# Assignment #1 Success Factors

- Project management: team leader, work plan, meetings, hours/work tracking ...
- Innovation comes from research! Read literature with a critical mind!
  - Implementation is not hard, but motivation and design are
- Scientific writing – learn from paper examples
- Discuss your work with Karen (email/phone appointment preferred)

# 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 (integrated DSVLs for event handling)
    - Kaitiaki event flow
    - MaramaTatau formulae
    - ViTABaL-WS high-level event architecture
    - Generalisation to an event abstraction framework
  - Critic authoring
  - Back end code/model import/export
  - Thin-client diagramming
  - Collaboration/awareness
  - Sketching-based input
  - Other stuff

# Modification, integration, extension

- Marama is live,
  - Changes to a tool specification are immediately reflected in executing models using that tool (well – usually have to close/reopen the editing views in the in-use tool… ☺)
- Formulae and Handlers provide behavioural extension capability
  - Formulae compiled to OCL & interpreted
  - Handlers via API, code modified in the invoking Eclipse
- EMF data structures and Marama APIs provide internal integration with other Marama tools and other Eclipse plug-ins
  - Can have multiple Marama tools communicate
  - Can control/exchange data with other Eclipse plug-ins
- Can add XSLT-based backends manipulating the XML save format
  - Eg for code generation and reverse engineering
  - MaramaTorua data transformation tool being integrated into Marama meta-tools to support this "nicely"…
- RMI interfaces provide external integration capability
  - Have used for developing generic thin client and mobile phone modeller interfaces, process modelling and enactment tool, collaboration and group awareness tools, integration with project management tool

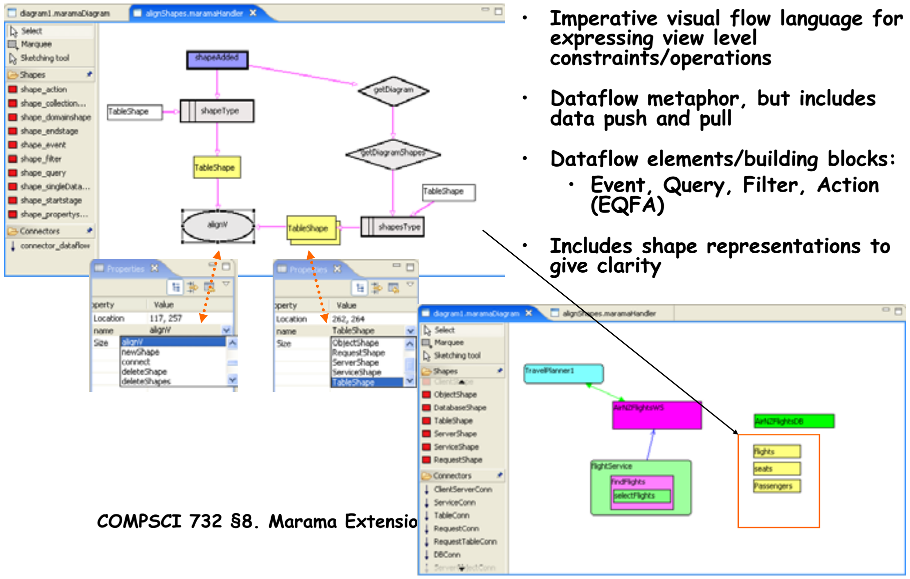
# Exercise/Discussion

- What modelling behaviours do you want a DSVL tool to have?
- Are there any common abstractions for DSVL tool behaviour specifications?
- In pairs come up with a list (2-3 mins)
- In pairs of pairs exchange and discuss your lists (2 mins)

# 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
    - Aimed at handlers for view manipulation
  - MaramaTatau meta-model constraint language
    - OCL expns + visual assistance for specifying computations at meta-model level (like spreadsheets at a type level)
  - ViTABaL-WS high level event flow
    - Use Tool Abstraction based ideas
- Status
  - All these projects completed by Karen Li (PhD)
  - Formulae added to Marama meta-tools (disabled in current version),
  - Kaitiaki, ViTABal-WS to come (proofs of concept done)…

