Karen's part in COMPSCI 732

k.li@auckland.ac.nz; ext 89834

	Mon	Tue	Thu
Week 2	08/03/10 Marama	09/03/10 MS DSL Tools Assignment One Start	11/03/10 Marama Lab (GCL) – Jun Huh
Week 3	15/03/10 MS DSL Tools Lab (GCL)	16/03/10 Marama Extensions	18/03/10 MaramaDSL (Seminar Room)
Week 6			22/04/10 Assignment One Demo (GCL) Submission Due 23/04/10

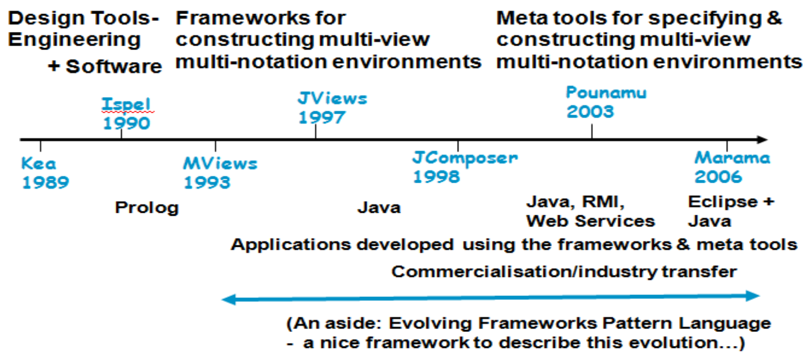
Marama

- Aim of section:
  - Examine Marama, a meta tool for constructing extensible DSVL environments
    - Very much beta software
- Contents
  - Historical development
  - Motivations/requirements
  - Marama overview
  - Applications

References
1. Grundy, J., et al., *Marama: an Eclipse Meta-toolset for Generating Multi-view Environments, in ICSE'08.*
2. Marama Wiki https://wiki.auckland.ac.nz/display/csidst/Welcome

History

- Original interest: Visual class diagramming tool ISPEL
- Led to long term interest in frameworks and tools for constructing such systems


Design Tools-Engineering + Software
Frameworks for constructing multi-view multi-notation environments
Meta tools for specifying & constructing multi-view multi-notation environments
Ispel 1990
JViews 1997
Pounamu 2003
Kea 1989
MViews 1993
JComposer 1998
Marama 2006
Prolog
Java
Java, RMI, Web Services
Eclipse + Java
Applications developed using the frameworks & meta tools
Commercialisation/industry transfer
(An aside: Evolving Frameworks Pattern Language - a nice framework to describe this evolution...)

Pounamu development

- Design and development of core system: Nianping Zhu, John Grundy, John Hosking
- Shape definer extensions: Xiaomin Tian (Project)
- Thin Client interface: Feng Luo (Project) Penny Cao (MSc)
- Collaboration interface: Akhil Mehra (Project & MSc)
- Web services interface: Therese Helland (MSc), Penny, Nianping, Akhil
- Mobile phone interface: Joe Zhao (MSc)
- Visual event handler definition: Karen Li (PhD), Kelvin Jin (MSc)
- Zoomable user interface: Karen Li (Project)

Marama development

- Developed Eclipse-based plug-ins (Marama) to read Pounamu DSVL tool specs and generate Eclipse-based tools (by Grundy, in 2005)
- Then decided to "retire" Pounamu (be thankful! ☺ - CS732 has used for 3 years and SE462 and SE710 two years ☺…)
- Development of Marama meta-tools incl. visual behaviour tools; improvement of editing tools
  - Karen Li (PhD), Jun Huh, John Grundy
- Various enhancements to Marama – Matthew Gatland, Hung Pham, Piran Tata …
- A few Marama based tools e.g. MaramaCritics – Norhayati Md. Ali (PhD)
- John G has hacked a few extensions to Marama (see later lecture on Marama Extensions) e.g. thin-client diagramming via SVG+JavaScript, collaborative work support via diagram diffing/merging, MaramaSketch hand-drawn support for diagram entry

Models, models everywhere…

- Software engineering:
  - OOA/D, requirements, processes, networks, tests, configurations, code, …
- Construction/Engineering/Comp Systems:
  - Structures, plant, plumbing/electrics, materials, …
  - VHDL, electromagnetics, processes/tasks, …
- Health:
  - Patient diagnoses, treatments, imaging, …
- Business:
  - Processes/workflow, financial, economic (!), …
- Others:
  - Families, friends/social/business networks, ...

