

# Marama Extensions

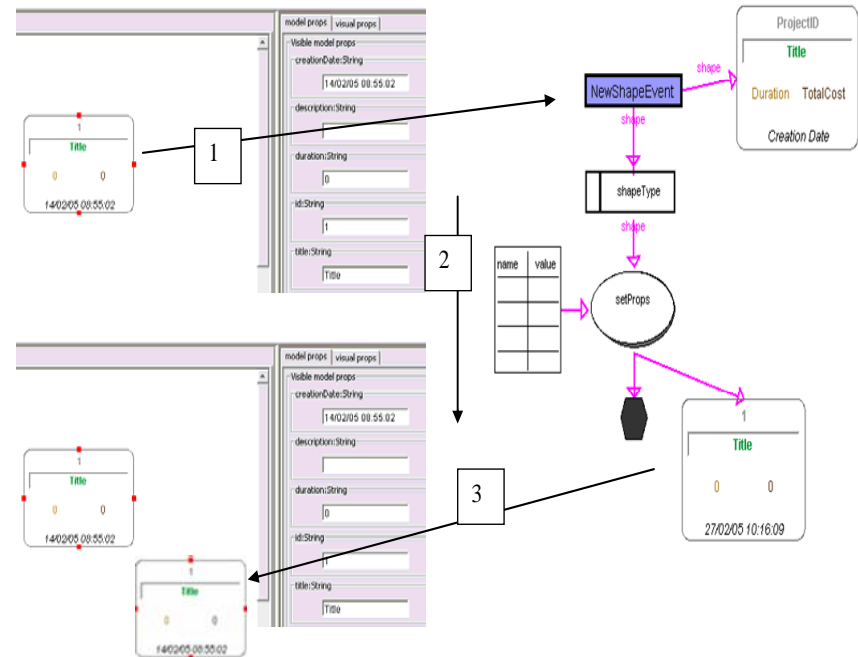
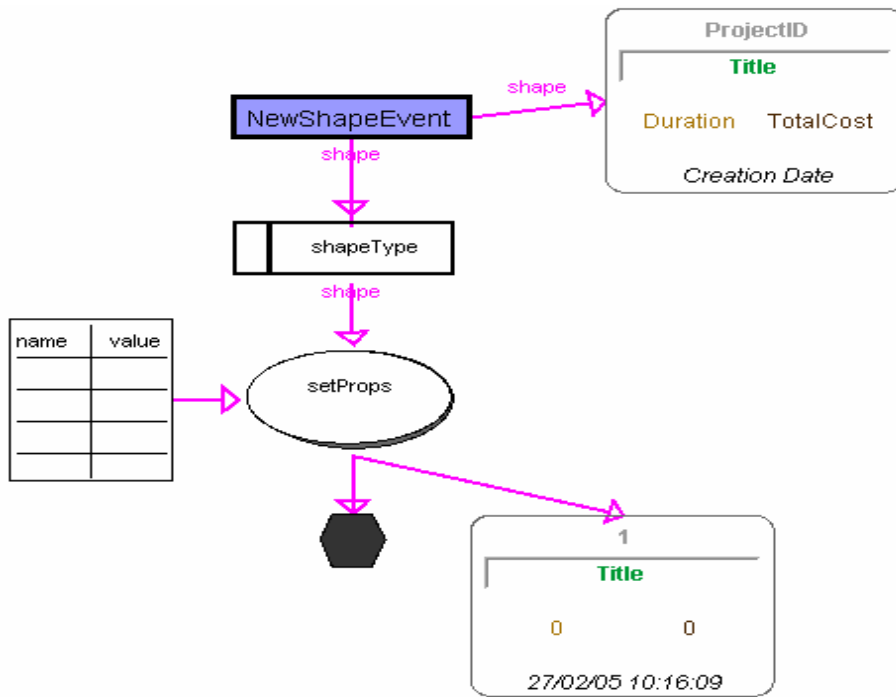
- **Aim of section:**
  - Look at work undertaken to extend Marama core features (recent and current)
  - Problems being addressed and solutions adopted
- **Contents**
  - Behaviour specification
    - Formulae
    - DSL for event handling
  - Back end code/model import/export
  - Collaboration/awareness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

