems being addressed and solutions adopted
- Contents
  - Behaviour specification
    - Formulae
    - DSVL for event handling
  - Back end code/model import/export
  - Collaboration/awareness
  - Thin-client diagramming
  - Sketching-based input

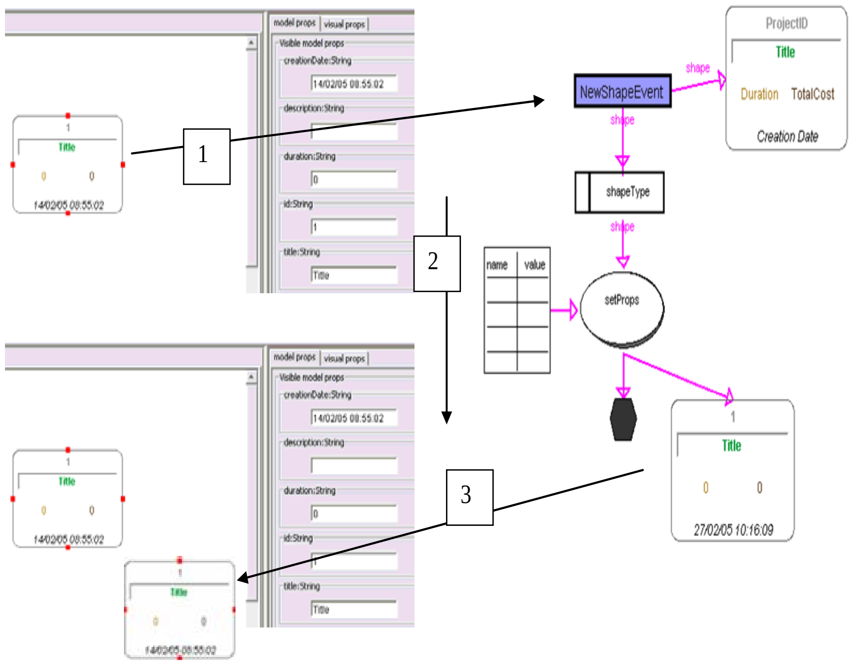
Behaviour specification

- Problems
  - Original event handler specification approach required sophisticated user
    - Understanding of Java
    - Familiarity with Marama API
  - Difficult to debug
- Solutions
  - Kaitiaki visual event handler specification tool (Karen Liu PhD)
    - Aimed at handlers for view manipulation
    - Debug view
  - Metamodel constraint language
    - Like OCL for specifying computations at meta model level (like spreadsheets at a type level)
- Status
  - Both projects completed by Karen Liu (PhD)
  - Formulae added to Marama meta-tools, Kaitiaki to come…

Kaitiaki

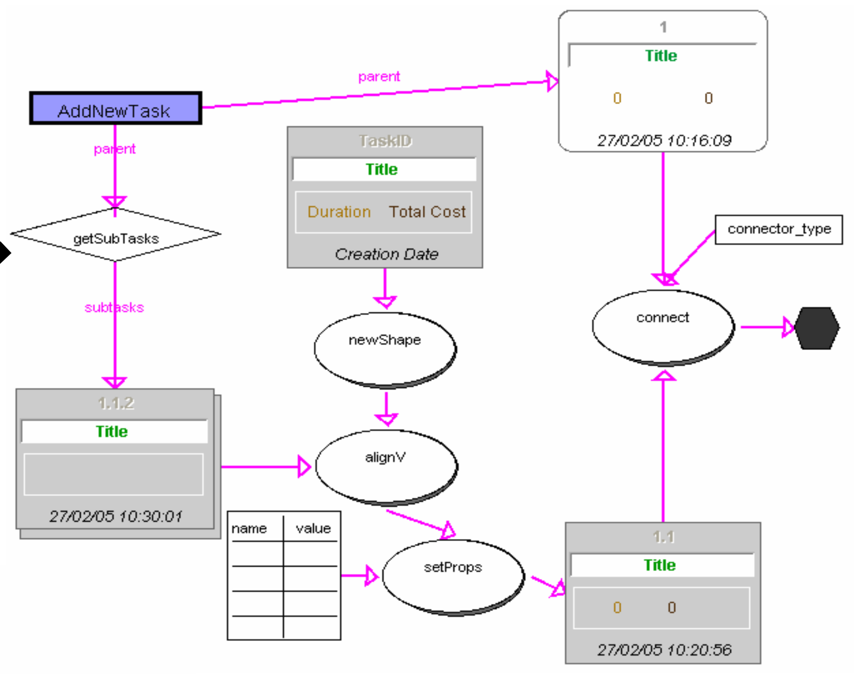


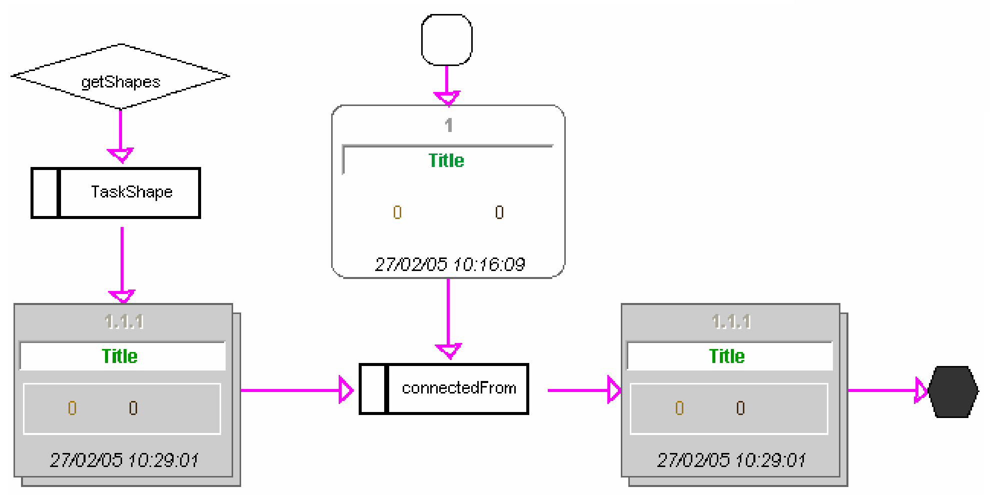
- Dataflow metaphor, but includes data push and pull
- Includes shape representations to give clarity