Working with models

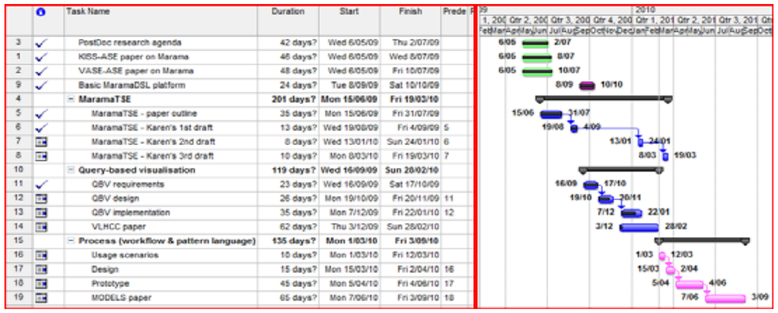
- Authoring, visualising, navigating, transforming, understanding, evolving, …
- Requires appropriate TOOLS to support these
- Tools must be usable, scalable, sharable, robust, extensible
- Ideally we want to provide *domain-specific visual languages* (DSVLs) to represent (parts of) models in "closeness of fit" to end user/domain
- We want tools to support these DSVLs
- BUT - building such DSVL modelling tools is HARD!

Meta tools

- A meta tool is a tool that allows you to define meta models which can be used to generate environments for modelling using instances of the meta models
- Most meta tools allow the definitions of:
  - meta models
  - visual notations
  - modelling views (notation mappings)
  - modelling behaviours
- Facilitate easy creation of domain-specific visual language (DSVL) tools

Exercise

- What meta model elements need to be defined, to generate an environment supporting modelling this:



		Task Name	Duration	Start	Finish	Prede
3	✓	PostDoc research agenda	42 days?	Wed 6/05/09	Thu 2/07/09	
1	✓	KISS-ASE paper on Marama	46 days?	Wed 6/05/09	Wed 8/07/09	
2	✓	VASE-ASE paper on Marama	48 days?	Wed 6/05/09	Fri 10/07/09	
9	✓	Basic MaramaDSL platform	24 days?	Tue 8/09/09	Sat 10/10/09	
4		− **MaramaTSE**	**201 days?**	**Mon 15/06/09**	**Fri 19/03/10**	
5	✓	MaramaTSE - paper outline	35 days?	Mon 15/06/09	Fri 31/07/09	
6	✓	MaramaTSE - Karen's 1st draft	13 days?	Wed 19/08/09	Fri 4/09/09	5
7	▦	MaramaTSE - Karen's 2nd draft	8 days?	Wed 13/01/10	Sun 24/01/10	6
8	▦	MaramaTSE - Karen's 3rd draft	10 days?	Mon 8/03/10	Fri 19/03/10	7
10		− **Query-based visualisation**	**119 days?**	**Wed 16/09/09**	**Sun 28/02/10**	
11	✓	QBV requirements	23 days?	Wed 16/09/09	Sat 17/10/09	
12	▦	QBV design	26 days?	Mon 19/10/09	Fri 20/11/09	11
13	▦	QBV implementation	35 days?	Mon 7/12/09	Fri 22/01/10	12
14	▦	VLHCC paper	62 days?	Thu 3/12/09	Sun 28/02/10	
15		− **Process (workflow & pattern language)**	**135 days?**	**Mon 1/03/10**	**Fri 3/09/10**	
16	▦	Usage scenarios	10 days?	Mon 1/03/10	Fri 12/03/10	
17	▦	Design	15 days?	Mon 15/03/10	Fri 2/04/10	16
18	▦	Prototype	45 days?	Mon 5/04/10	Fri 4/06/10	17
19	▦	MODELS paper	65 days?	Mon 7/06/10	Fri 3/09/10	18