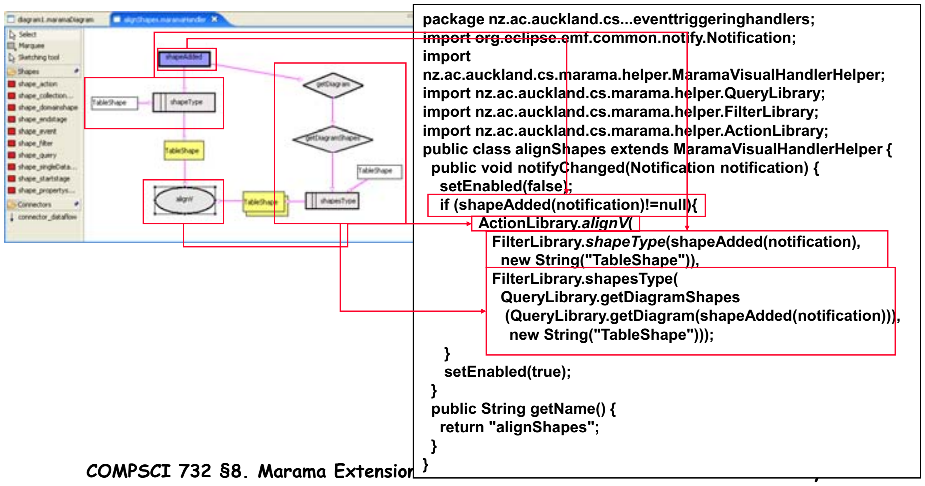
# Kaitiaki



- Imperative visual flow language for expressing view level constraints/operations
- Dataflow metaphor, but includes data push and pull
- Dataflow elements/building blocks:
  - Event, Query, Filter, Action (EQFA)
- Includes shape representations to give clarity

# Kaitiaki

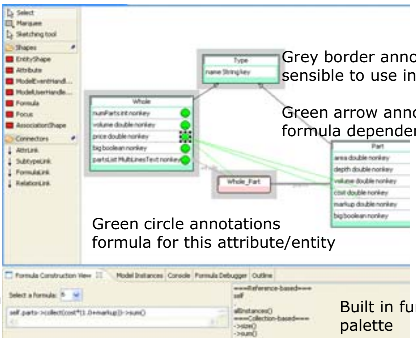
- Code generation



```
package nz.ac.auckland.cs...eventtriggeringhandlers;
import org.eclipse.emf.common.notify.Notification;
import
nz.ac.auckland.cs.marama.helper.MaramaVisualHandlerHelper;
import nz.ac.auckland.cs.marama.helper.QueryLibrary;
import nz.ac.auckland.cs.marama.helper.FilterLibrary;
import nz.ac.auckland.cs.marama.helper.ActionLibrary;
public class alignShapes extends MaramaVisualHandlerHelper {
  public void notifyChanged(Notification notification) {
    setEnabled(false);
    if (shapeAdded(notification)!=null){
        ActionLibrary.alignV(
          FilterLibrary.shapeType(shapeAdded(notification),
            new String("TableShape")),
          FilterLibrary.shapesType(
            QueryLibrary.getDiagramShapes
             (QueryLibrary.getDiagram(shapeAdded(notification))),
              new String("TableShape")));
     }
     setEnabled(true);
  }
  public String getName() {
    return "alignShapes";
  }
}
```

# MaramaTatau – model level constraints

- MaramaTatau allows constraints to be specified as OCL expressions over the meta-model elements:
  - Textual OCL expression
  - But constructed using spreadsheet approaches
  - Click and connect
  - High level visual repn



Grey border annotations sensible to use in formula

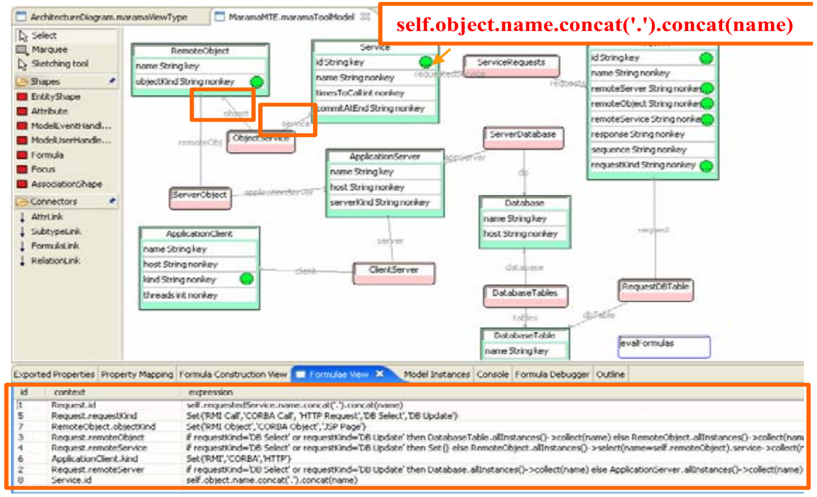
Green arrow annotations formula dependencies