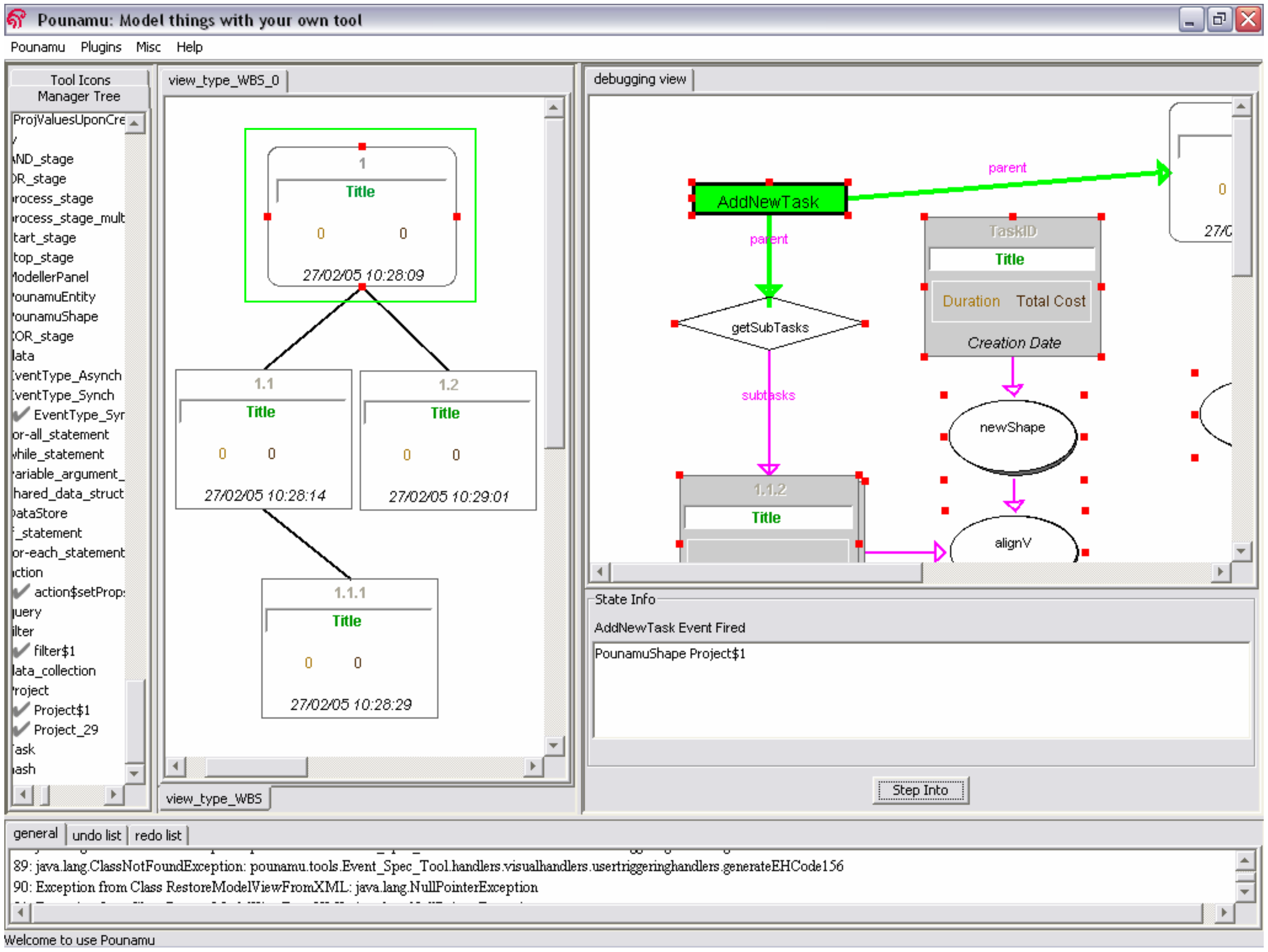
More complex example

- Parameterised subtasks
- Creation of new objects or connectors
- Alignment etc





Kaitiaki Debug view



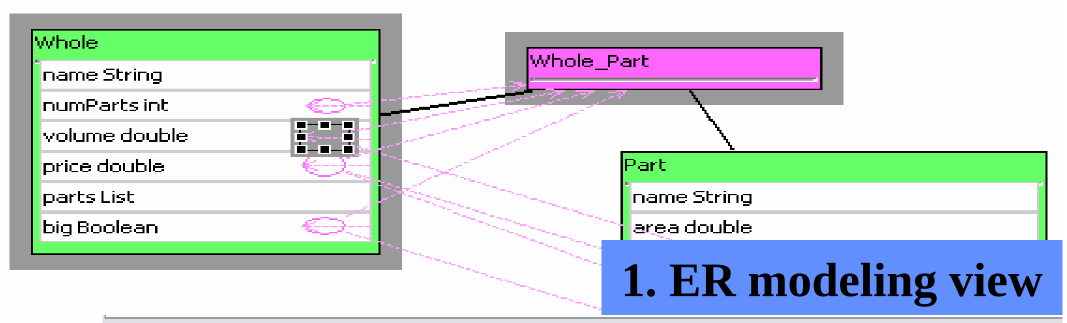
AddNewTask Event Fired

PounamuShape Project$1

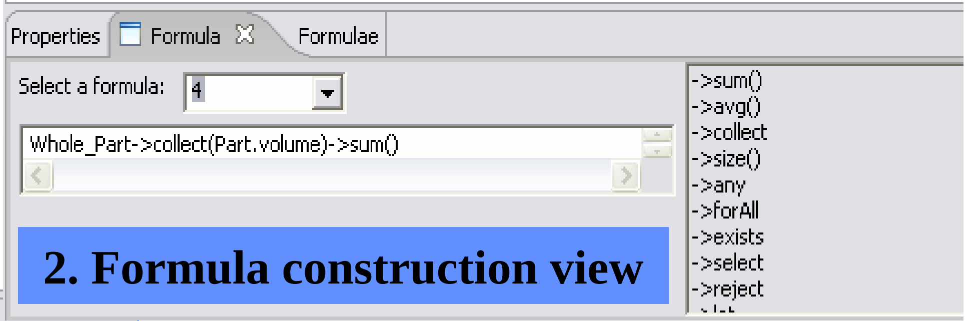
```
89: java.lang.ClassNotFoundException: pounamu.tools.Event_Spec_Tool.handlers.visualhandlers.usertriggeringhandlers.generateEHCode156
90: Exception from Class RestoreModelViewFromXML: java.lang.NullPointerException
```

Formula definer for meta-model

- Design-time
  - ER modeling view
  - Formula construction view
  - Formulae view
- Run-time
  - Instance view
  - Master-details tabular view with an on-demand tooltip showing the underlying formula



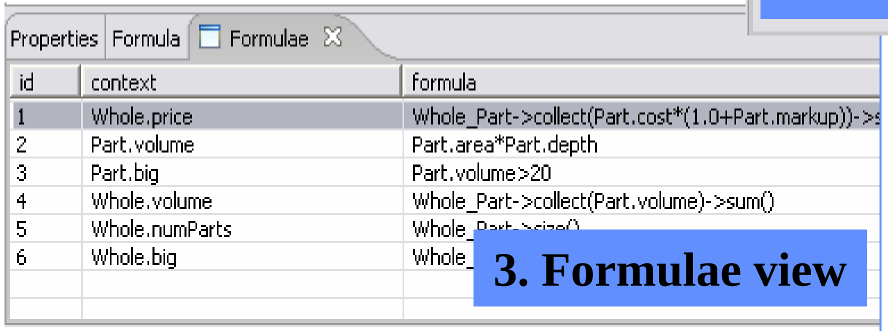
1. ER modeling view



Select a formula: 4

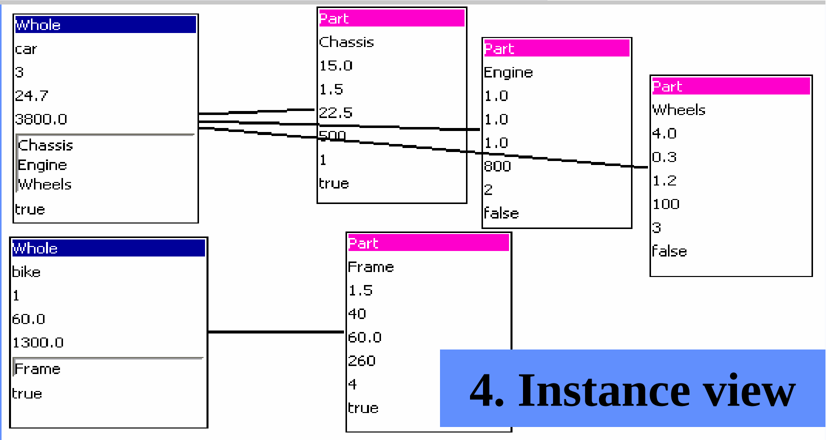
```
Whole_Part->collect(Part.volume)->sum()
```

2. Formula construction view

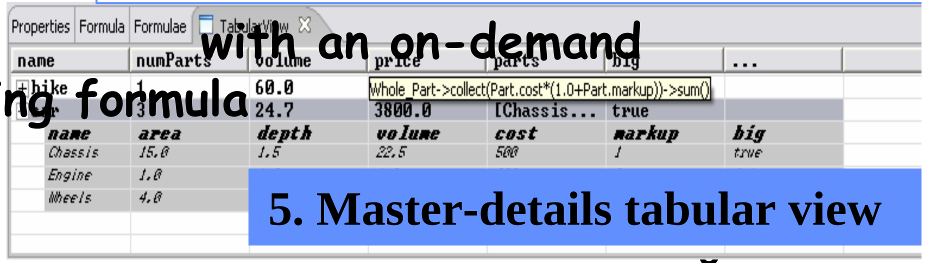


id	context	formula
1	Whole.price	Whole_Part->collect(Part.cost*(1.0+Part.markup))->s
2	Part.volume	Part.area*Part.depth
3	Part.big	Part.volume>20
4	Whole.volume	Whole_Part->collect(Part.volume)->sum()
5	Whole.numParts	Whole_Part->size()
6	Whole.big	Whole_