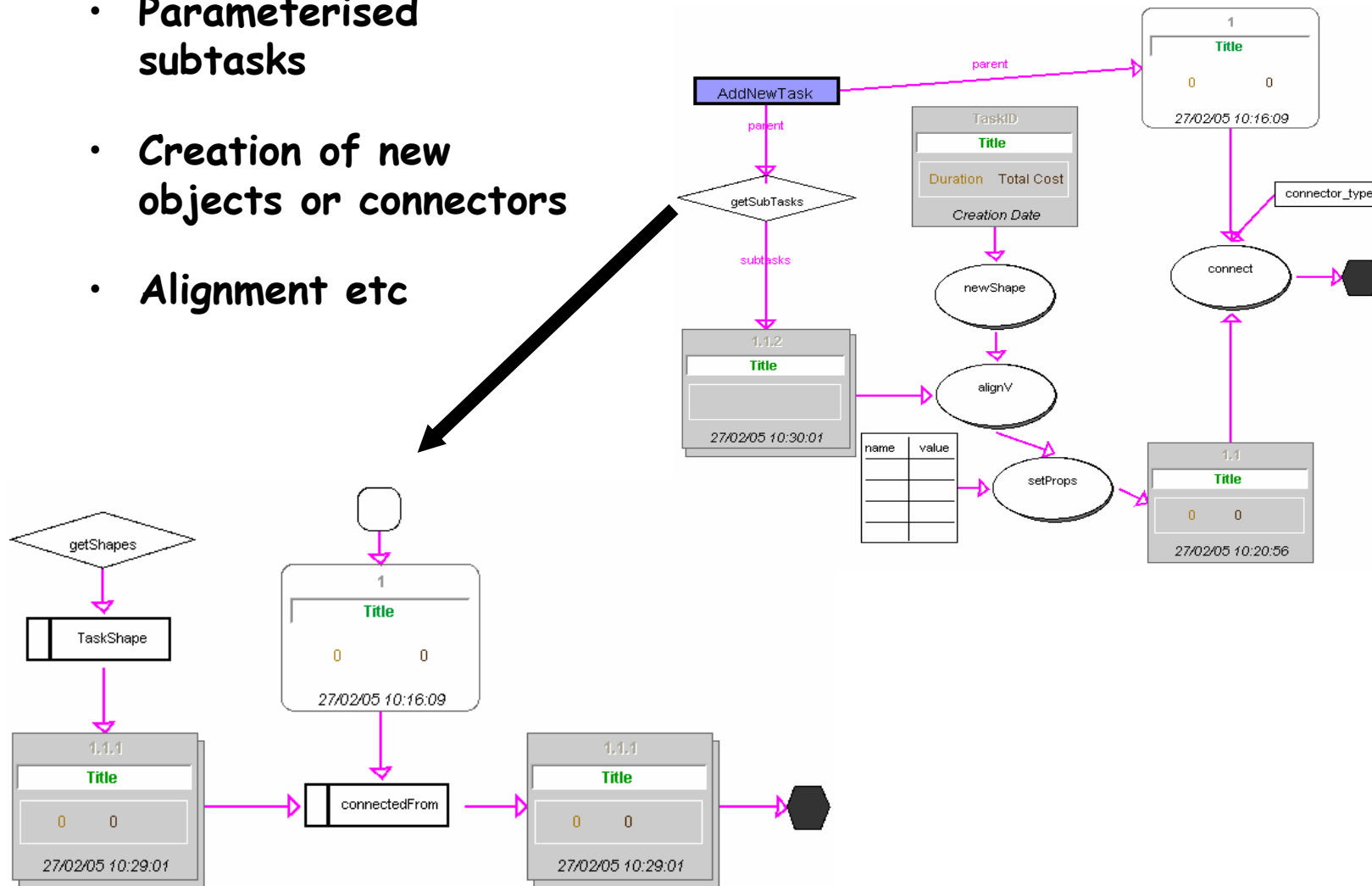
# Kaitiaki

- 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



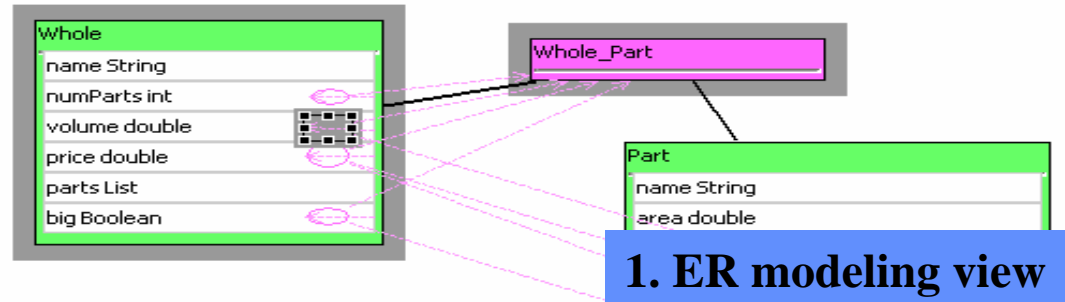
# Kaitiaki Debug view

The screenshot displays the Pounamu application window, titled "Pounamu: Model things with your own tool". The interface is divided into several panes:

- Tool Icons Manager Tree:** A list of tool icons on the left side, including "ProjValuesUponCre", "AND\_stage", "OR\_stage", "process\_stage", "process\_stage\_mult", "start\_stage", "top\_stage", "ModellerPanel", "PounamuEntity", "PounamuShape", "OR\_stage", "data", "Event\_Type\_Asynch", "Event\_Type\_Synch", "Event\_Type\_Sync", "or-all\_statement", "while\_statement", "variable\_argument", "shared\_data\_struct", "dataStore", "statement", "or-each\_statement", "action", "action\$setProp", "query", "filter", "filter\$1", "data\_collection", "project", "Project\$1", "Project\_29", "task", and "task\$1".
- view\_type\_WBS\_0:** A Work Breakdown Structure (WBS) diagram showing a hierarchical tree of tasks. The root node is labeled "1" and contains a "Title" field and two "0" values. It has two children, "1.1" and "1.2", each with a "Title" field and two "0" values. Node "1.1" has a child "1.1.1" with a "Title" field and two "0" values. Timestamps are shown below each node: "27/02/05 10:28:09" for the root, "27/02/05 10:28:14" for "1.1", "27/02/05 10:29:01" for "1.2", and "27/02/05 10:28:29" for "1.1.1".
- debugging view:** A diagram showing the execution flow of the "AddNewTask" process. It includes a decision diamond labeled "getSubTasks" and a task box labeled "1.1.2" with a "Title" field. A "parent" relationship is shown between "AddNewTask" and "getSubTasks", and a "subtasks" relationship between "getSubTasks" and "1.1.2". A "TaskID" box contains fields for "Title", "Duration", "Total Cost", and "Creation Date". Below it, a "newShape" oval is connected to an "alignV" oval.
- State Info:** A panel showing the current state of the application, including "AddNewTask Event Fired" and "PounamuShape Project\$1".
- Log:** A panel at the bottom showing error messages: "89: java.lang.ClassNotFoundException: pounamu.tools.Event\_Spec\_Tool.handlers.visualhandlers.usertriggeringhandlers.generateEHCode156" and "90: Exception from Class RestoreModelViewFromXML: java.lang.NullPointerException".