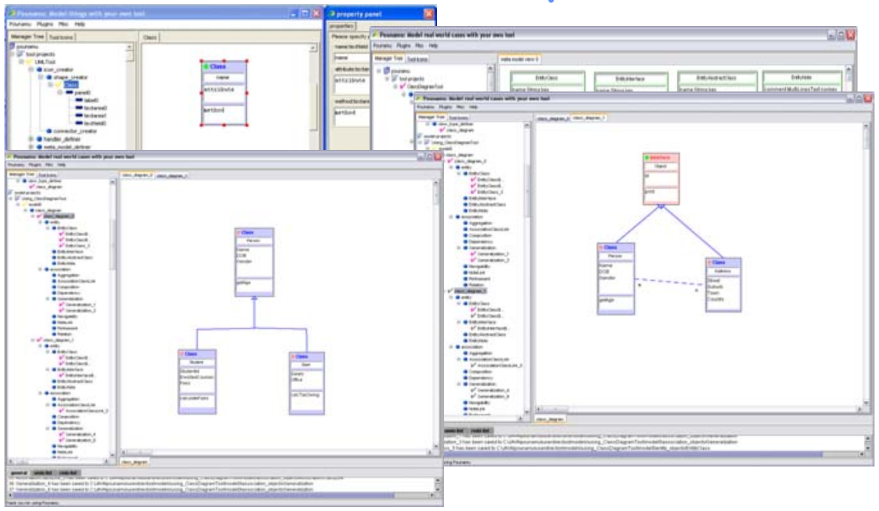
- In pairs come up with a list (2-3 mins)

- In pairs of pairs exchange and discuss your lists (2 mins)

Pounamu

- 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).

Pounamu examples



Pounamu problems

- Provided some good things:
  - Quick to build prototype DSVL tools
  - Used for wide range of exemplar systems e.g. software architecture, performance engineering tools; UML tools; UI design tools; project management tools; software process modelling tool; stats design tool; Jimi Hendriks GuitarGeek.com set-up DSVL… ☺
- BUT
  - Easy(ish) to define shapes, meta-model views BUT event handlers required Java scripting/APIs
  - All custom-built so lots of effort to maintain ourselves
  - Rather clunky UI
  - Good extension e.g. full web services API but all our own proprietary APIs

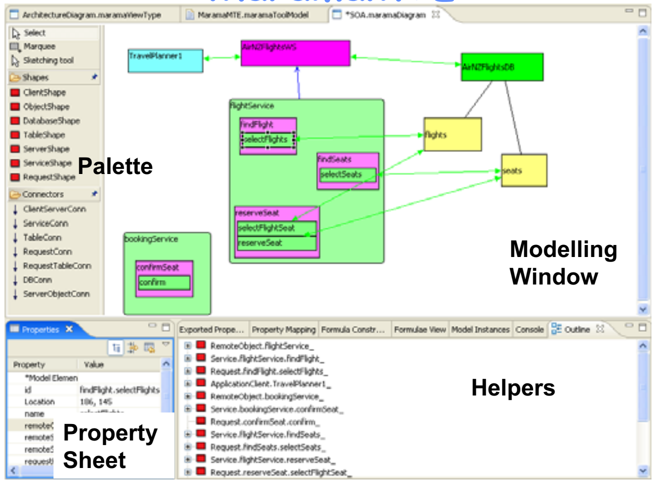
Marama – some key goals

- Make modelling tool implementation easier for:
  - Experienced domain *modellers* (may not be developers!)
  - Familiar with basic *modelling* concepts
    - Eg EER, OCL, meta models
  - Construct basic modelling tools within 1 day
    - Plus time for backend code generators etc
- Leverage strength of Eclipse platform
  - Standalone Pounamu left us with too much to support infrastructure to develop e.g. save/load, XML, UI integration, remoting
    - Make use of EMF, GEF, JET, events, etc
  - Eclipse community & open source attractive
- Paper at ASE06 on early version of Marama
  - Used Pounamu meta-tool
  - Realised tools in Eclipse using Marama runtime plugin
- Paper at ICSE08 on (more or less) latest Marama toolset

Marama – some key requirements

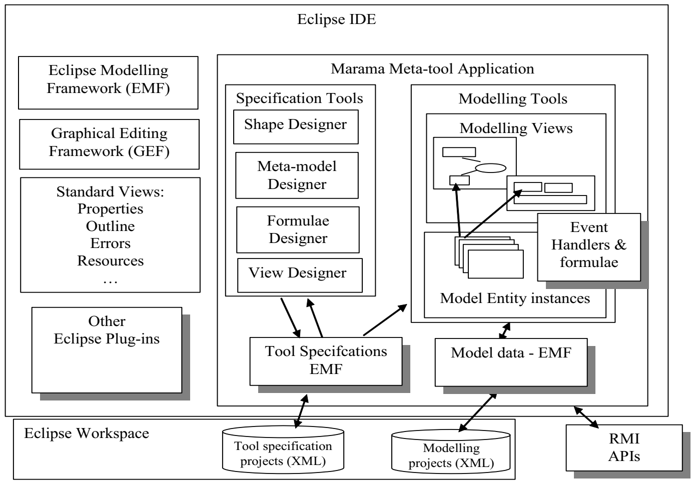
- Need to be able to specify and generate:
  - Meta model
    - represents the target model elements
  - Icons and connectors
    - visual representation(s) of model
  - Views and view to model mappings
    - View – model consistency
  - Behaviour
    - Constraints, operations
  - Model transformations
    - Backend code generation
    - Tool integration
  - Tool deployment
    - Scalable, sharable, usable, intelligent, ... tools