3. Formulae view



4. Instance view



name	numParts	volume	price	parts	big	...
bike	1	60.0				
car	3	24.7	3800.0	[Chassis...	true	

name	area	depth	volume	cost	markup	big
Chassis	15.0	1.5	22.5	500	1	true
Engine	1.0					
Wheels	4.0					

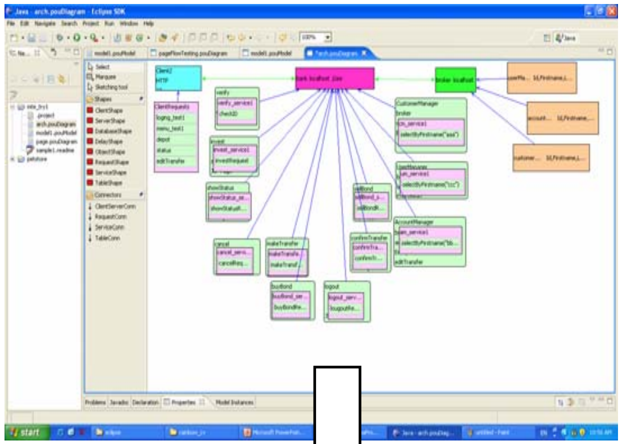
Whole_Part->collect(Part.cost*(1.0+Part.markup))->sum()

5. Master-details tabular view

Back end code import/export

- Problem
  - Backend code generation and code import facilities require bespoke code for each generator/importer
- Solutions
  - Event handlers to walk EMF data structures & generate code OR create/modify EMF structures from parsed code
  - Used JET (Eclipse EMF) template-based code generator
  - Developed MaramaVMLPlus XSLT generator for complex data transformation
- Status
  - MaramaVMLPlus tool developed by Jun Huh
  - Being integrated into Marama meta-tools

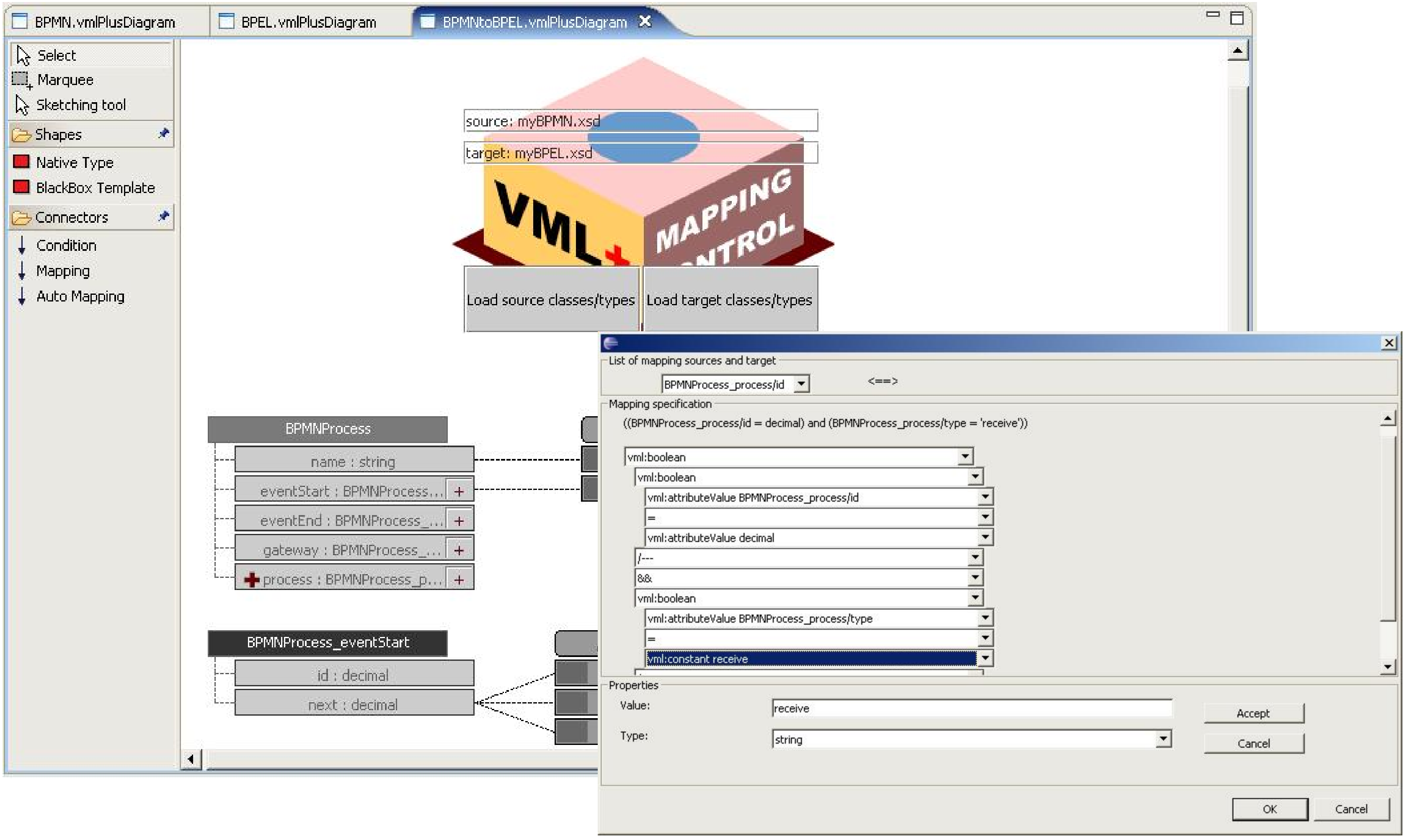
Code Export



```
<%@ jet package="nz.ac.auckland.cs.marama.userdirectory.tools.MaramaMTE.codegen"
imports="nz.ac.auckland.cs.marama.model.project.* java.util.* " class="BasicClientGen" %>

 <% MaramaEntity client = (MaramaEntity) argument; %>

 <%
 String className = (String) client.getAttributeValue("name");
 String threads = client.getAttributeValueAsString("threads");
 if(threads == null)
 threads = "1";
 List services = client.getParentEntities("Services");

 %>

 import java.rmi.Naming;
 import java.util.List;
 import java.util.ArrayList;

 public class <%=className%>
 {

  // declare remote objects
 <% for (int j=0; j < services.size(); j++) { %>
 <% MaramaEntity service = (MaramaEntity) services.get(j); %>
 <% String serviceName = service.getAttributeValueAsString("name"); %>
 <% List requests = service.getParentEntities("Requests"); %>
<% for (int i=0; i < requests.size(); i++ ) { %>
 <% MaramaEntity request = (MaramaEntity)requests.get(i); %>
 <% if(request.getAttributeValueAsString("remoteObject") != null) { %>
 public static <%=request.getAttributeValue("remoteObject")%>
<%=serviceName%>_<%=request.getAttributeValue("remoteObject")%>_<%=i%>;
<% } %>
 <% } %>
 <% } %>


...
```

```
import java.rmi.Naming;
import java.util.List;
import java.util.ArrayList;

public class Client1
{
// declare remote objects
                     public static CustomerManager clientTest1_CustomerManager_0;
                     public static CustomerManager clientTest1_CustomerManager_2;
 ...


public static void main(String args[])
{
                     // threads = 10

                     // look up remote objects...
                     try {

 clientTest1_CustomerManager_0 = (CustomerManager)
                     // need to put host name in...!

 clientTest1_CustomerManager_2 = (CustomerManager)
                     // need to put host name in...!
                     Naming.lookup("localhost/CustomerManager");

 ClientTest2_UserManager_1 = (UserManager)
                     // need to put host name in...!
                     Naming.lookup("localhost/UserManager");
ClientTest2_CustomerManager_2 = (CustomerManager)
                     // need to put host name in...!
                     Naming.lookup("localhost/CustomerManager");

...
}

                     // start the client threads & wait until they have all have finished...

                     for(int i=0; i < 10; i++) {
                                          Thread thread = (Thread) threads.get(i);
                                          thread.start();
                     }

                     long startTime = System.currentTimeMillis();

                     // wait on the client threads to finish

                     for(int i=0; i < 10; i++) {
                                          Client1Thread thread = (Client1Thread) threads.get(i);
                                          thread.doWait();
                     }

                     long endTime = System.currentTimeMillis();

                     System.out.println("Time taken = "+(endTime-startTime));
```