# Formula definer for meta-model

- Design-time
  - ER modeling view
  - Formula construction view
  - Formulae view



1. ER modeling view

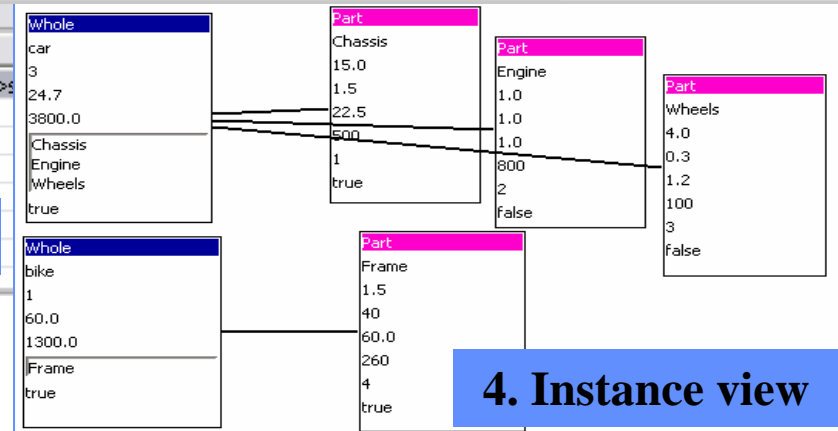
The interface shows a 'Formulae' tab with a dropdown menu set to '4'. The formula editor contains: `Whole_Part->collect(Part.volume)->sum()`. A list of operations is visible on the right: `->sum()`, `->avg()`, `->collect()`, `->size()`, `->any()`, `->forall()`, `->exists()`, `->select()`, `->reject()`.

2. Formula construction view

id	context	formula
1	Whole.price	Whole_Part->collect(Part.cost*(1.0+Part.markup))->sum()
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_Part->any()

3. Formulae view

- Run-time
  - Instance view
  - Master-details tabular view with an on-demand tooltip showing the underlying formula



4. Instance view

name	numParts	volume	price	parts	big	...
bike	1	60.0	1300.0			
car	3	24.7	3800.0			
Chassis	15.0	1.5	22.5	500	1	true
Engine	1.0	1.0	1.0	800	2	false
Frame	1.5	40	60.0	260	4	true
Wheels	4.0	0.3	1.2	100	3	false

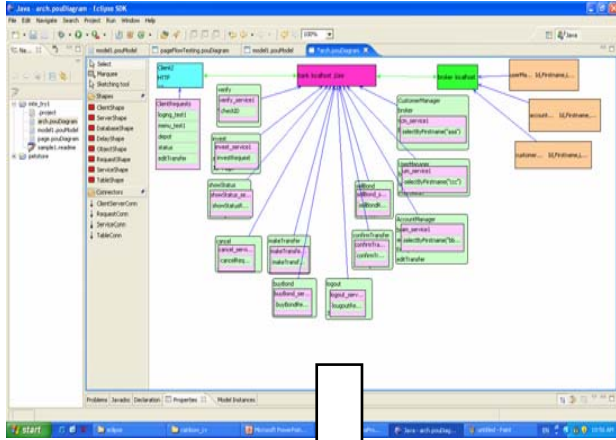
with an on-demand tooltip showing the underlying formula: `Whole_Part->collect(Part.cost*(1.0+Part.markup))->sum()`

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

The screenshot displays the MaramaVMLPlus software interface. At the top, there are three tabs: "BPMN.vmlPlusDiagram", "BPEL.vmlPlusDiagram", and "BPMNtoBPEL.vmlPlusDiagram". The main workspace features a 3D cube labeled "VML MAPPING CONTROL". The top face of the cube has two input fields: "source: myBPMN.xsd" and "target: myBPEL.xsd". Below the cube are two buttons: "Load source classes/types" and "Load target classes/types".

On the left side, there is a toolbar with the following options: Select, Marquee, Sketching tool, Shapes, Native Type, BlackBox Template, Connectors, Condition, Mapping, and Auto Mapping.

In the center, there are two class diagrams. The top diagram is "BPMNProcess" with attributes: name : string, eventStart : BPMNProcess..., eventEnd : BPMNProcess..., gateway : BPMNProcess..., and process : BPMNProcess\_p... (with a plus sign). The bottom diagram is "BPMNProcess\_eventStart" with attributes: id : decimal and next : decimal. Dashed lines indicate relationships between these attributes and the mapping specification dialog.

The "List of mapping sources and target" dialog box is open, showing a dropdown menu with "BPMNProcess\_process/id" and a double-headed arrow. The "Mapping specification" section contains the following XML snippet: 

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

 Below this, there are several dropdown menus and input fields for defining the mapping, including "vml:boolean", "vml:attributeValue BPMNProcess\_process/id", "=", "vml:attributeValue decimal", "/--", "&&", "vml:boolean", "vml:attributeValue BPMNProcess\_process/type", "=", and "vml:constant receive". The "Properties" section at the bottom shows "Value: receive" and "Type: string", with "Accept" and "Cancel" buttons.