Generated modeller example: MaramaMTE



RemoteObject.flightService_
Service.flightService.findFlight_
Request.findFlight.selectFlights_
ApplicationClient.TravelPlanner1_
RemoteObject.bookingService_
Service.bookingService.confirmSeat_
Request.confirmSeat.confirm_
Service.flightService.findSeats_
Request.findSeats.selectSeats_
Service.flightService.reserveSeat_
Request.reserveSeat.selectFlightSeat_

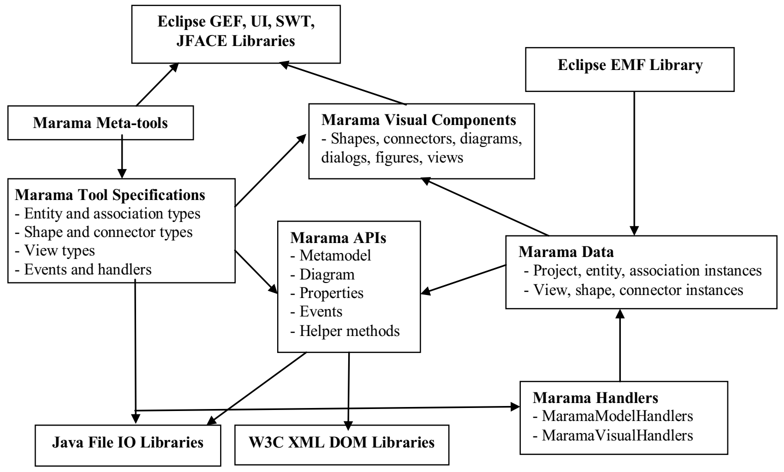
Marama approach



Marama components

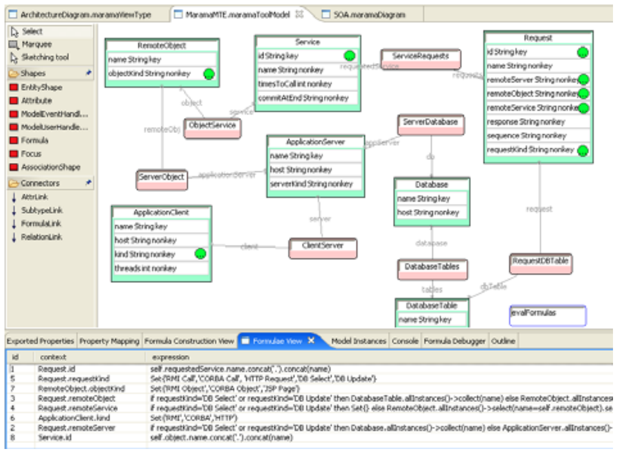
- Tool projects
  - Meta model designer tool
    - Specifies tool meta models incl. constraint-based formulae over the model
  - Shape and connector creator tool
    - Used to define icons, connectors and associated properties
  - View type designer tool
    - Specifies an editor for a set of shapes, connectors and handlers, and their relationship to a meta model
  - Event handlers
    - Specifies dynamic behaviour in response to events (eg shape creation). Currently done via formulae, "Kaitiaki" event specification tool or Java code using Marama APIs
- Model projects
  - Instances of a specified tool in use