Green circle annotations formula for this attribute/entity

Formula construction area

Built in function palette

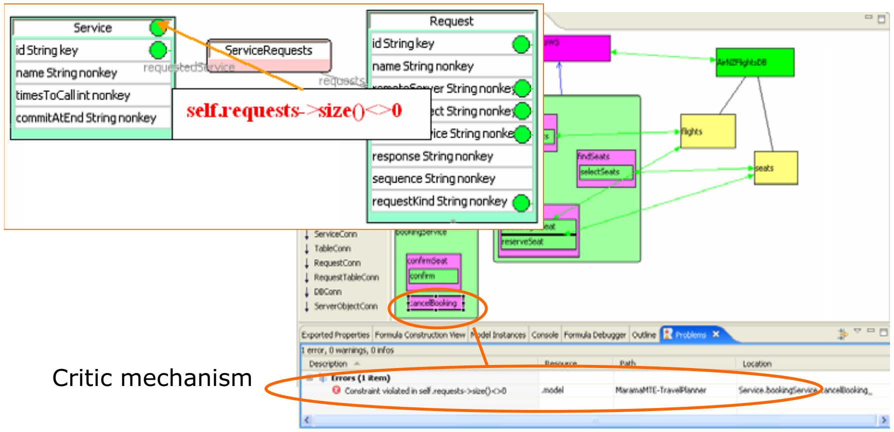
# MaramaMTE example



self.object.name.concat('.').concat(name)

| id | context | expression |
|---|---|---|
| 1 | Request.id | self.requestedService.name.concat('.').concat(name) |
| 5 | Request.requestKind | Set{'RMI Call','CORBA Call', 'HTTP Request','DB Select','DB Update'} |
| 7 | RemoteObject.objectKind | Set{'RMI Object','CORBA Object','JSP Page'} |
| 3 | Request.remoteObject | if requestKind='DB Select' or requestKind='DB Update' then DatabaseTable.allInstances()->collect(name) else RemoteObject.allInstances()->collect(nam |
| 4 | Request.remoteService | if requestKind='DB Select' or requestKind='DB Update' then Set {} else RemoteObject.allInstances()->select(name=self.remoteObject).service->collect(r |
| 6 | ApplicationClient.kind | Set{'RMI','CORBA','HTTP'} |
| 2 | Request.remoteServer | if requestKind='DB Select' or requestKind='DB Update' then Database.allInstances()->collect(name) else ApplicationServer.allInstances()->collect(name) |
| 0 | Service.id | self.object.name.concat('.').concat(name) |

# Constraint violation



self.requests->size()<>0

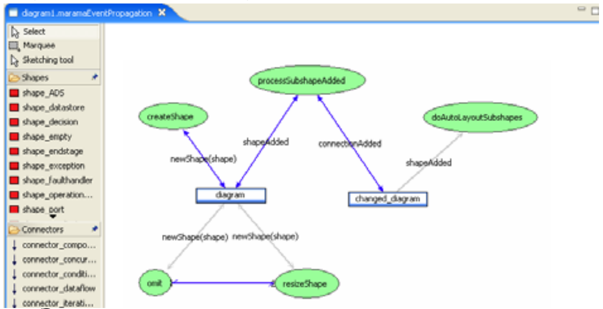
Critic mechanism

1 error, 0 warnings, 0 infos

| Description | Resource | Path | Location |
|---|---|---|---|
| Errors (1 item) | | | |
| Constraint violated in self.requests->size()<>0 | .model | MaramaMTE-TravelPlanner | Service.bookingService.cancelBooking_ |

# ViTABaL-WS

- High level Tool Abstraction based view
- Links together toolies (Marama library functions) and abstract data structures (Marama shared data structures)
- Describes event-based inter-connections between abstract components (encapsulating event response details)

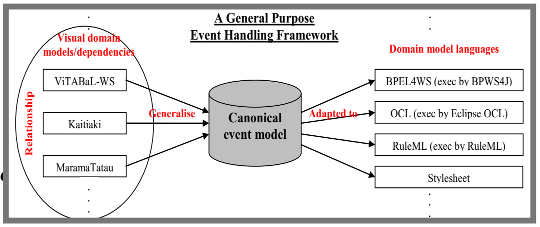


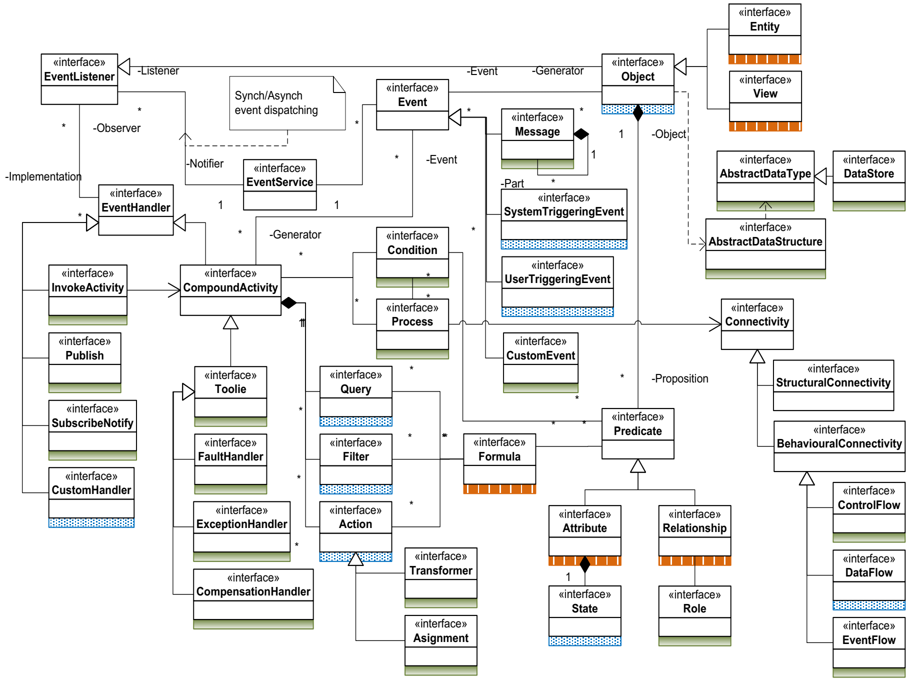
# Integration of 3 DSVLs for event handling