# Thin-client/Remote interfaces

- **Problems**

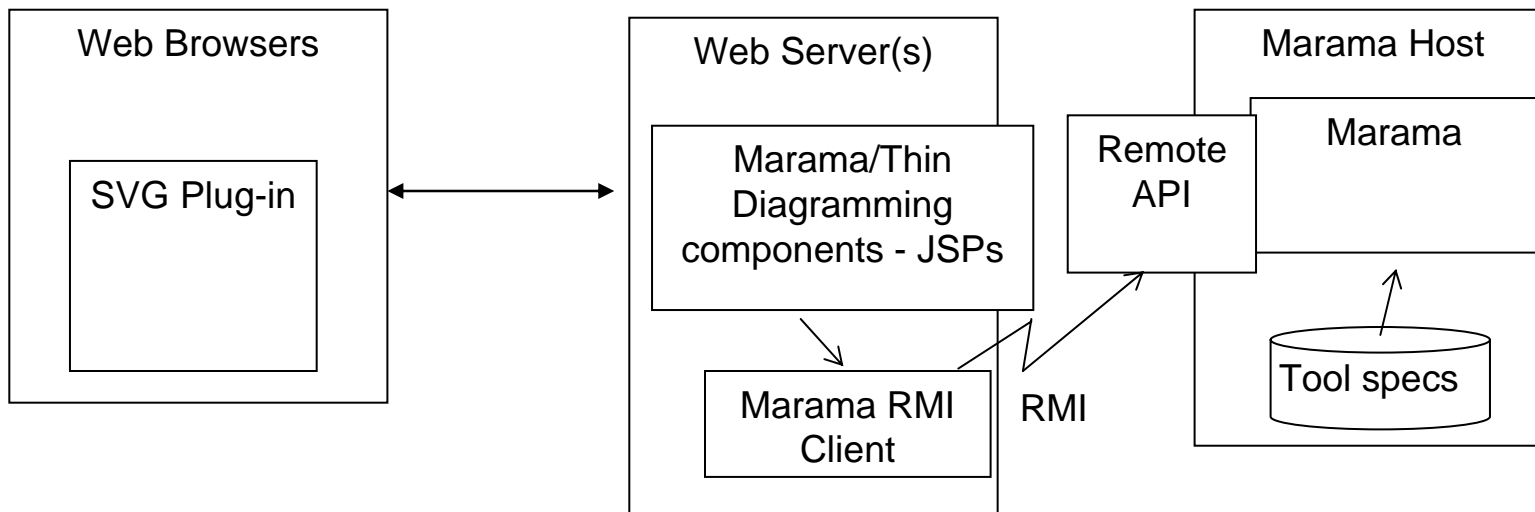
- Need to access Marama tools remotely on a variety of different devices
- Need to drive Marama remotely

- **Solutions**

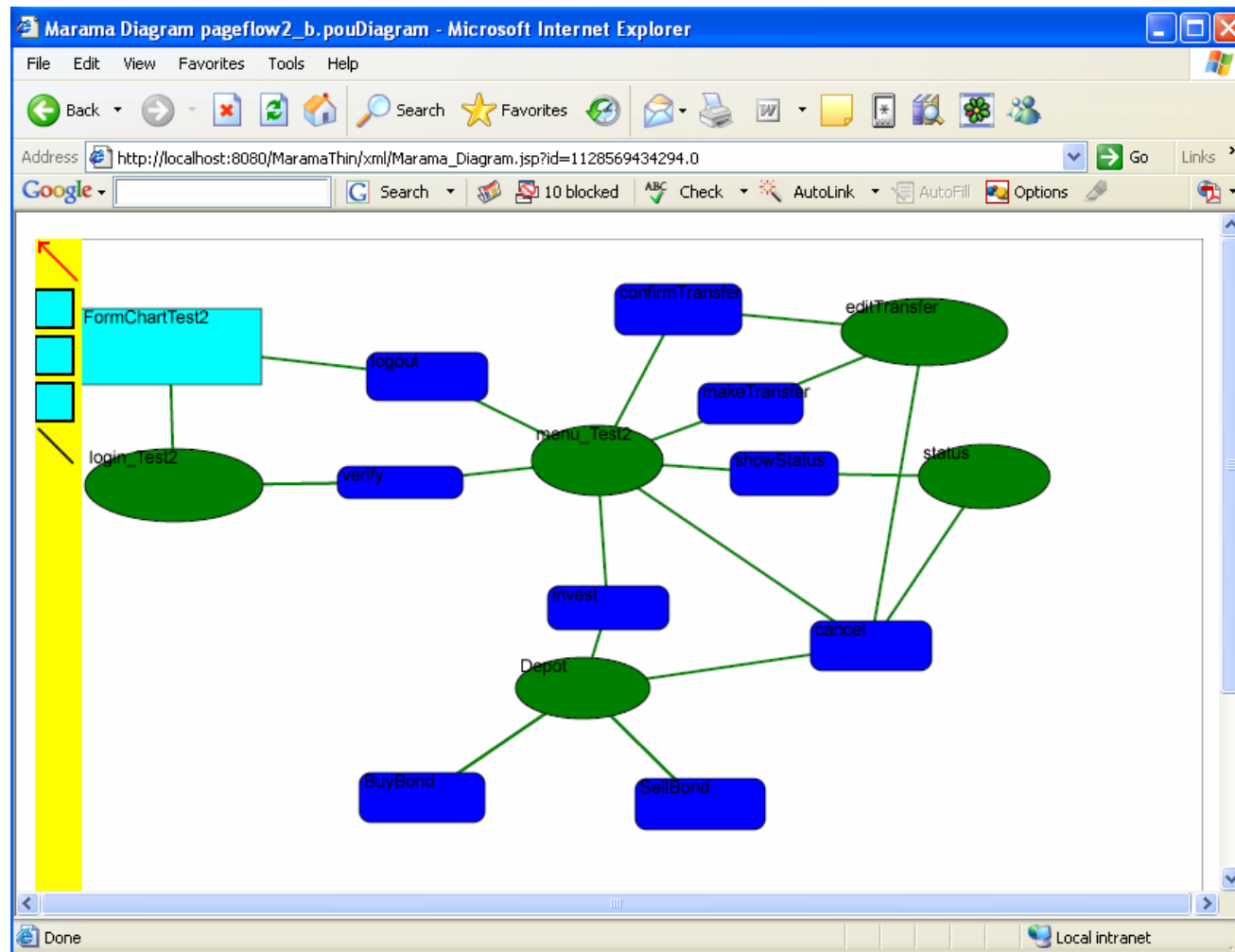
- RMI interface to Marama API
- Thin client interface for web browser interaction with any Marama generated tool (Penny Cao MSc thesis done)
- Mobile phone interface for Marama generated tools (Joe Zhao MSc thesis done)
- Generalise framework and add VRML interface (Joe Zhao done)
- Add games engine 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