Component structure



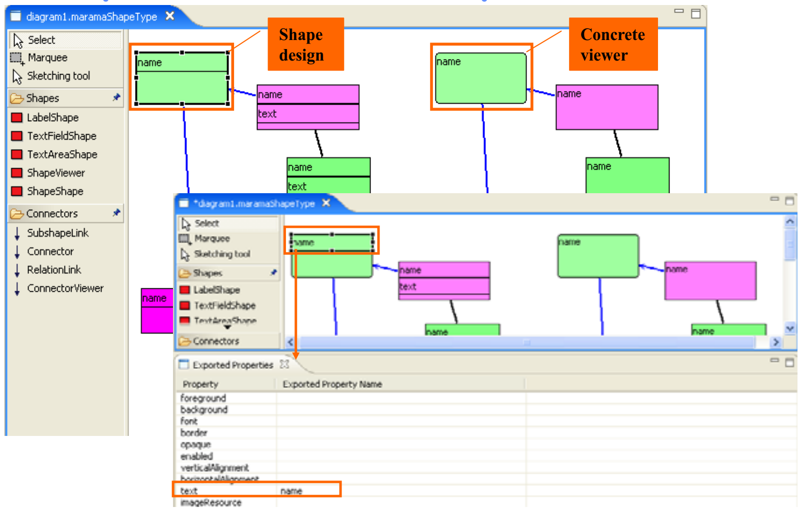
Metamodel specification

- EER
  - Entities
  - Relationships
  - Subtyping
  - Roles
  - Attributes
  - Keys
- OCL constraints (see later lecture on Marama Extensions)
  - Attribute calcns
  - Invariants
  - Cardinalities



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
4	Request.remoteService	if requestKind='DB Select' or requestKind='DB Update' then Set{} else RemoteObject.allInstances()->select(name=self.remoteObject).se
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()-
8	Service.id	self.object.name.concat('.').concat(name)

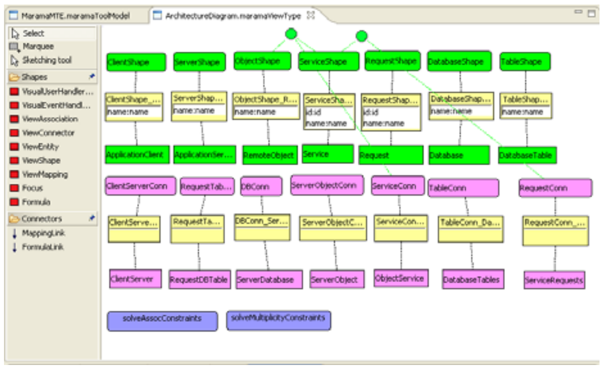
Shape and connector specification



Property	Exported Property Name
foreground	
background	
font	
border	
opaque	
enabled	
verticalAlignment	
horizontalAlignment	
text	name
imageResource	

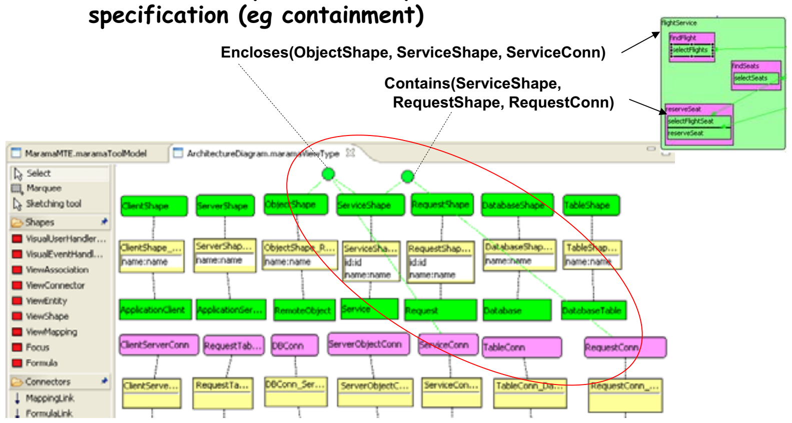
View and view-model mapping specn

- Elements in view
- Mappings
  - Entity to shape
  - Relationship to connector
  - Model attribute to visual property



Visual layout constraints in views

- Can add some predefined layout constraints in view specification (eg containment)

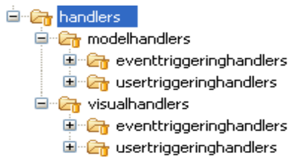

Encloses(ObjectShape, ServiceShape, ServiceConn)
Contains(ServiceShape, RequestShape, RequestConn)

Marama events

- Defined in package nz.ac.auckland.cs.marama.model.events
- Built-in events
  - Model semantic events
    - entityAdded, entityDeleted, entityUpdated, associationUpdated,propertyUpdated ...
  - Low-level visual events
    - shapeAdded, connectorAdded, shapeDeleted,connectorDeleted,shapeMoved, shapeResized ...
- Custom user event
  - Context menus

Marama event handling structure

- Marama incorporates event notifications and event handlers
- Events are notified by the event generators and propagated to all the event handlers at runtime
- Model handlers - specify reactions to model events (e.g. entity/association changes)
- Visual handlers - specify reactions to visual view-based events (e.g. shape/connector changes).
- Both model and visual handlers are sub-typed further by specialising them to event triggering and user triggering (via user menu-click action) natures.