- Recap
  - Kaitiaki (dataflow metaphor): diagramming-based design tool interactions at low-medium abstraction level
  - MaramaTatau (spreadsheet metaphor): declarative meta-model structural dependencies and constraints at mixed low/high abstraction level
  - ViTABaL-WS (tool abstraction metaphor): event architecture description at high abstraction level
- Generalised the 3 DSVLs to an integrated visual approach for event handling specification
  - Derived a canonical event behaviour model
  - Enabled interoperability between the 3 event models
  - Supported synthesised runtime visualisation

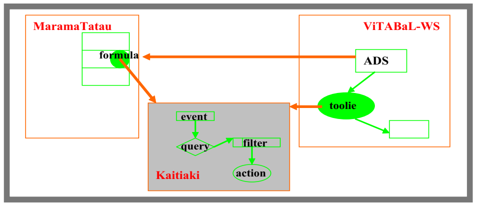
# Approach to generalisation

- Evolving Frameworks Pattern Language (Roberts and Johnson, 96)
  - The Three Examples pattern for establishing a framework
- Identified common abstraction
  - Combined atomic primitives extended by the 3 exemplar DSVLs
  - Removed redundancies
  - Added bridging elements
- Reserve metaphoric views in the style of the 3 exemplars
- Allow mapping between related concepts in each metaphor for model transformation – MaramaTorua (Huh et al, 2007) mapping specifications

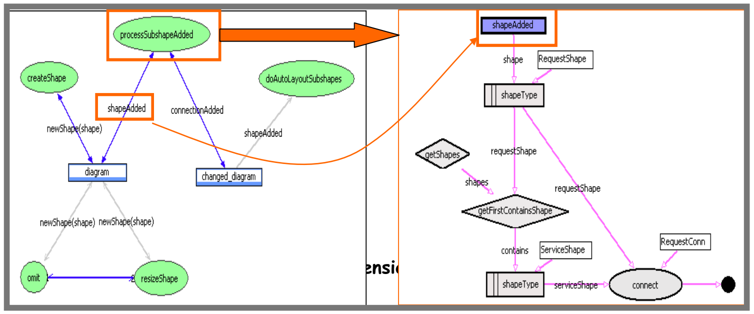




# Interoperability

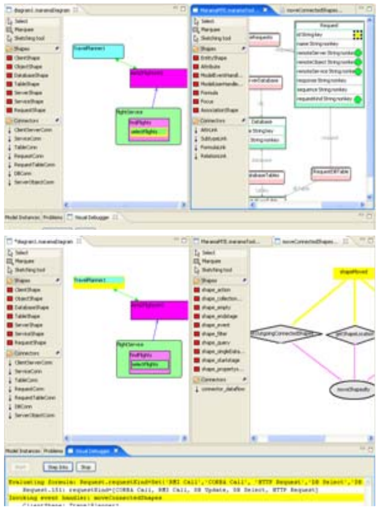


- TA metaphor is used to define high-level abstract data structures and functions and their coordination
- Abstract data structures are further constrained using formulas
- Abstract functions are further refined referencing EQFA specs
- Formulas can also reference EQFA specs



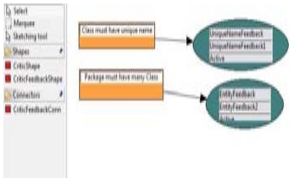
# Synthesised visual debugging

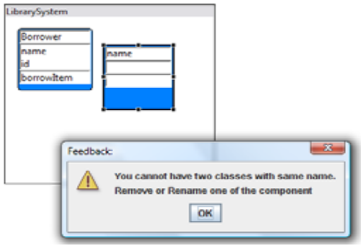
- Tool support for tracing and visualising event propagations and their effects
- Visualise both static dependency structure and dynamic event handling behaviour
- User controlled step-by-step visualisation
- Reuse design-level abstraction in runtime visualisation
- Aim to represent visual debugging at a high abstraction level, based on user-defined queries in a visual query language