MaramaVMLPlus



source: myBPMN.xsd

target: myBPEL.xsd

List of mapping sources and target

BPMNProcess_process/id <==>

Mapping specification

((BPMNProcess_process/id = decimal) and (BPMNProcess_process/type = 'receive'))

```
vml:boolean
  vml:boolean
    vml:attributeValue BPMNProcess_process/id
    =
    vml:attributeValue decimal
  /---
  &&
  vml:boolean
    vml:attributeValue BPMNProcess_process/type
    =
    vml:constant receive
```

Properties

Value: receive

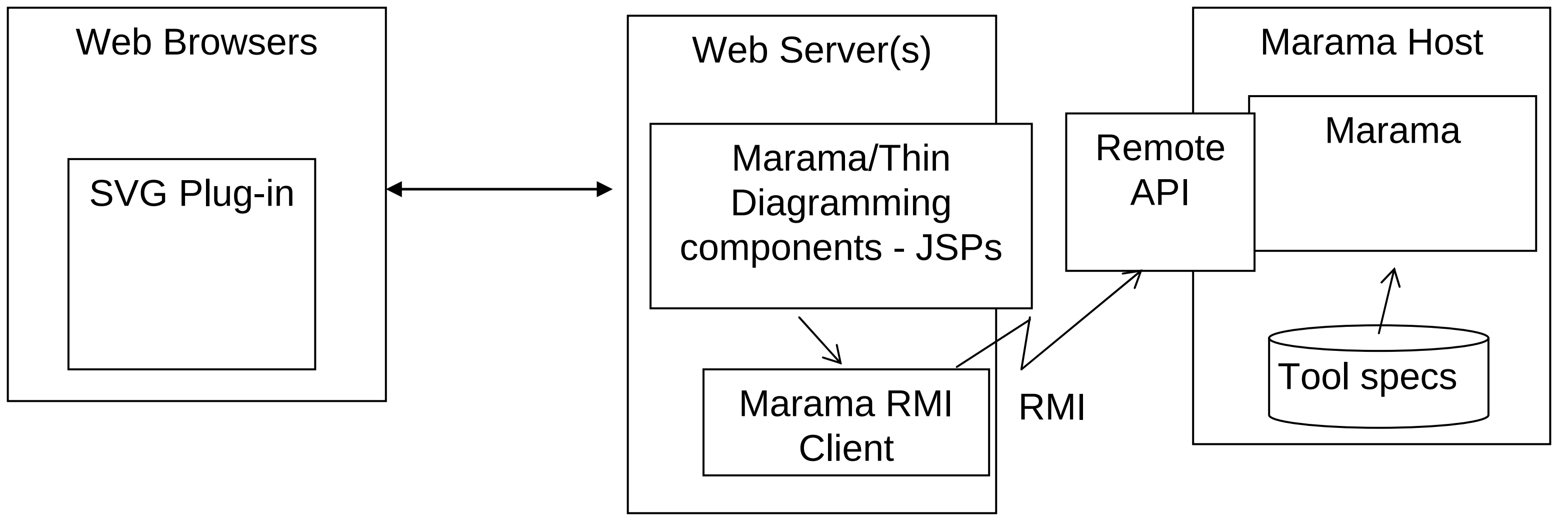
Type: string

Thin-client/Remote interfaces

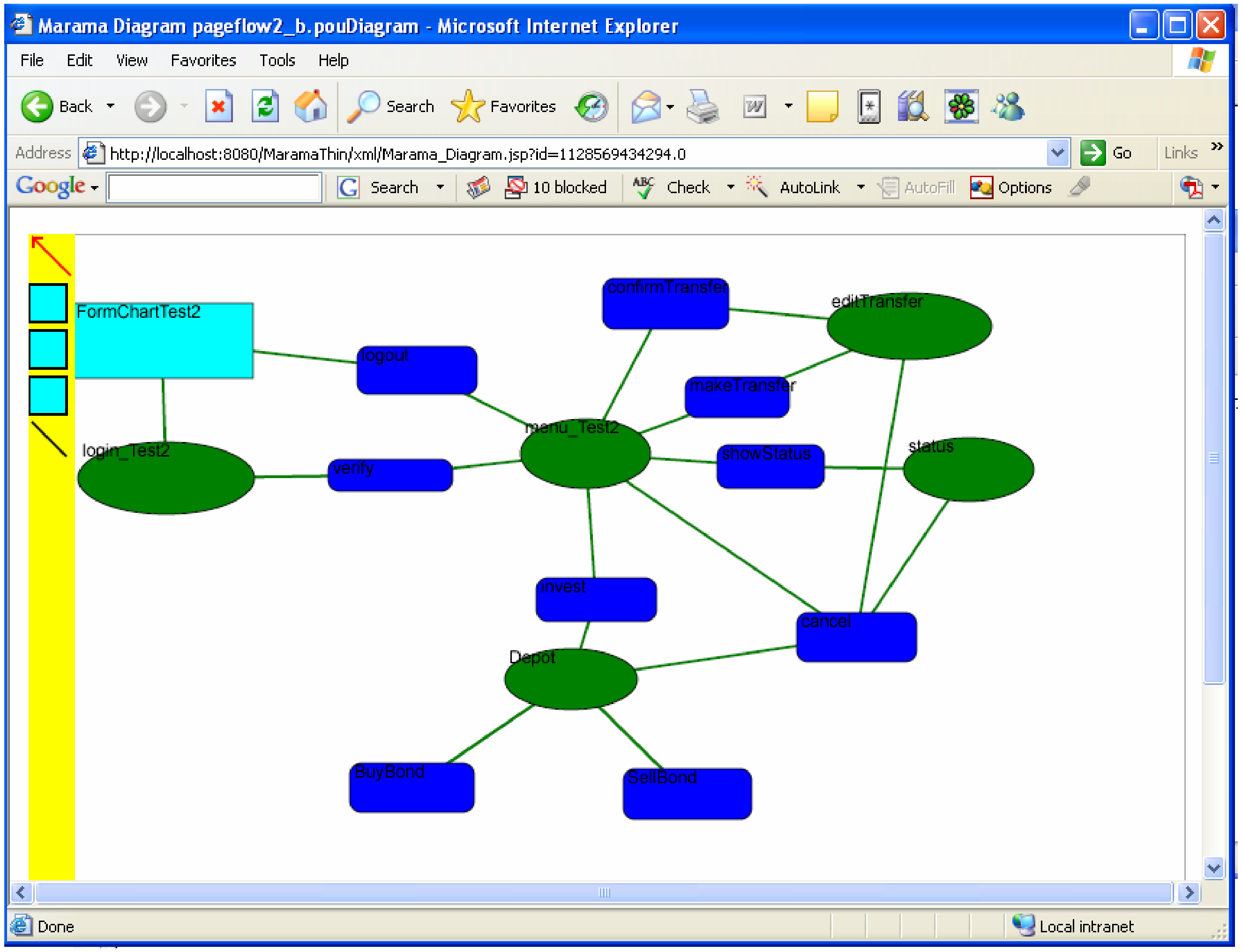
- Problems
  - Need to access Marama tools remotely on a variety of different devices
  - Need to drive Marama remotely
- Solutions
  - RMI interface to Marama API
  - Thin client interface for web browser interaction with any Marama generated tool (Penny Cao MSc thesis done)
  - Mobile phone interface for Marama generated tools (Joe Zhao MSc thesis done)
  - Generalise framework and add VRML interface (Joe Zhao done)
  - Add games engine interface (Mek Bhumiwat & Joseph Shi 2005 SE Part 4 project done)