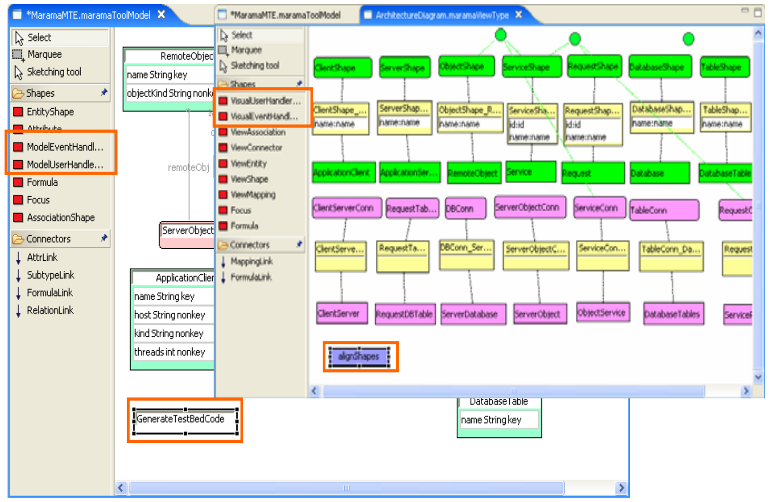


```
handlers
  modelhandlers
    eventtriggeringhandlers
    usertriggeringhandlers
  visualhandlers
    eventtriggeringhandlers
    usertriggeringhandlers
```

Registering event handlers

- A model event/user handler must be
  - first defined in a meta model designer (by drag-dropping a ModelEventHandler/ModelUserHandler icon from the palette to the meta-model diagram), and
  - then coded as a handler class (and saved in the corresponding handlers' folder of the tool's source code repository).
- The name of the diagram handler icon and that of the Java class must be consistent in order to get the handler registered and fired correctly.
- Visual event/user handlers are all defined in a similar way but in a view type designer.

Registering event handlers



Model handlers

- A model event/user handler is defined as a subclass of MaramaModelHandler (in package nz.ac.auckland.cs.marama.model.events), saved as a Java file in the corresponding directory.
- The method "public void notifyChanged(Notification notification)" receives all the event notifications and implements the reaction behaviour for a filtered event or list of events.

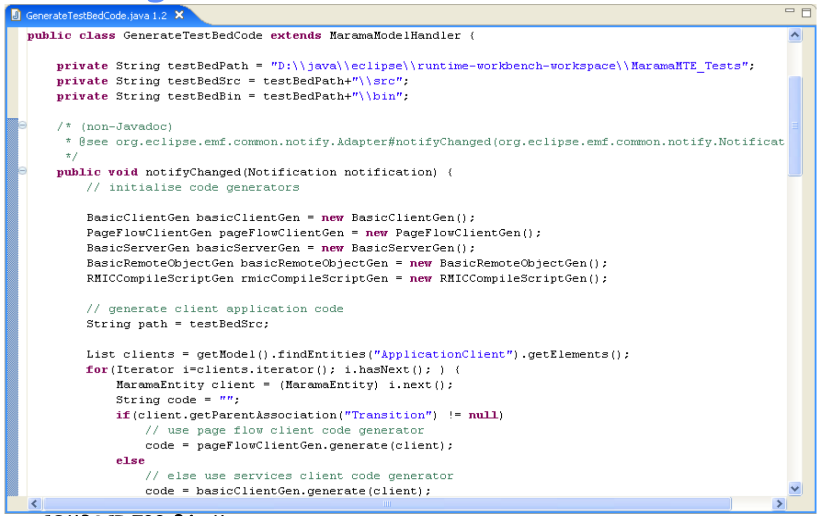
```
public class AModelEventHandler  extends MaramaModelHandler  {
      public void notifyChanged(Notification  notification)  {
            /** Reaction code goes here. */
      }
      public String getName() {
            return "A model event handler"; // handler name/description
      }
}
```

Visual handlers

- A visual event/user handler is defined in the same way but extends MaramaVisualHandler (in package nz.ac.auckland.cs.marama.model.events).

```
public class AVisualEventHandler  extends MaramaVisualHandler  {
      public void notifyChanged(Notification  notification)  {
            /** Reaction code goes here. */
      }
      public String getName() {
            return "A visual event handler"; //handler name/description
      }
}
```