# Critic Support

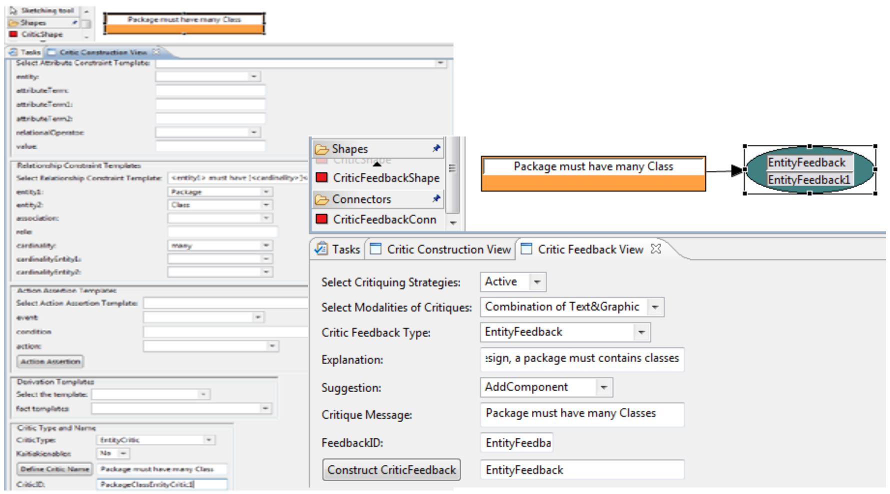
- Problems
  - Want to be able to rapidly specify critics a la ArgoUML to guide and assist tool users
- Solution
  - Critic authoring extension for Marama meta-tools
  - Allows critics to be specified as part of overall tool specification
  - (Norhayati Ali, PhD)





You cannot have two classes with same name.
Remove or Rename one of the component

# MaramaCritics



Select Attribute Constraint Template:
entity:
attributeTerm:
attributeTerm1:
attributeTerm2:
relationalOperator:
value:

Relationship Constraint Templates
Select Relationship Constraint Template: <entity1> must have [<cardinality>]<
entity1: Package
entity2: Class
association:
role:
cardinality: many
cardinalityEntity1:
cardinalityEntity2:

Action Assertion Templates
Select Action Assertion Template:
event:
condition:
action:

Derivation Templates
Select the template:
fact templates:

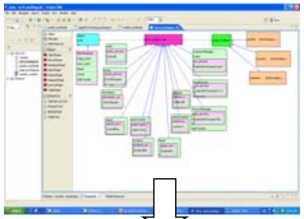
Critic Type and Name
CriticType: EntityCritic
KaitiakiEnabled: No
Package must have many Class
CriticID: PackageClassEntityCritic1

Select Critiquing Strategies: Active
Select Modalities of Critiques: Combination of Text&Graphic
Critic Feedback Type: EntityFeedback
Explanation: :sign, a package must contains classes
Suggestion: AddComponent
Critique Message: Package must have many Classes
FeedbackID: EntityFeedba
EntityFeedback

# 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 MaramaTorua XSLT generator for complex data transformation
- Status
  - MaramaTorua tool developed by Jun Huh
  - Being integrated into Marama meta-tools (near complete)

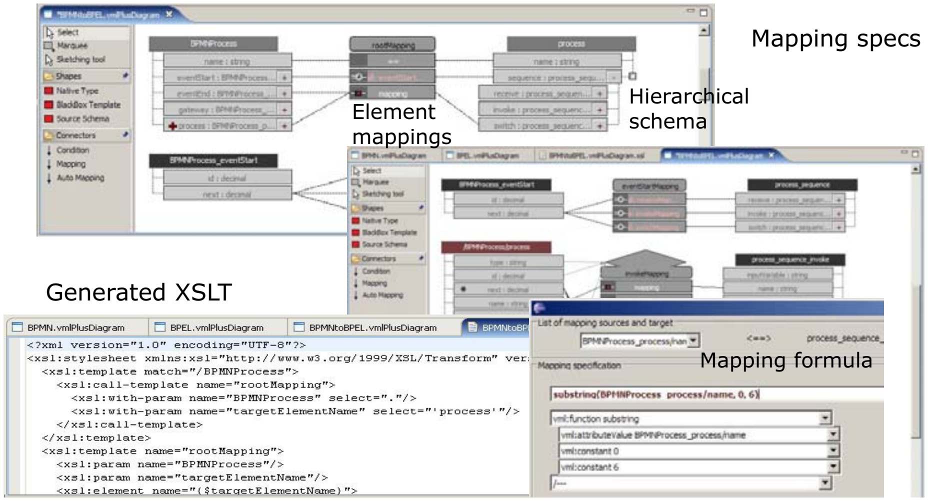
# Code Gen using Jet



```
<%@ 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%>;
<% } %>
<% } %>
<% } %>


...
```