Thin client interface

- Originally developed by Penny Cao (MSc thesis) for Pounamu
- New version developed for Marama by John G
  - Uses RMI API to generate SVG version of Marama model views
  - Can interact with these to perform editing actions
  - Support multi-user interaction with Marama tools

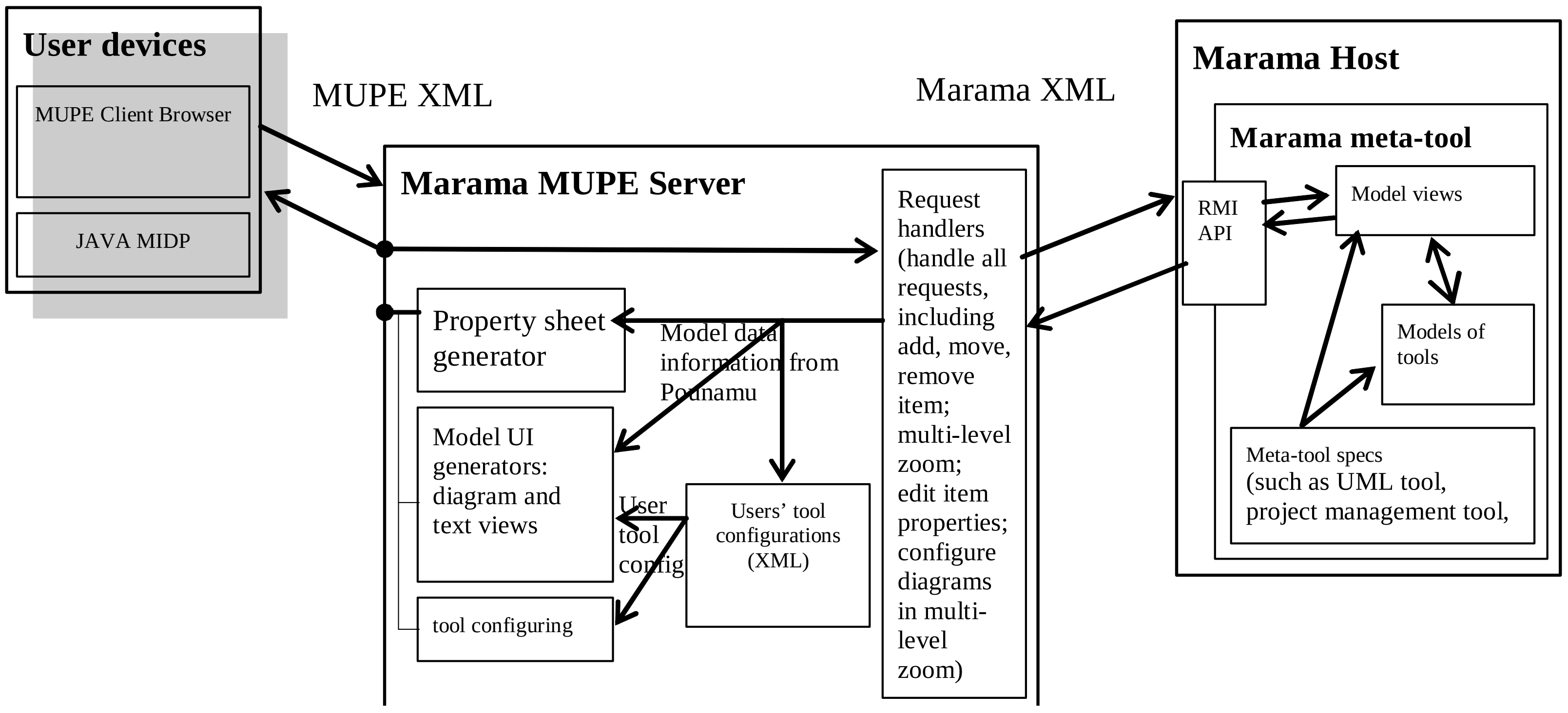


Thin client interface example

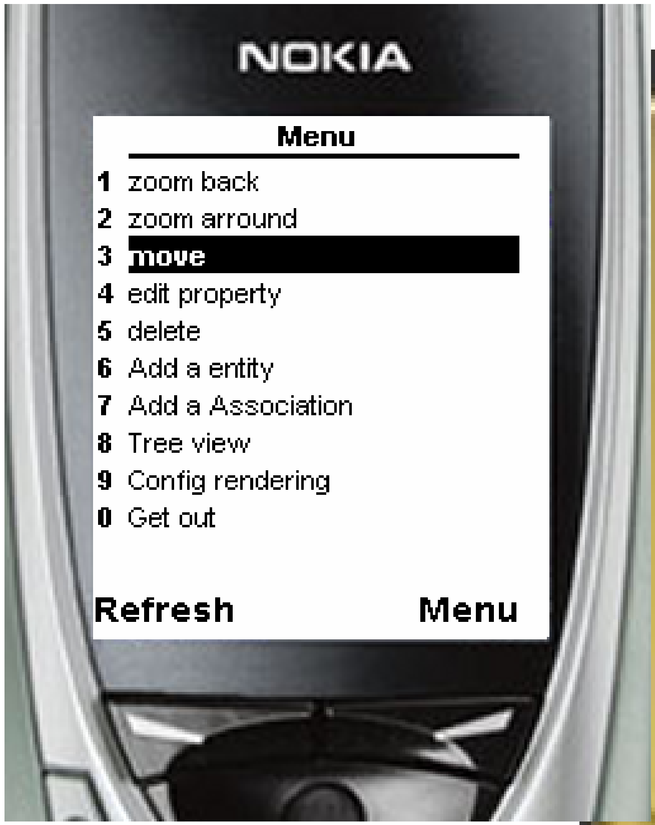


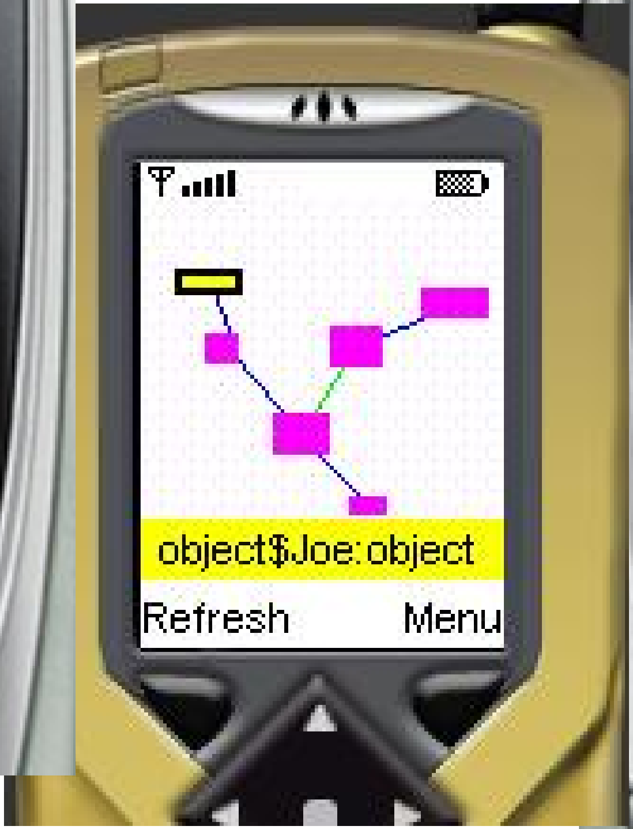
MUPE interface

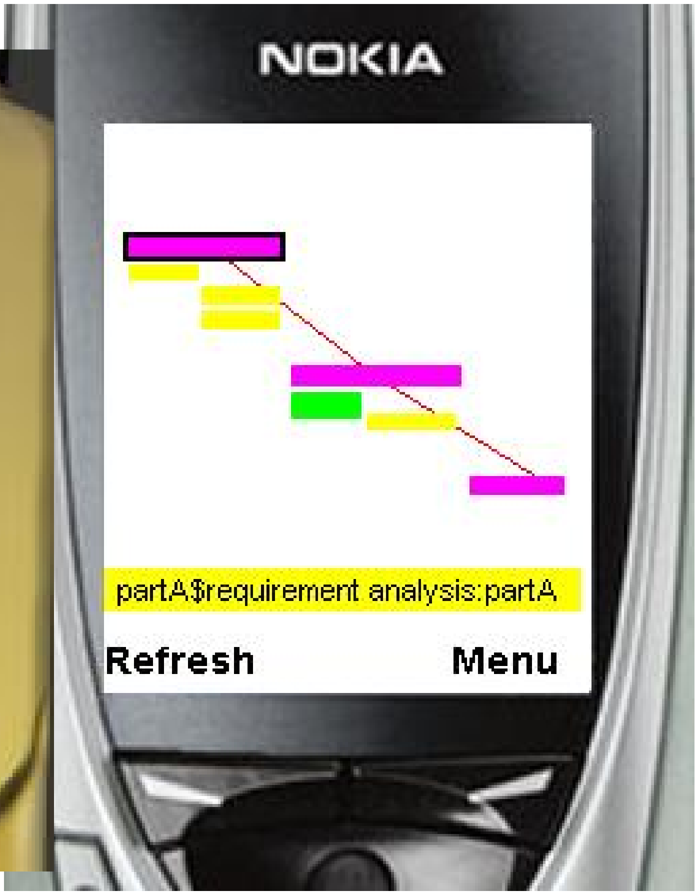
- Support for viewing and editing Pounamu & Marama tool views on cellphones
- Uses Nokia's MUPE open source mobile collaboration server plus MUPE client on phone
- Has several features for semantic zooming to allow diagrams to be sensibly visualised/edited on small screen


User devices
MUPE Client Browser
JAVA MIDP
MUPE XML
Marama XML
Marama MUPE Server
Request handlers (handle all requests, including add, move, remove item; multi-level zoom; edit item properties; configure diagrams in multi-level zoom)
Property sheet generator
Model data information from Pounamu
Model UI generators: diagram and text views
User tool config
Users' tool configurations (XML)
tool configuring
Marama Host
Marama meta-tool
RMI API
Model views
Models of tools
Meta-tool specs (such as UML tool, project management tool,