Coding event handlers for behaviours



GenerateTestBedCode.java 1.2

```
public class GenerateTestBedCode extends MaramaModelHandler {

    private String testBedPath = "D:\\java\\eclipse\\runtime-workbench-workspace\\MaramaMTE_Tests";
    private String testBedSrc = testBedPath+"\\src";
    private String testBedBin = testBedPath+"\\bin";

    /* (non-Javadoc)
     * @see org.eclipse.emf.common.notify.Adapter#notifyChanged(org.eclipse.emf.common.notify.Notificat
     */
    public void notifyChanged(Notification notification) {
        // initialise code generators

        BasicClientGen basicClientGen = new BasicClientGen();
        PageFlowClientGen pageFlowClientGen = new PageFlowClientGen();
        BasicServerGen basicServerGen = new BasicServerGen();
        BasicRemoteObjectGen basicRemoteObjectGen = new BasicRemoteObjectGen();
        RMICCompileScriptGen rmicCompileScriptGen = new RMICCompileScriptGen();

        // generate client application code
        String path = testBedSrc;

        List clients = getModel().findEntities("ApplicationClient").getElements();
        for(Iterator i=clients.iterator(); i.hasNext(); ) {
            MaramaEntity client = (MaramaEntity) i.next();
            String code = "";
            if(client.getParentAssociation("Transition") != null)
                // use page flow client code generator
                code = pageFlowClientGen.generate(client);
            else
                // else use services client code generator
                code = basicClientGen.generate(client);
```

Coding event handlers for behaviours

- The handler code can include:
  - events of interests
  - queries of model/diagram states
  - filters on a collection of data
  - state changing actions (create/update/delete) on Marama model/visual elements

- model.findEntities(entType.getName())
- entity.getParentEntities("Realization")
- entity.getAttributeValue("name")
- diagram.getViewType()
- diagram.getChildren()
- diagram.getConnectors()
- ...

A few examples can be derived from code at nz.ac.auckland.cs.marama.handlerlibrary.helper

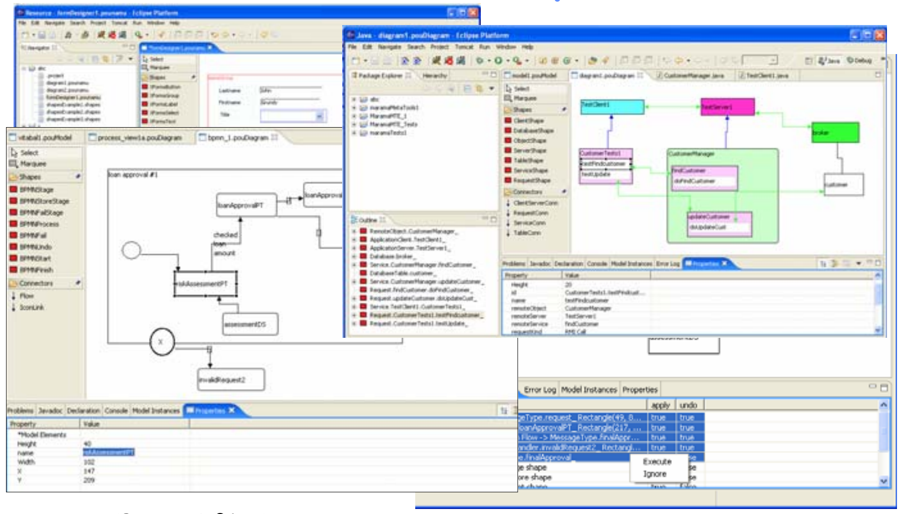
- Exploit Marama EMF and its implementations

- See later lecture on Marama Extensions for visual behaviour specification approaches