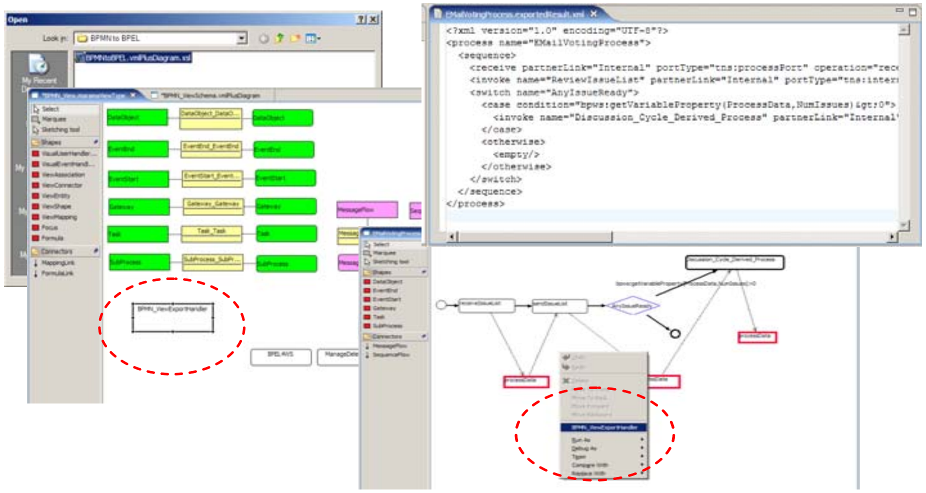
```
import java.rmi.Naming;
import java.util.List;
import java.util.ArrayList;

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));
```

# MaramaTorua –visual mapping/ model transformation specn and generation



```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" ver
  <xsl:template match="/BPMNProcess">
    <xsl:call-template name="rootMapping">
      <xsl:with-param name="BPMNProcess" select="."/>
      <xsl:with-param name="targetElementName" select="'process'"/>
    </xsl:call-template>
  </xsl:template>
  <xsl:template name="rootMapping">
    <xsl:param name="BPMNProcess"/>
    <xsl:param name="targetElementName"/>
    <xsl:element name="{$targetElementName}">
```

# Installing mapping into a Marama tool



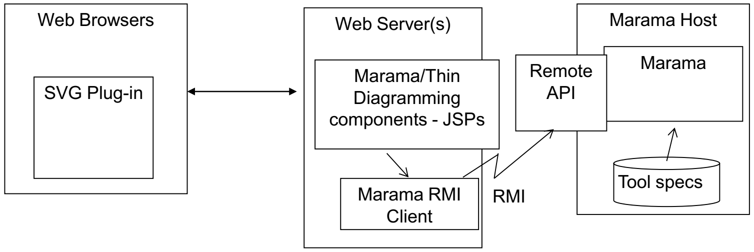
```
<?xml version="1.0" encoding="UTF-8"?>
<process name="EMailVotingProcess">
  <sequence>
    <receive partnerLink="Internal" portType="tns:processPort" operation="rec
    <invoke name="ReviewIssueList" partnerLink="Internal" portType="tns:inter
    <switch name="AnyIssueReady">
      <case condition="bpws:getVariableProperty(ProcessData,NumIssues)&gt;0">
        <invoke name="Discussion_Cycle_Derived_Process" partnerLink="Internal'
      </case>
      <otherwise>
        <empty/>
      </otherwise>
    </switch>
  </sequence>
</process>
```

# Thin-client/Remote interfaces

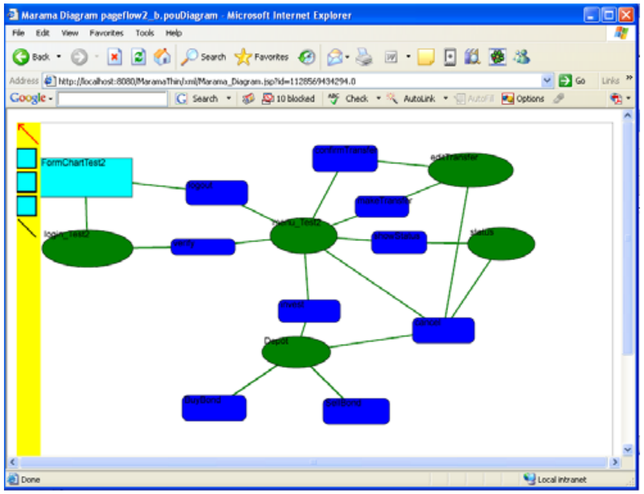
- Problems
  - Need to access Marama tools remotely on a variety of different devices
  - Need to drive Marama remotely
- Solutions
  - RMI interface to Marama API
  - Thin client interface for web browser interaction with any Marama generated tool (Penny Cao MSc thesis done)
  - Mobile phone interface for Marama generated tools (Joe Zhao MSc thesis done)
  - Laszlo based Flash or DHTML thin client interface (Tony Ip and Kelvin Lomberg 2007 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