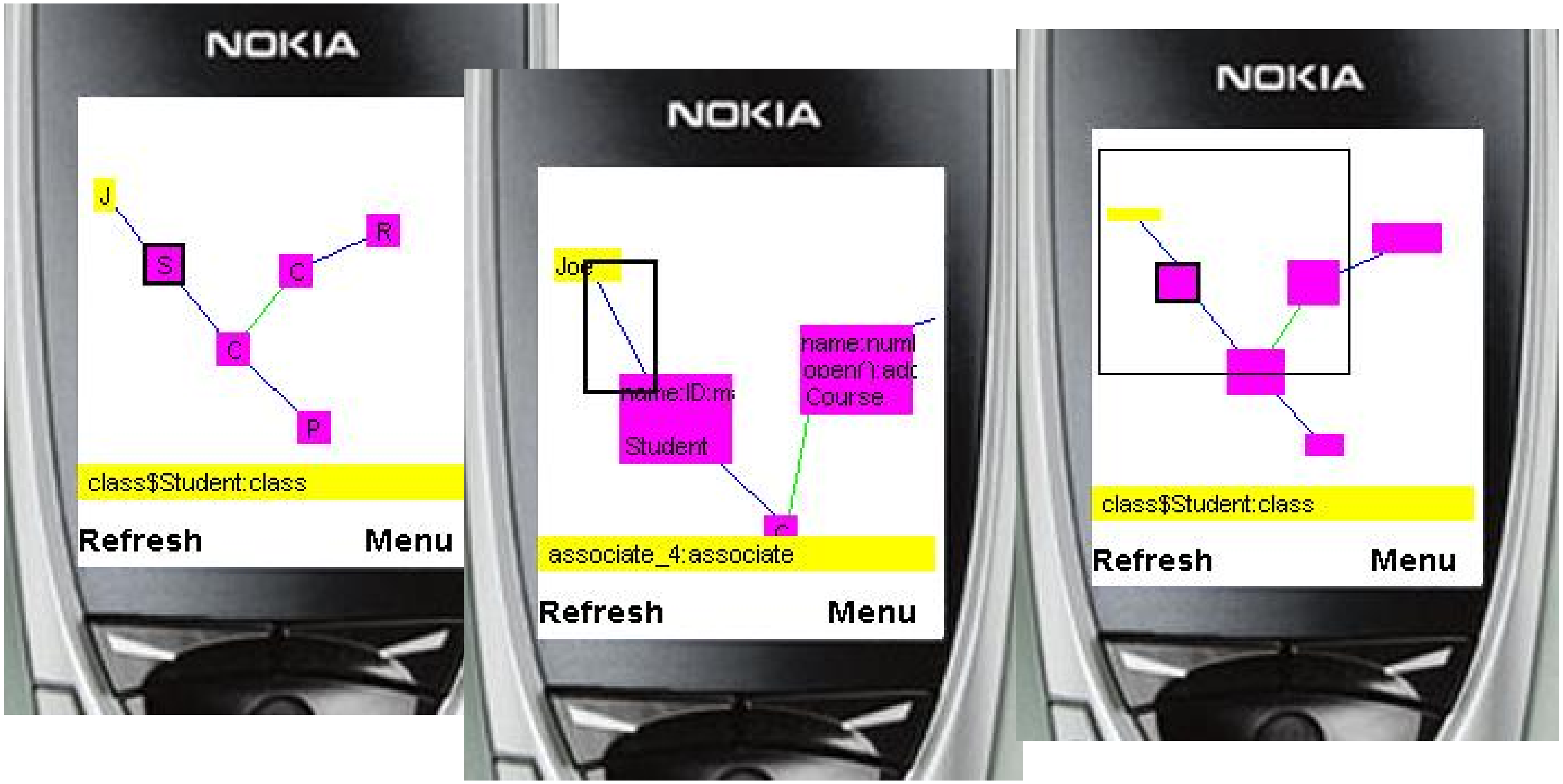
Example MUPE interface usage







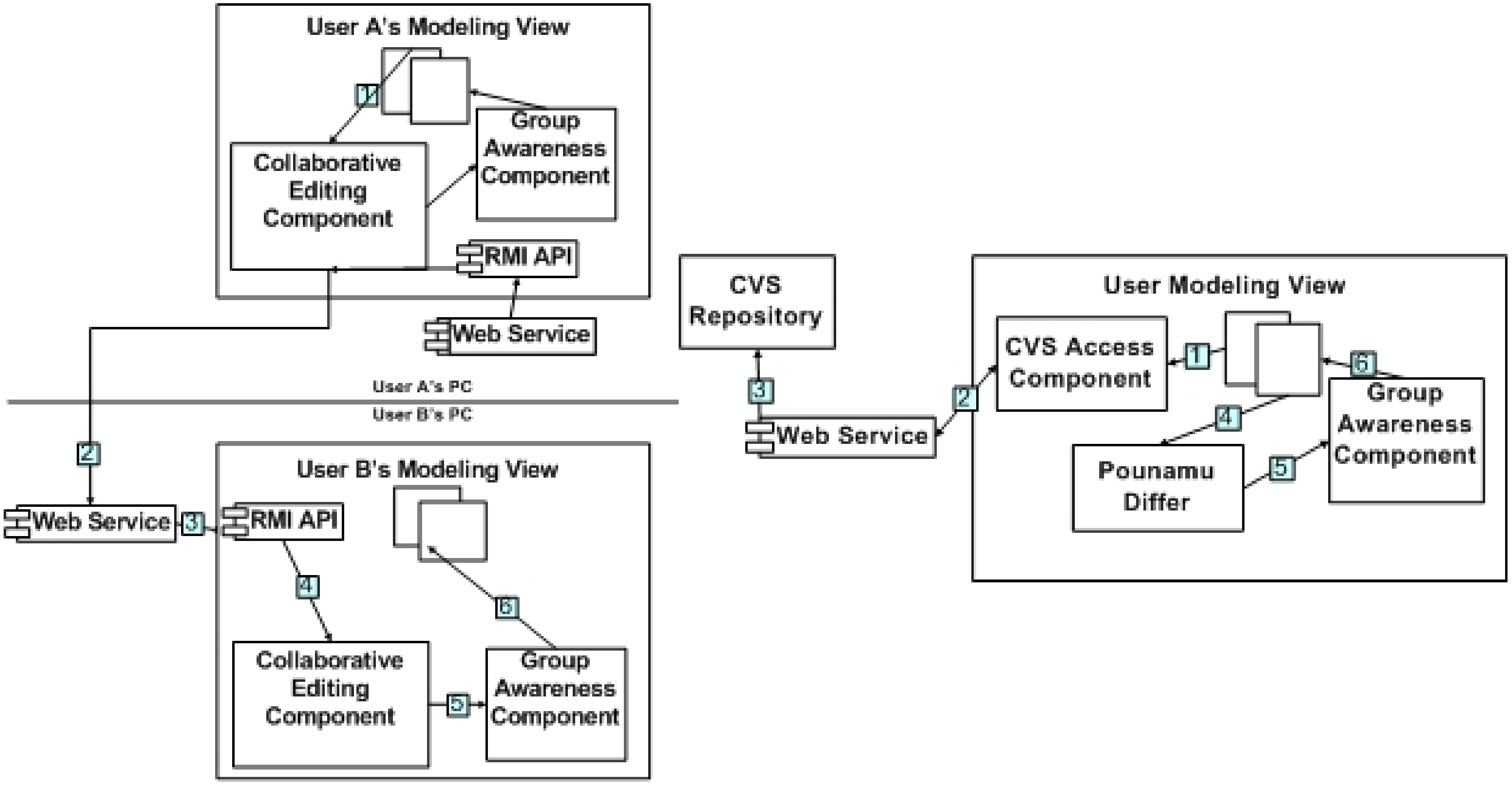
Element zooming and overview



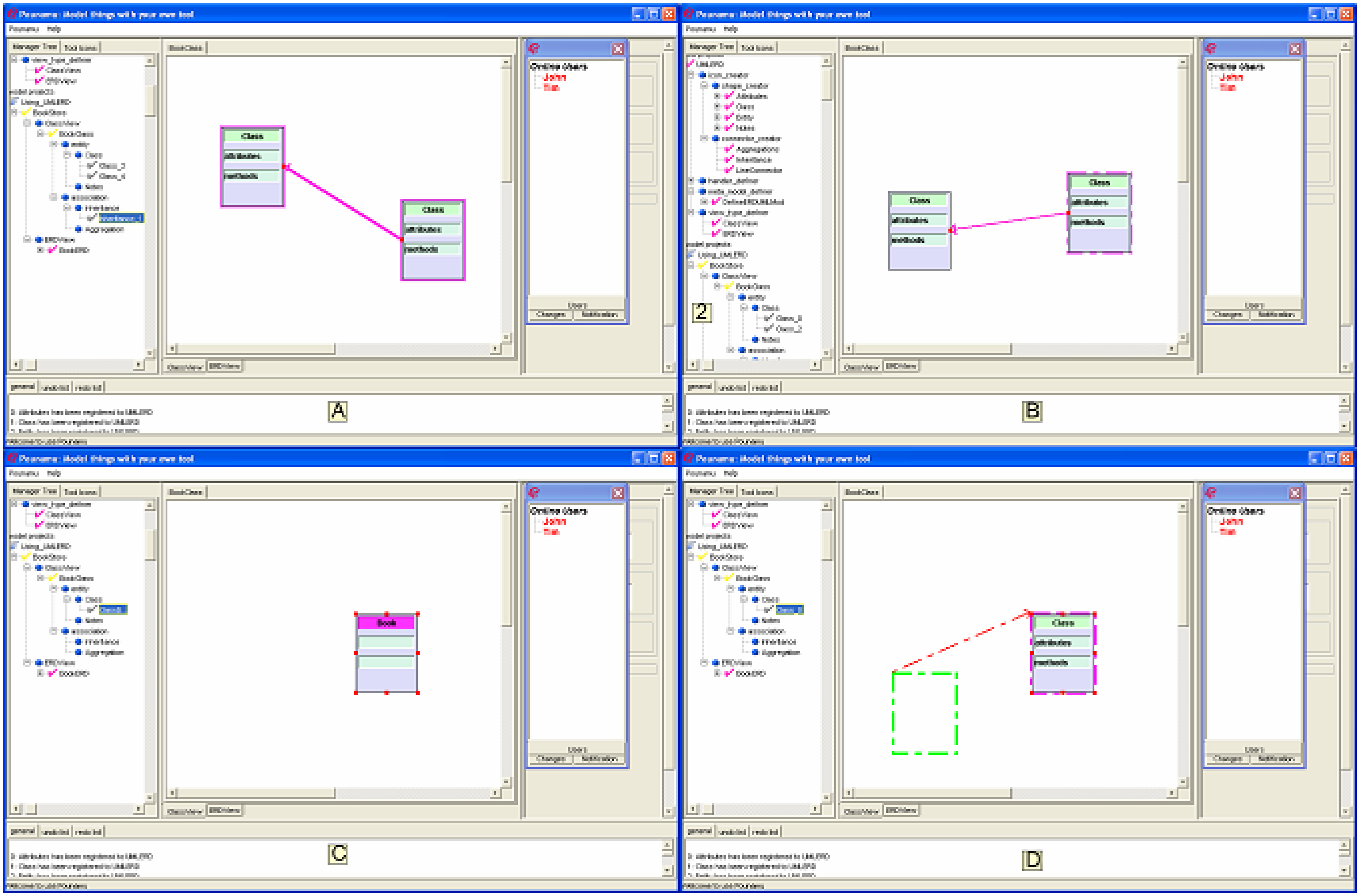
Collaboration support

- Problems
  - Want to use Marama tools in collaborative situations & hence need support for both synchronous and asynchronous collaboration
- Solutions
  - Pounamu - web service based collaboration plug in provides synch and asynch multi user support (Akhil Mehra 780 project)
  - Pounamu - web service based group awareness and CVS plugins extend to provide visual indication of other users' actions when collaboratively editing and shared document versioning (Akhil Mehra MSc thesis)
  - Marama – use of CVS via Eclipse workspace
  - Marama – differ & merger for DSVLs