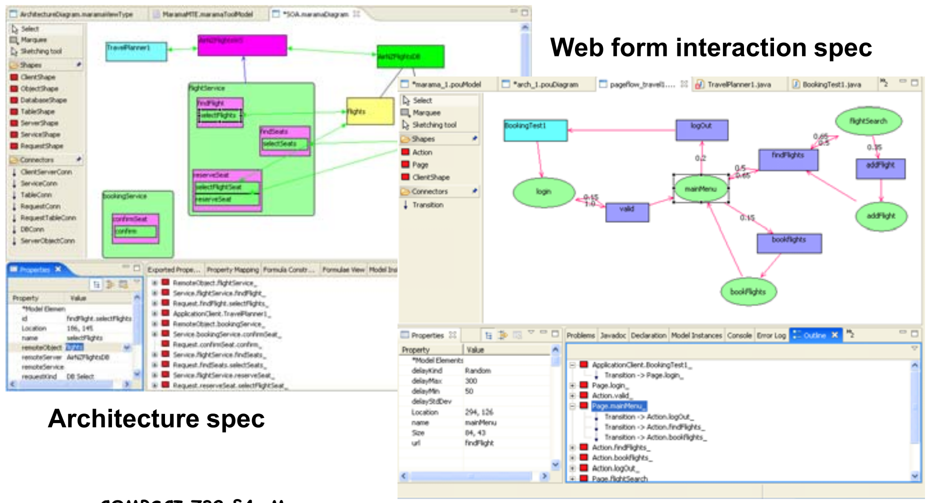
Marama examples

- Marama has been used to develop:
  - Marama meta-tools themselves ☺ – Karen Li PhD
  - MaramaMTE architecture modelling and performance engineering – Rainbow Cai PhD
  - MaramaEML enterprise modelling tool, BPMN tool – Richard Li PhD
  - MaramaDPML design pattern tool – David Maplesden, John Hosking and John Grundy – book chapter
  - Healthcare plan specification (& mobile deployment) - Abizer Khambati Msc
  - Various industry rapid prototypes e.g. UI design tool, business process modelling tool – John Grundy, consulting work

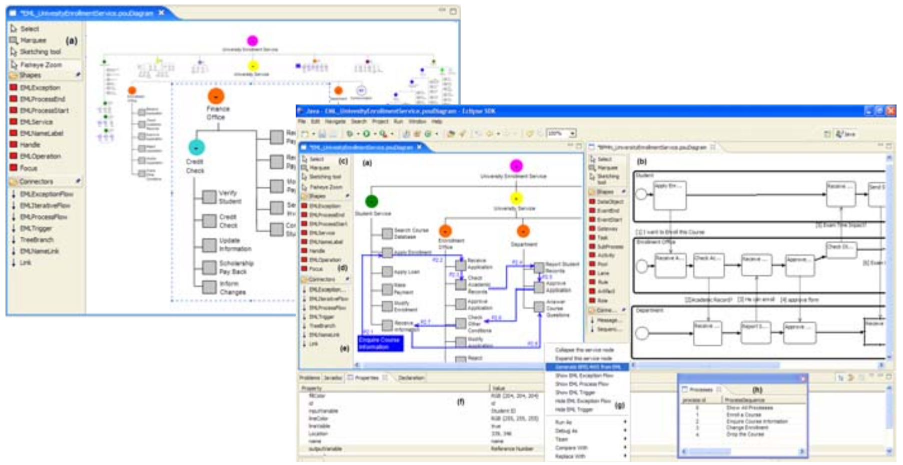
Marama examples



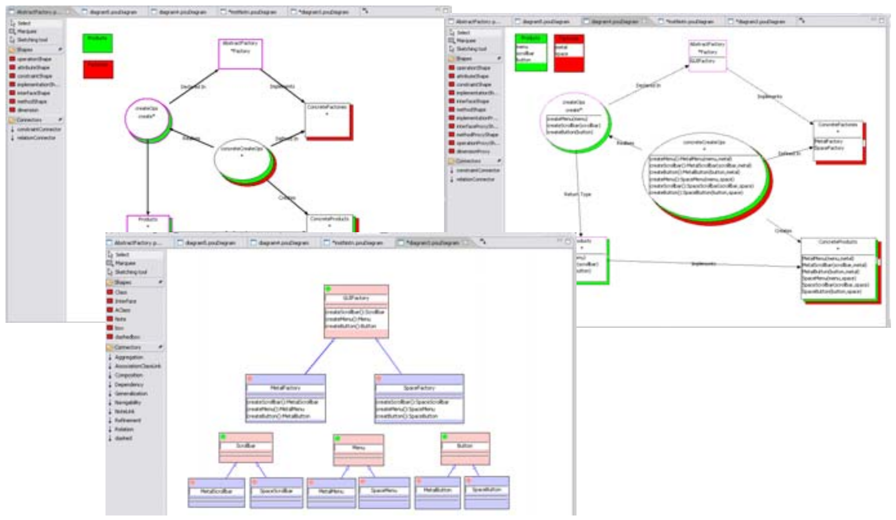
MaramaMTE – performance eng tool



MaramaEML – enterprise modelling (best demo paper ASE2008)



MaramaDPML tool – design patterns



VCPML & VPAM – health care plans