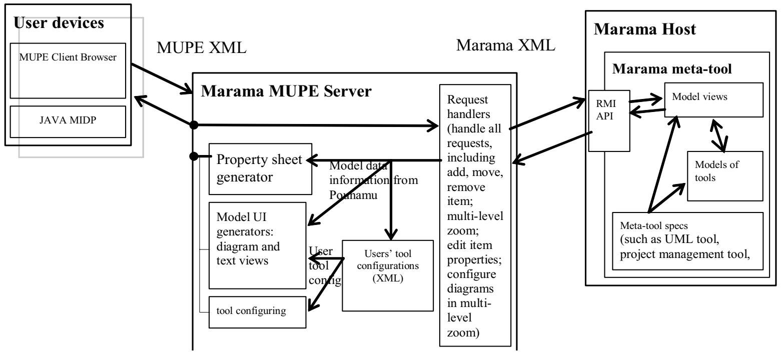


# Thin client interface example



# 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

Property sheet generator

Model data information from Pounamu

Model UI generators: diagram and text views

User tool config

Users' tool configurations (XML)

tool configuring

Request handlers (handle all requests, including add, move, remove item; multi-level zoom; edit item properties; configure diagrams in multi-level zoom)

Marama Host

Marama meta-tool

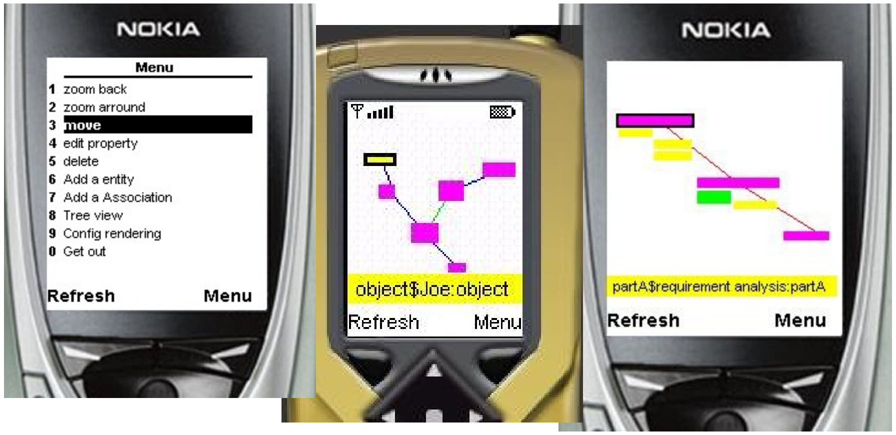
RMI API

Model views

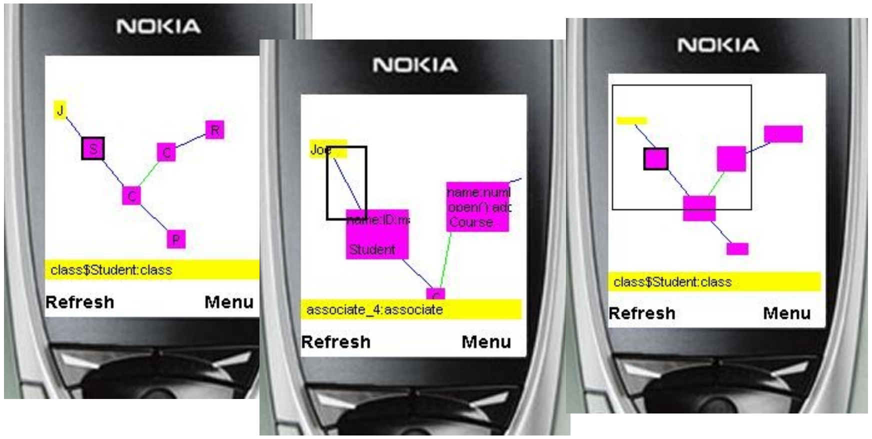
Models of tools

Meta-tool specs (such as UML tool, project management tool,

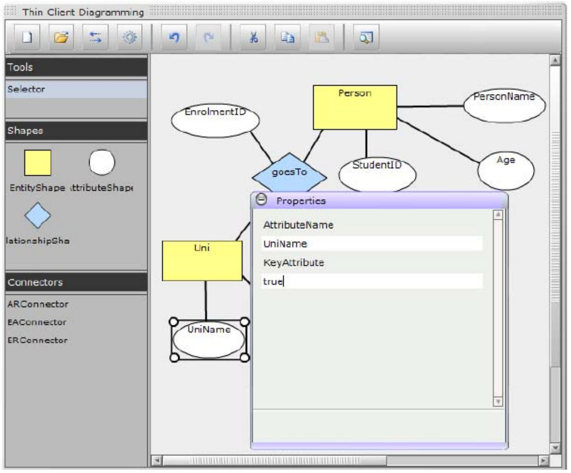
# Example MUPE interface usage



# Element zooming and overview



# Laszlo based Flash interface

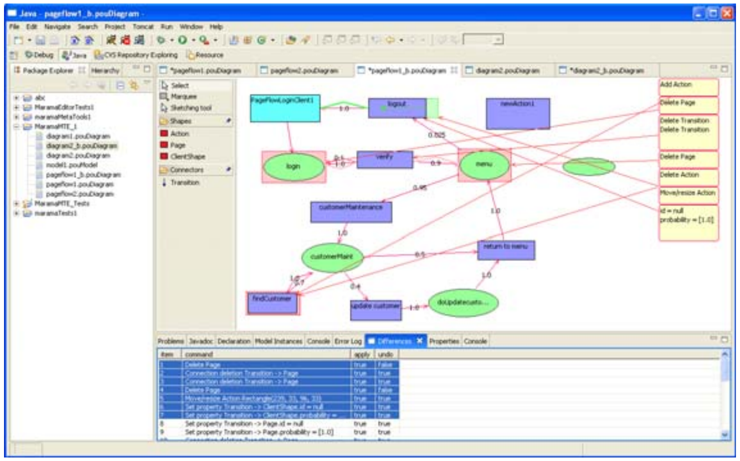


AttributeName
UniName
KeyAttribute
true

# 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/SVN via Eclipse workspace
  - Marama – differ & merger for DSVLs

# Visual Differ Example - Marama

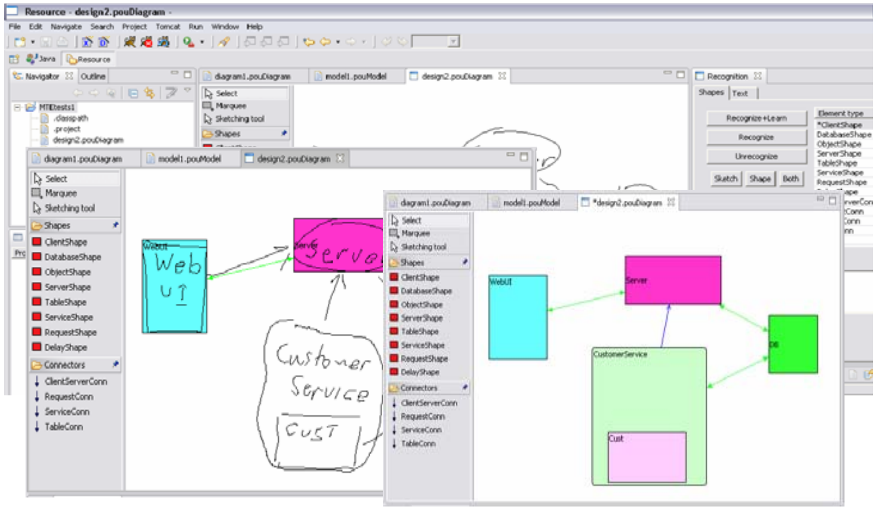


| item | command | apply | undo |
|---|---|---|---|
| 1 | Delete Page | true | false |
| 2 | Connection deletion Transition -> Page | true | true |
| 3 | Connection deletion Transition -> Page | true | true |
| 4 | Delete Page | true | false |
| 5 | Move/resize Action Rectangle(229, 33, 96, 33) | true | 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 |