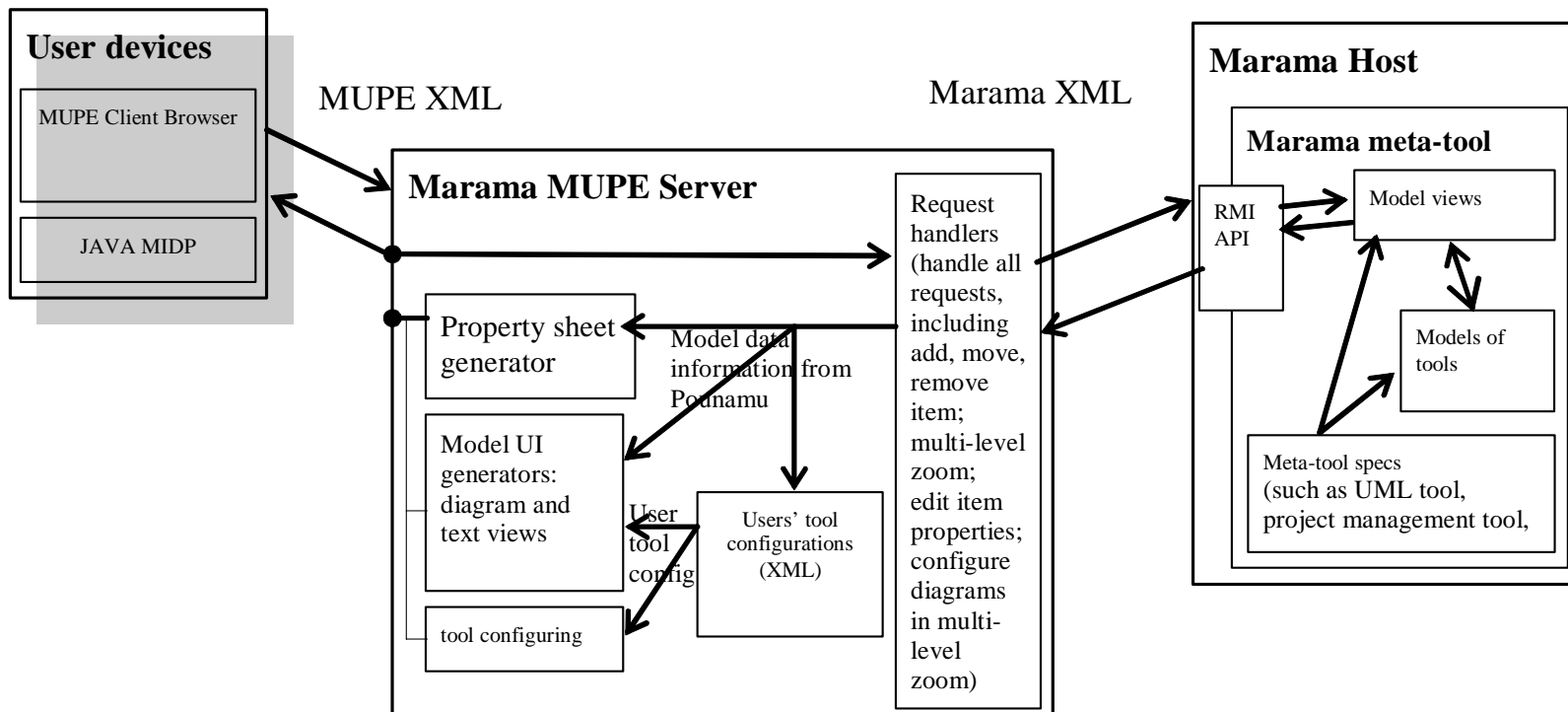


# Thin client interface example

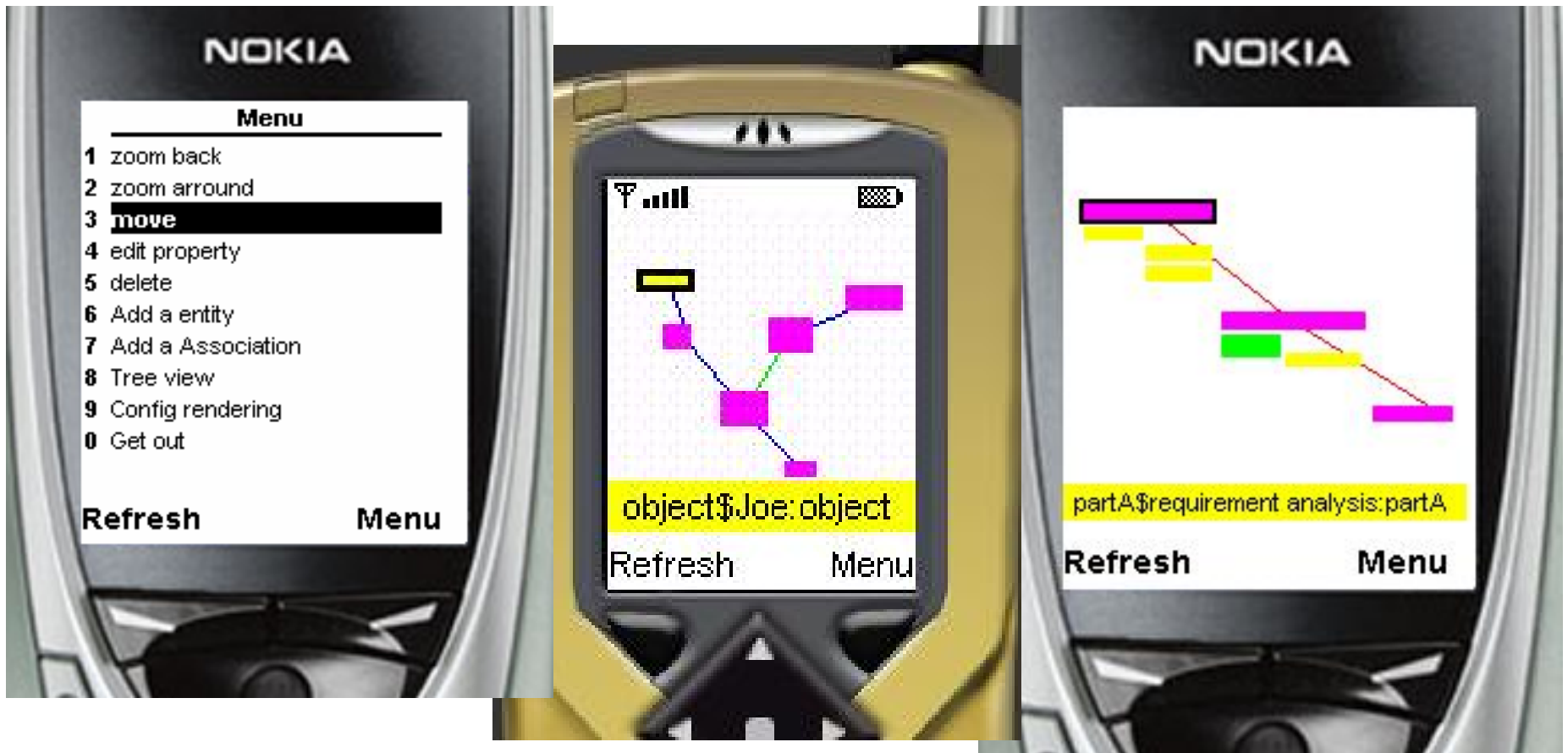


# MUPE interface

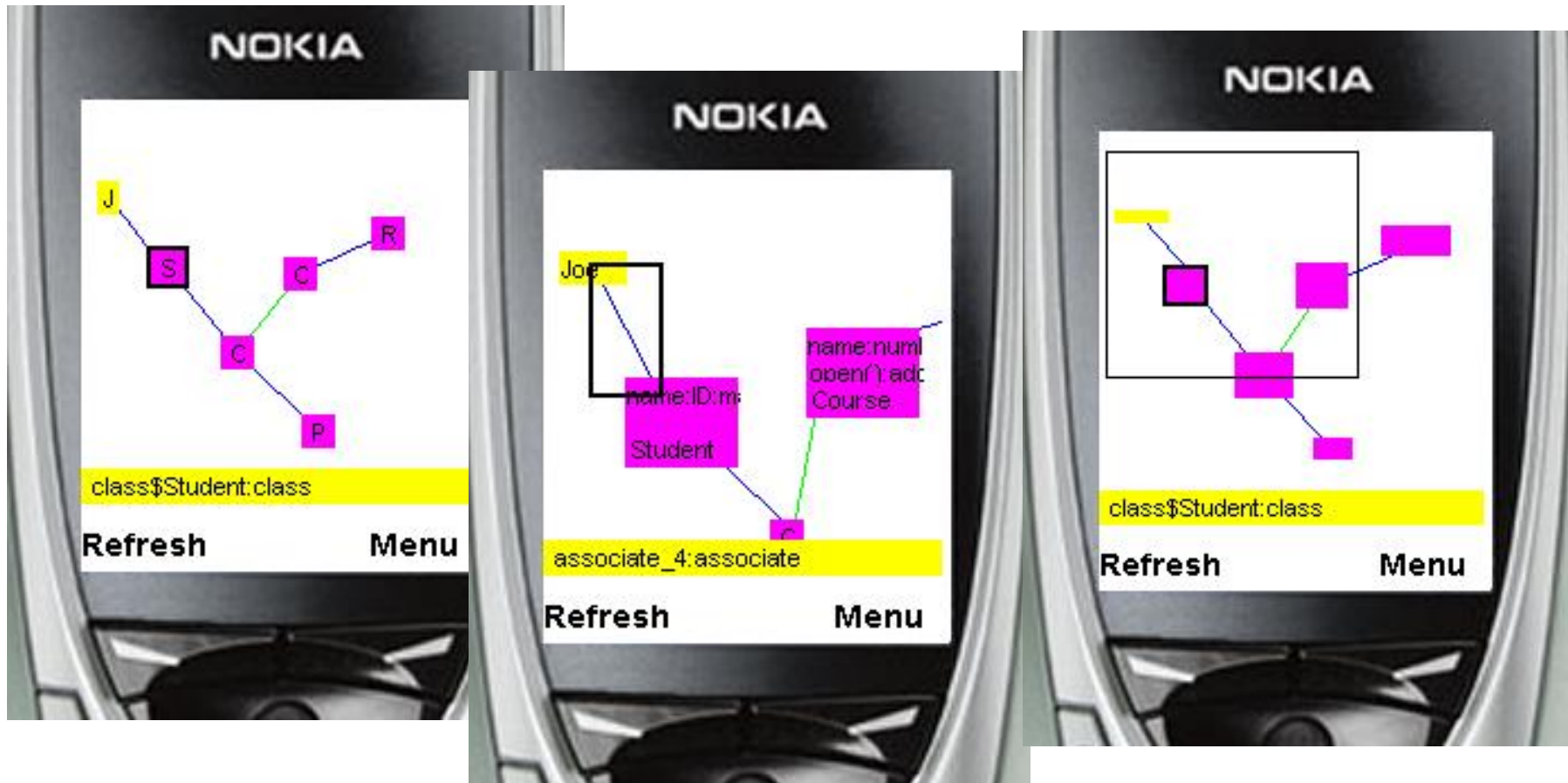
- Support for viewing and editing Pounamu & Marama tool views on cellphones
- Uses Nokia's MUPE open source mobile collaboration server plus MUPE client on phone
- Has several features for semantic zooming to allow diagrams to be sensibly visualised/edited on small screen