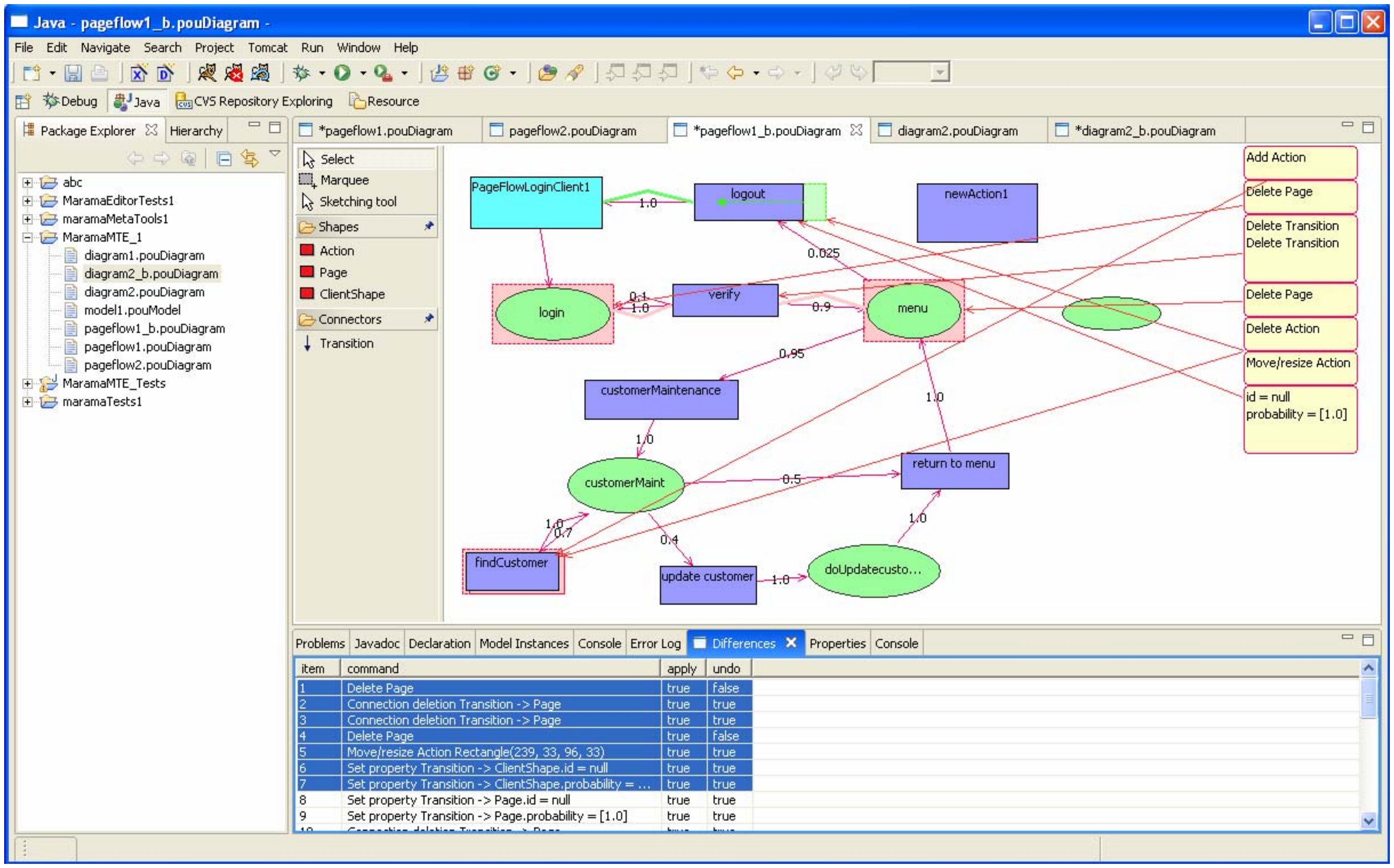
General collaboration architecture



Group Awareness Example - Pounamu



Visual Differ Example - Marama

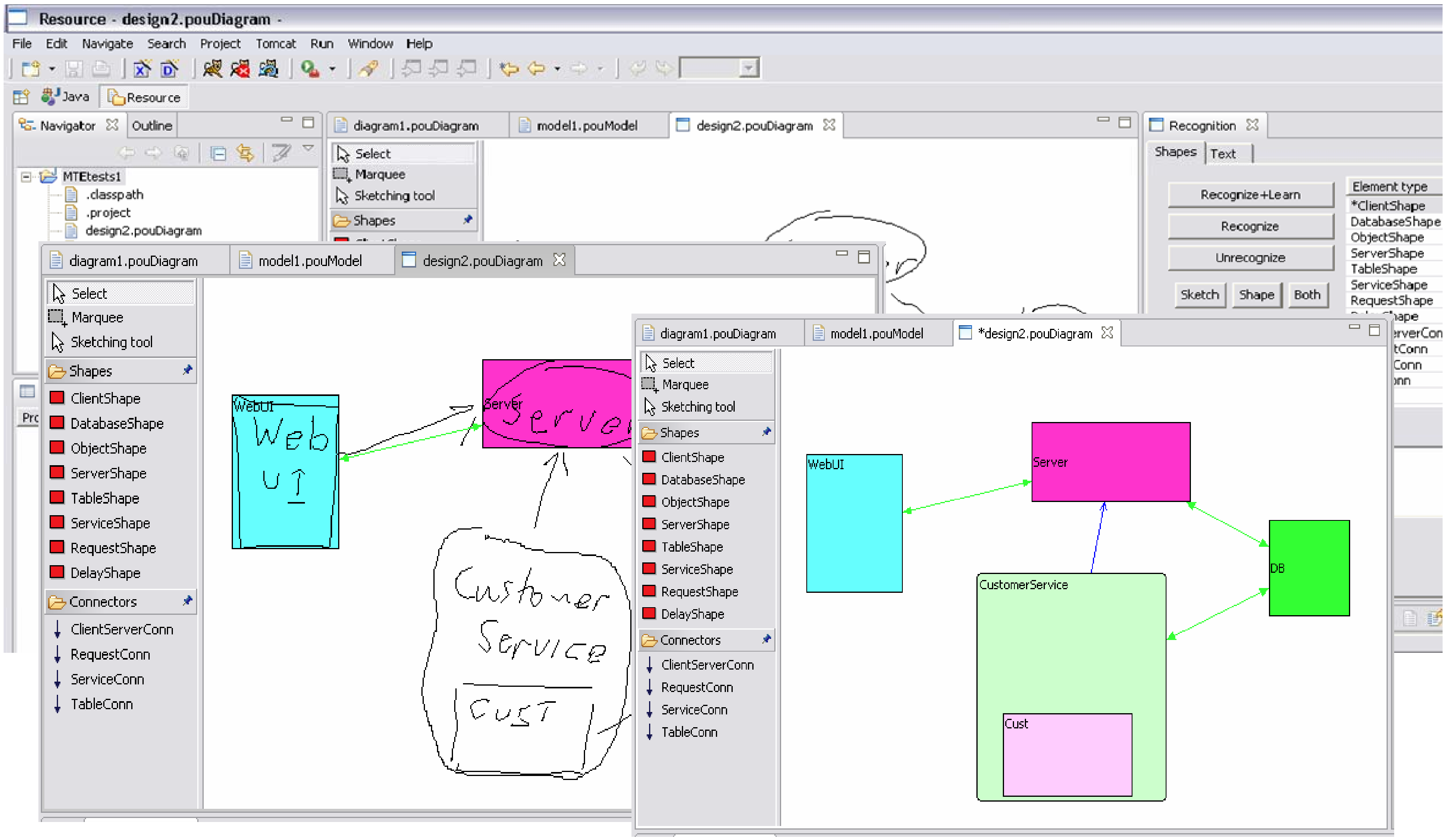


item	command	apply	undo
1	Delete Page	true	false
2	Connection deletion Transition -> Page	true	true
3	Connection deletion Transition -> Page	true	true
4	Delete Page	true	false
5	Move/resize Action Rectangle(239, 33, 96, 33)	true	true
6	Set property Transition -> ClientShape.id = null	true	true
7	Set property Transition -> ClientShape.probability = ...	true	true
8	Set property Transition -> Page.id = null	true	true
9	Set property Transition -> Page.probability = [1.0]	true	true
10	Connection deletion Transition -> Page	true	true

Sketching-based UI

- Problems
  - Classical tool bar-mouse interaction
  - Want to support more flexible input of DSVL elements
  - Want to support pen-based interaction e.g. TabletPC, stylus on Palm/PDAs, large E-whiteboards, touch screens…
- Solutions
  - MaramaSketch plug-in
  - Augments Marama editor to support pen-based editing
  - Training set of shapes/text specified by users
  - Works for any Marama-implemented DSVL tool

MaramaSketch interface



Summary

- Marama is an evolving tool that has itself been developed out of earlier tool projects (MViews, JViews, Pounamu)
- Very much a research prototype to provide proof of concept implementation of research ideas
  - However, now developed to a level of semi-robustness
  - Fifth year of use in 732!
- Plenty of scope to undertake projects/theses developing or applying Marama