# 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 (done- ICSE07 paper)
  - 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



# Other stuff

- Stuff we've got underway:
- Started 2008
  - Event handler library support (summer students)
  - Query views (summer students)
  - Layout support specn and implmn (Shan Yap BSc(Hons))
  - Open source hardening/productisation (with Sofismo)
  - Testing DSVLs (M Farid Jafaar)
  - Better extension point architecture
  - Rework Marama underlying EMF implementation (with Sofismo)
- Coming
  - DSVL knowledge base (Karen Li Postdoc)
  - Speech interface (touchy, feely interfaces ☺ )…

# 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
  - Hardened to point of commercial deployment of generated tools
    - Tools developed using Marama are in commercial use
  - Eighth year of use in CS732/SE450!
    - (Pounamu -> Marama)
- Plenty of scope to undertake projects/theses developing or applying Marama or its successors

# Where to next (bigger picture)?

- Better integration with workflow/ process/ knowledge management tools e.g. the "visual wiki" (see: thinkbase.cs.auckland.ac.nz for prototype)
- Handling (well) model evolution; collaborative modelling; cross-domain modelling; model integration
- Reusing others model checking, validation etc work
- Modelling vs visualisation – integration of the concepts via multiple views
- How do we design and validate DSVLs effectively?
- "End-user" DSVLs tools - much wider applications