# Example MUPE interface usage



# Element zooming and overview



# Collaboration support

- **Problems**

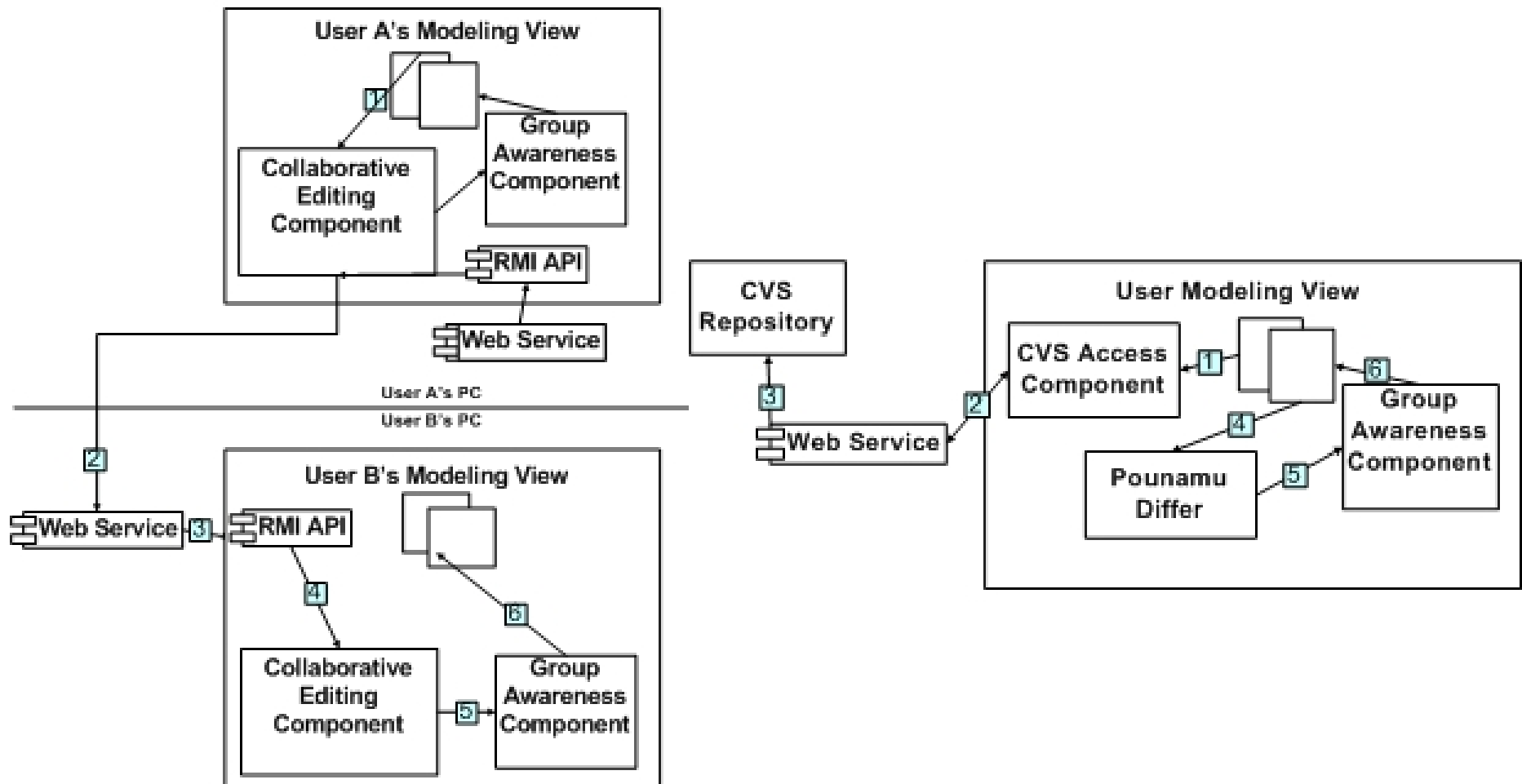
- Want to use Marama tools in collaborative situations & hence need support for both synchronous and asynchronous collaboration

- **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

The image displays four sequential screenshots (A, B, C, D) of a UML modeling tool interface, illustrating group awareness in a multi-user environment. Each screenshot shows a 'Manager Tree' on the left, a central workspace, and an 'Online Users' window on the right.

- Screenshot A:** Shows a 'Class' box with 'attributes' and 'methods' sections. A pink arrow points from the 'attributes' section to another 'Class' box. The 'Online Users' window shows 'John' and 'Tim'.
- Screenshot B:** Shows the same 'Class' box. A pink arrow points from the 'attributes' section to the 'methods' section. A yellow box with the number '2' is visible in the 'Manager Tree'.
- Screenshot C:** Shows a 'Book' box with 'attributes' and 'methods' sections. The 'Online Users' window shows 'John' and 'Tim'.
- Screenshot D:** Shows the 'Book' box. A red dashed arrow points from the 'attributes' section to the 'methods' section. A green dashed box highlights the 'Book' box.

Each screenshot includes a status bar at the bottom with the following text: 'attributes has been registered to UML/RO', 'Class has been registered to UML/RO', and 'Book has been registered to UML/RO'. The status bar also indicates the user's role: 'general | unlocked | read-only'.

# Visual Differ Example - Marama

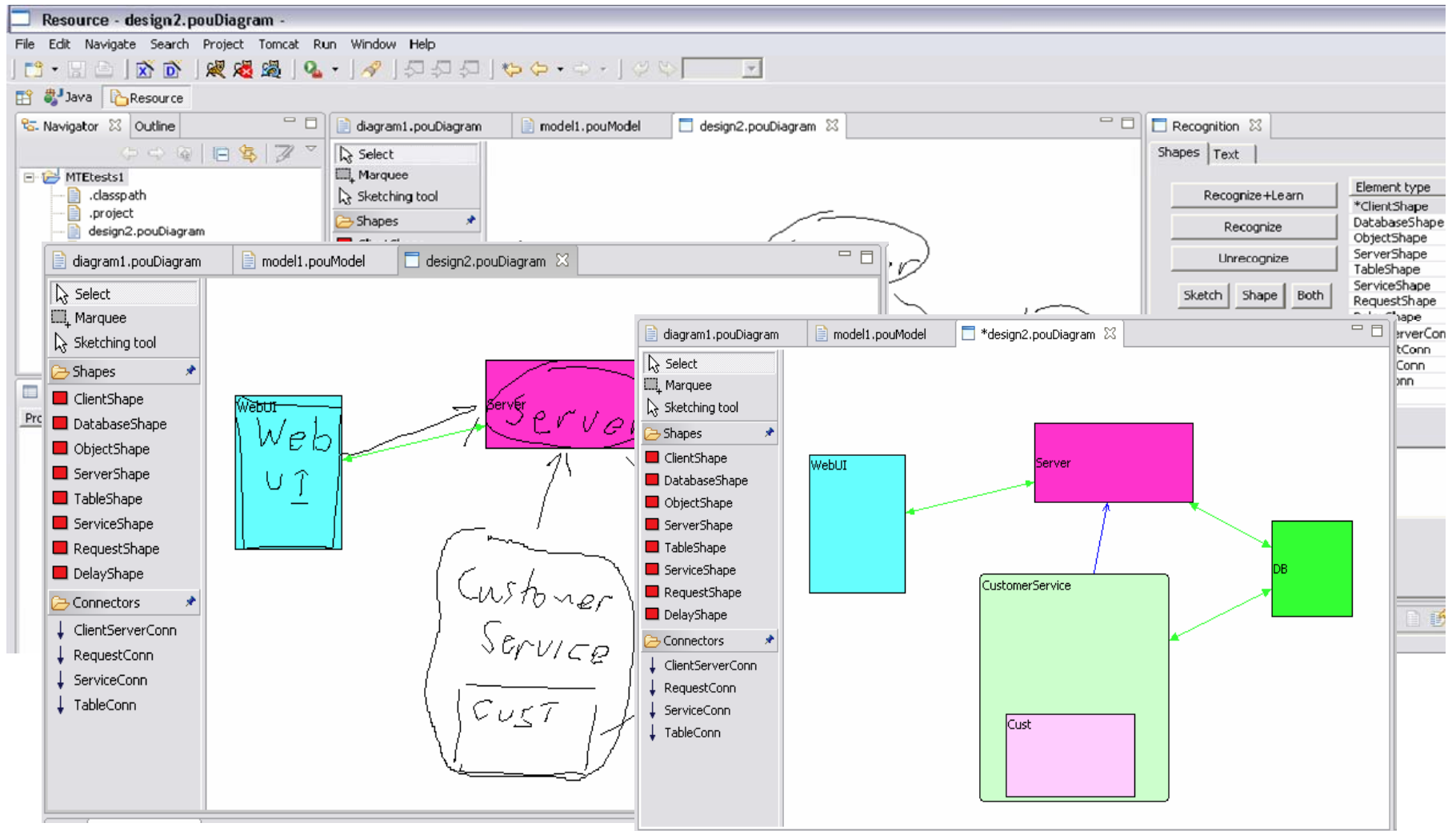
The screenshot shows the Marama IDE interface. The main workspace displays a state transition diagram with various nodes and transitions. A 'Differences' panel on the right lists actions such as 'Delete Page', 'Delete Transition', 'Move/resize Action', and 'id = null probability = [1.0]'. A 'Differences' table at the bottom shows a list of items with their commands and apply/undo status.

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 DSL 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 DSL 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**